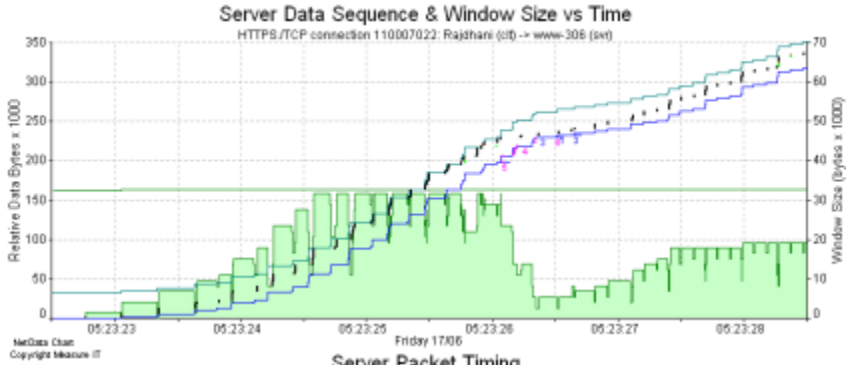
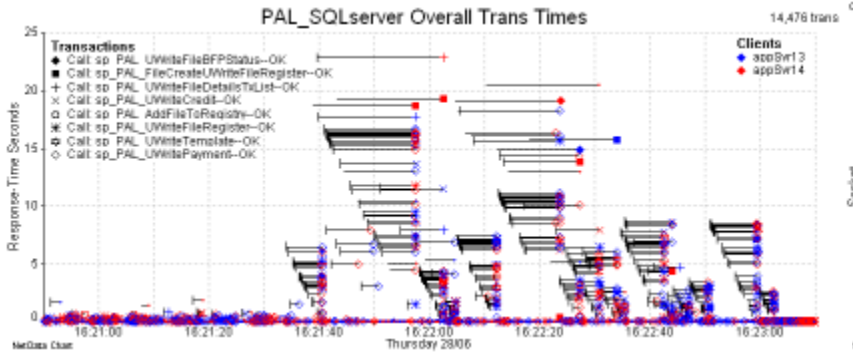
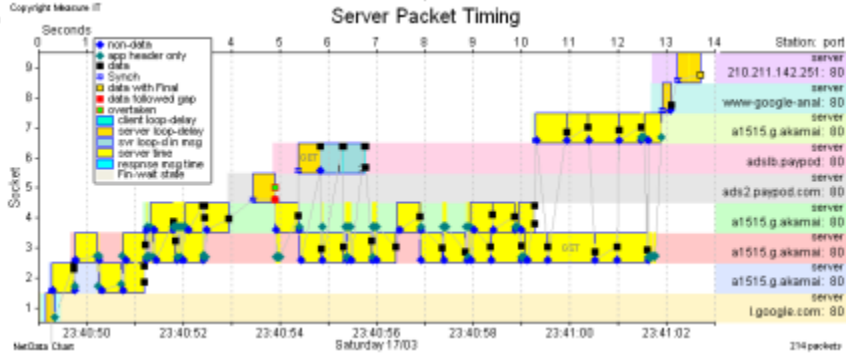
"Measure what is measurable, and make measurable what is not so" **Galileo Galilei**

NetData

visualising IT performance







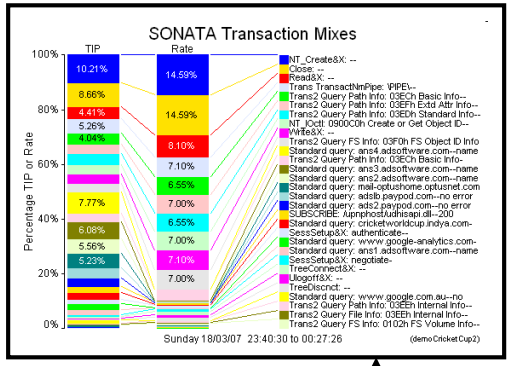
The new way to diagnose network-, application-, and the most complex system-performance problems

Summary

- NetData reconstructs system operation by analysing network traffic captured by any type of sniffer
- It builds a comprehensive database of system activity:

  packets, transactions, transaction types, connections, dialogues, network events, protocol abnormalities, application errors, queue lengths, traffic volumes and other statistics

- The analyst can load into a charting module the records of any data type, and any time range, that relate to any clients and servers or meet any search criteria
- NetData's system of charts and tables
  - are designed to reveal graphically all signs of stress and poor performance;
  - are synchronised to give different views of the same system behaviour;
  - support many different overlays for correlations;
  - are highly scalable and versatile;
  - are easy to navigate between charts and supporting tables.
- NetData presents detailed and accurate pictures of system operation that can be drawn only from network traffic

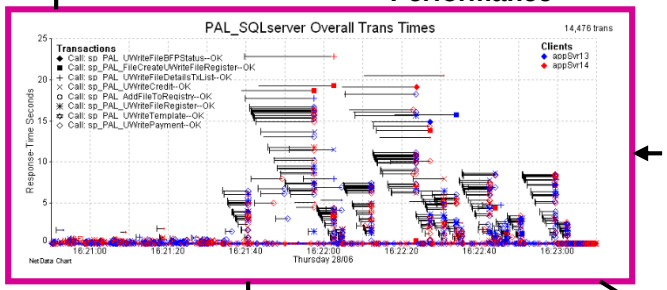


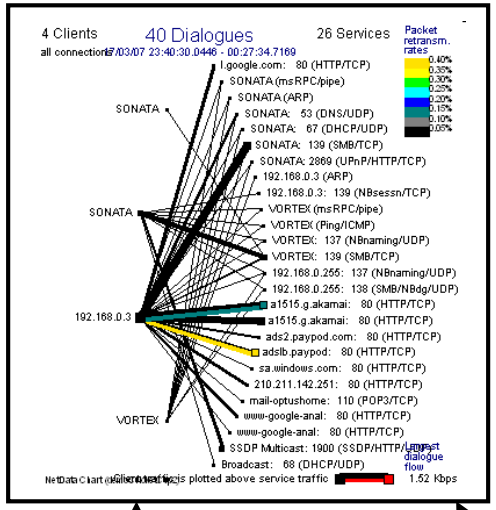
Slave chart

Slave tables

- Server Stats
- Trans-Type Stats
- Transactions
- Connections
- Events
- Event Stats

Master chart **Performance**

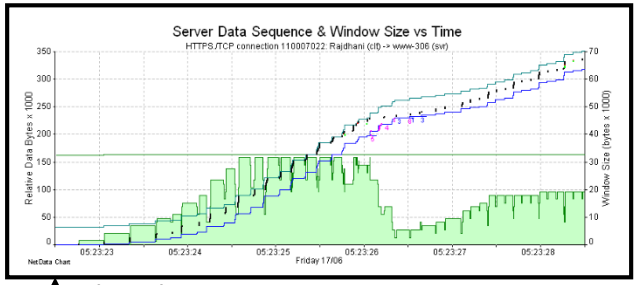




Slave tables

- Dialogues
- Services
- Clients
- Nodes
- (App) Types

Optional time-of-day alignment between master charts

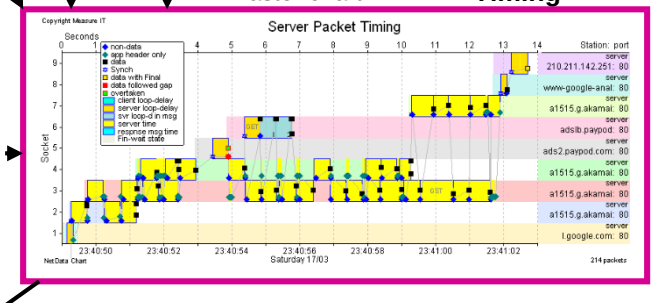


Slave chart

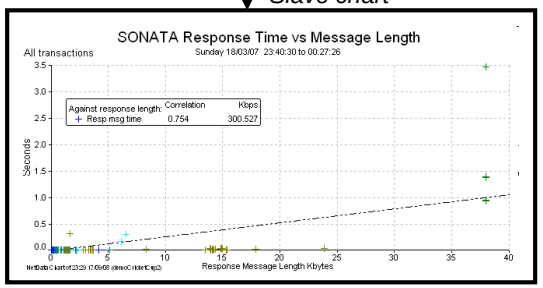
- Packets
- Connections

Slave tables

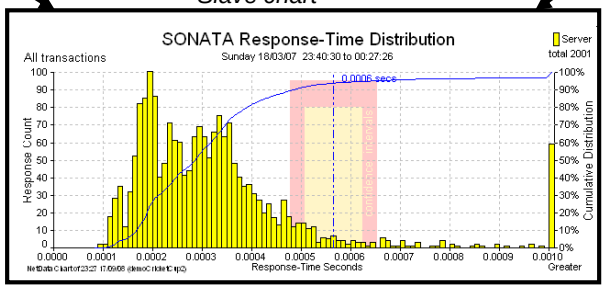
Master chart **Timing**



Slave chart



Slave chart



NetData Charts & Tables

The material in many of the following charts will appear to be complex, well beyond what a network engineer is normally expected to wrestle with. That is simply because modern IT systems are themselves complex, with sophisticated operating systems, applications and protocols, in servers and networking equipment – all normally hidden from the user. Systems are prone to an infinite range of weaknesses, faults and problems, whether caused by poor design, software bug, hardware failure, or an inappropriate configuration parameter.

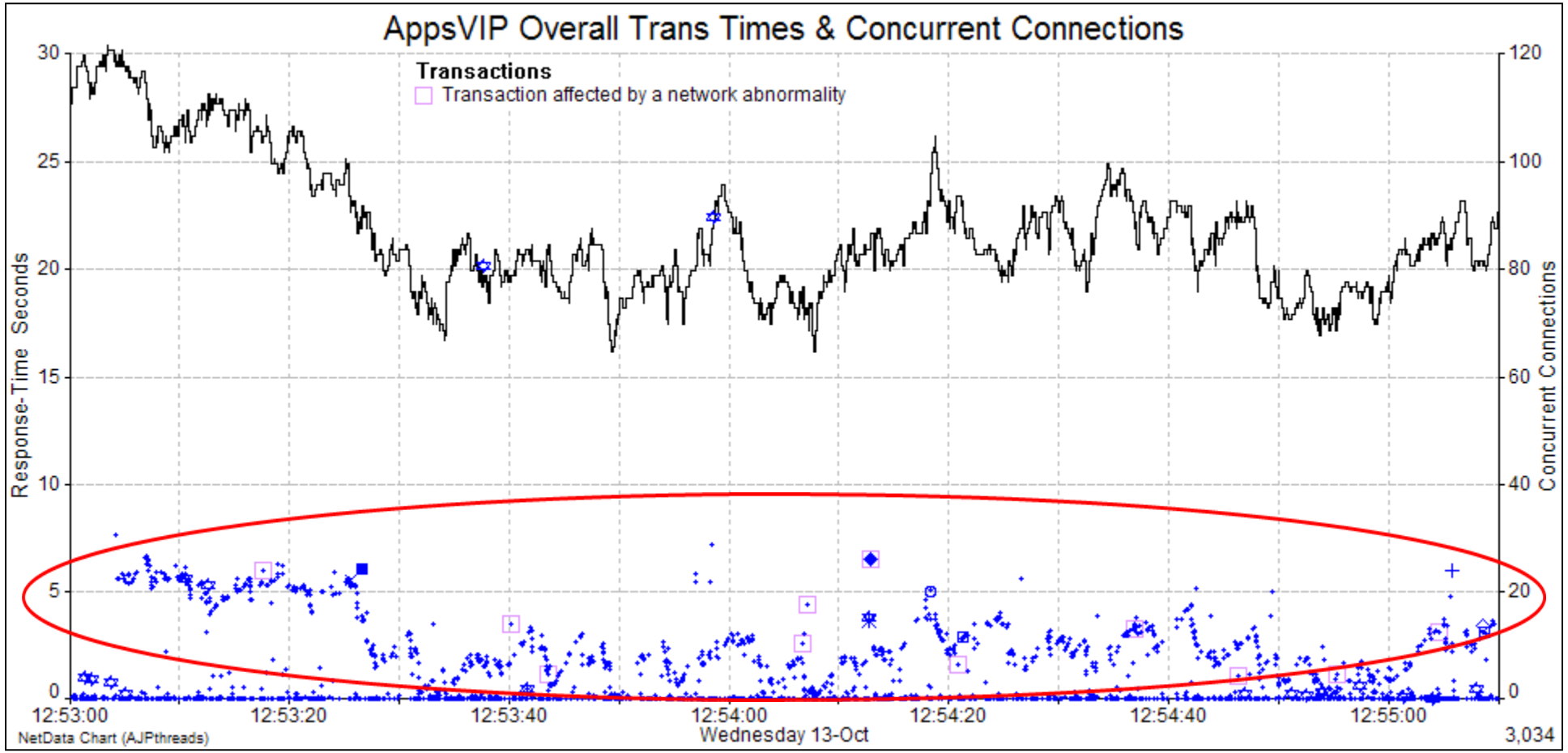
A problem can be fixed only by those who know how the system or module is meant to work, and that often presents a severe challenge to the diagnostic worker. NetData exposes unhealthy behaviour and helps close the gap between what network traffic can reveal and what an expert needs to fix a problem. It is no longer acceptable for the analyst to stop after declaring a network problem because there are many retransmissions, or after declaring an application problem because there is a long server-processing time.

The following case studies and examples have been chosen to demonstrate NetData's breadth and *depth.*

Response-Time Patterns, Queues and Ceilings

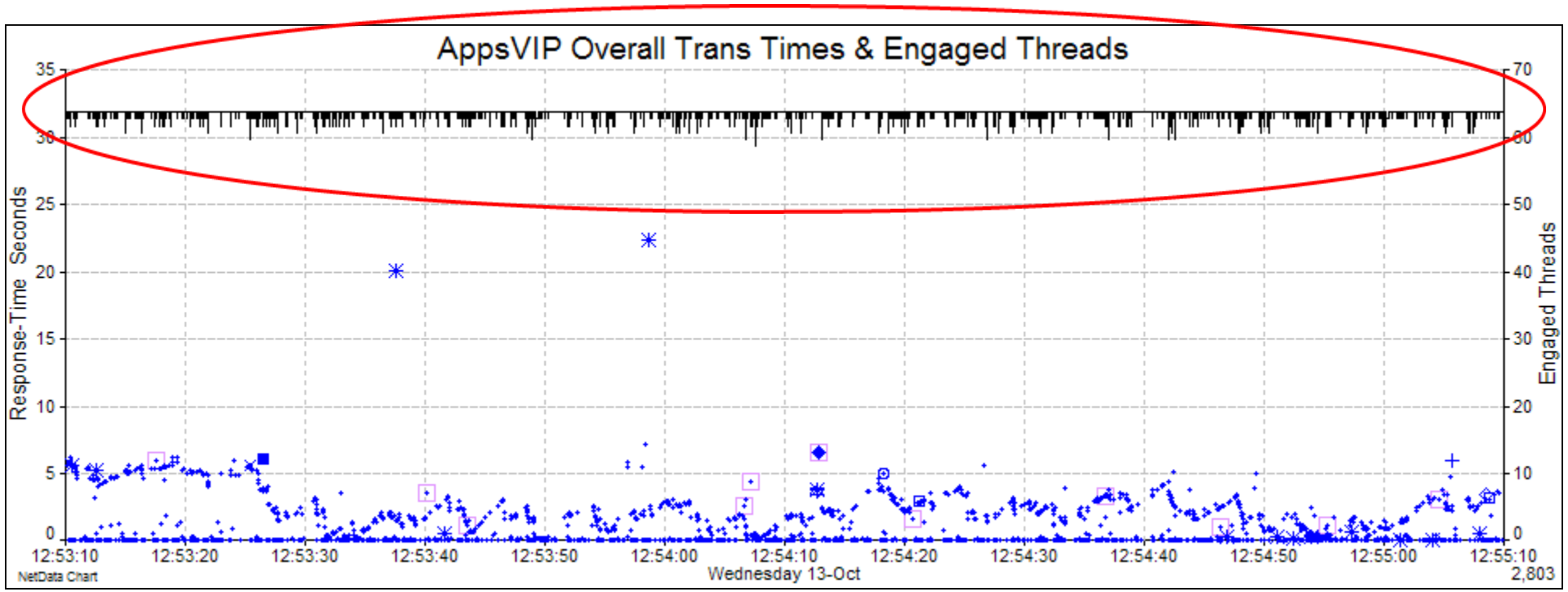
NetData's performance chart displays response times and is the best type of chart to corroborate a performance problem reported by others. It has many overlays:

- Transaction response times
- Client-preparation or think times
- Port-recycle times
- Transaction rates
- Concurrent connections
- Concurrent transactions (Transactions In Progress)
- Traffic volumes by protocol type or IP address
- Occurrence of significant network events and application errors
- Plots of variable data extracted from transaction messages



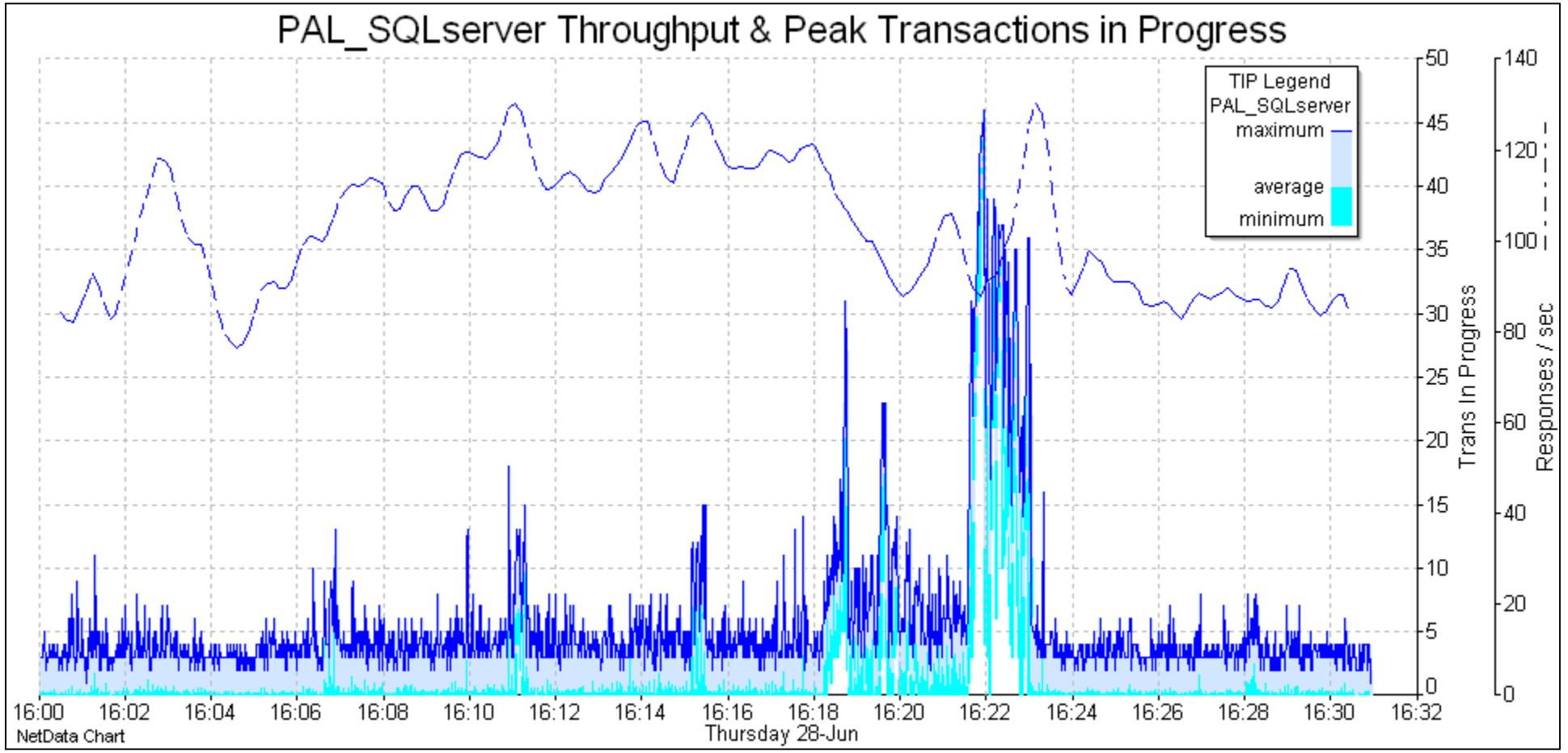
NetData **reconstructs and characterises all transactions**, including requests without responses and other types of failure. When individual response times are plotted the patterns can reveal many different problems. Averages are available for **trend analysis** and **capacity planning**, but they hide the diagnostic value.

A ‘wavy’ band of response-time markers as on this chart is evidence of a transaction queue whose average waiting time is indicated by the varying height of the marker band. The types of transactions caught in the queue help to identify the saturated resource causing the bottleneck.



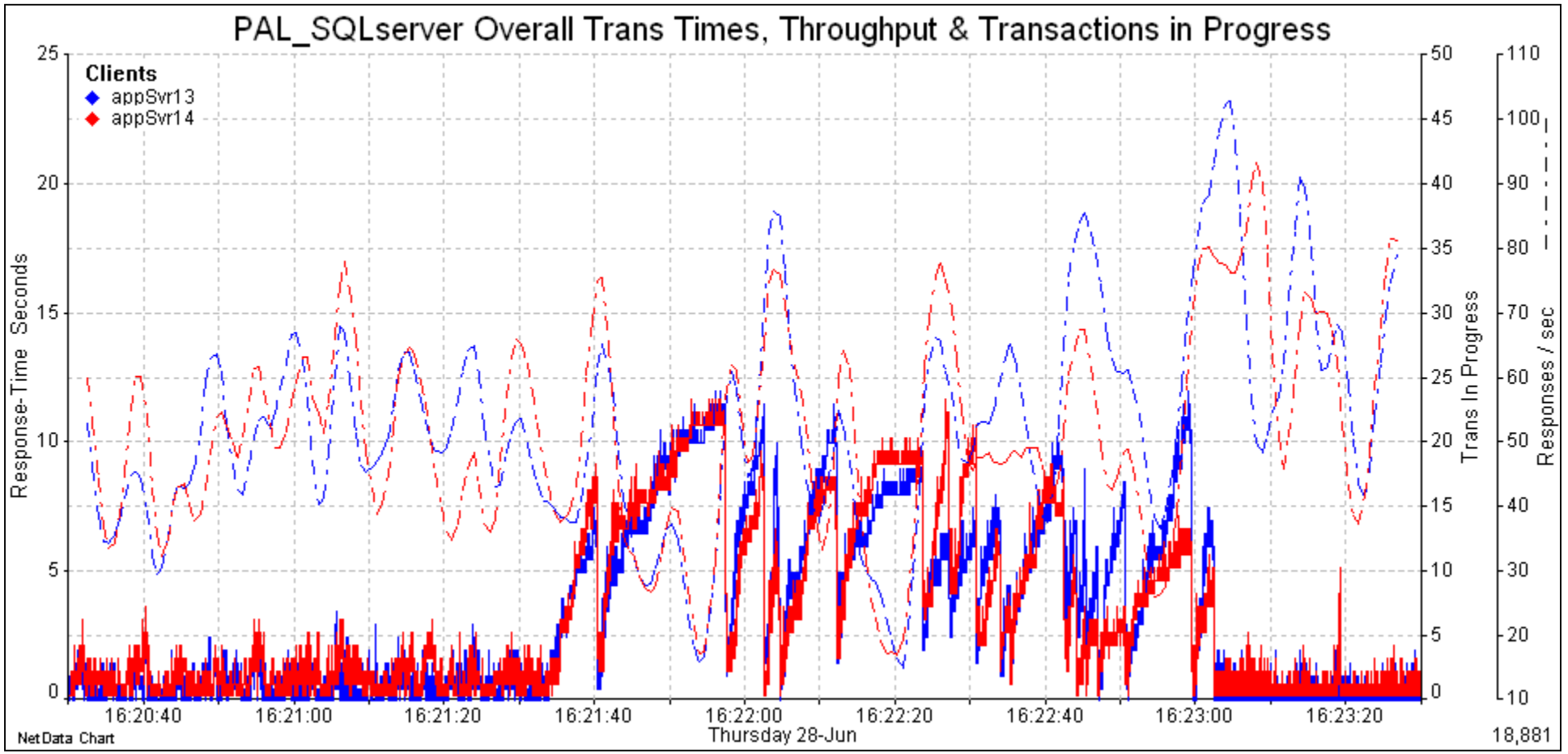
Overlaid graphs of the numbers of concurrent connections illustrate different aspects of system behaviour.

For this graph a connection was counted only when it carried its first server data packet, and the resulting number of concurrent connections is an estimate of the number of **engaged application threads**. It shows the system continually hitting a ceiling of 64 threads, and that is the bottleneck that caused the transaction queue and its large response times.



While several tools can display variations in server traffic volumes, very few problems produce a noticeable change in traffic volume. NetData too can analyse **100 GB of traffic or more** in a contiguous sequence of capture files, and can display traffic volumes, but because it records every transaction in its database it can also calculate and display queue lengths, and they are very sensitive indications of even the slightest processing pauses.

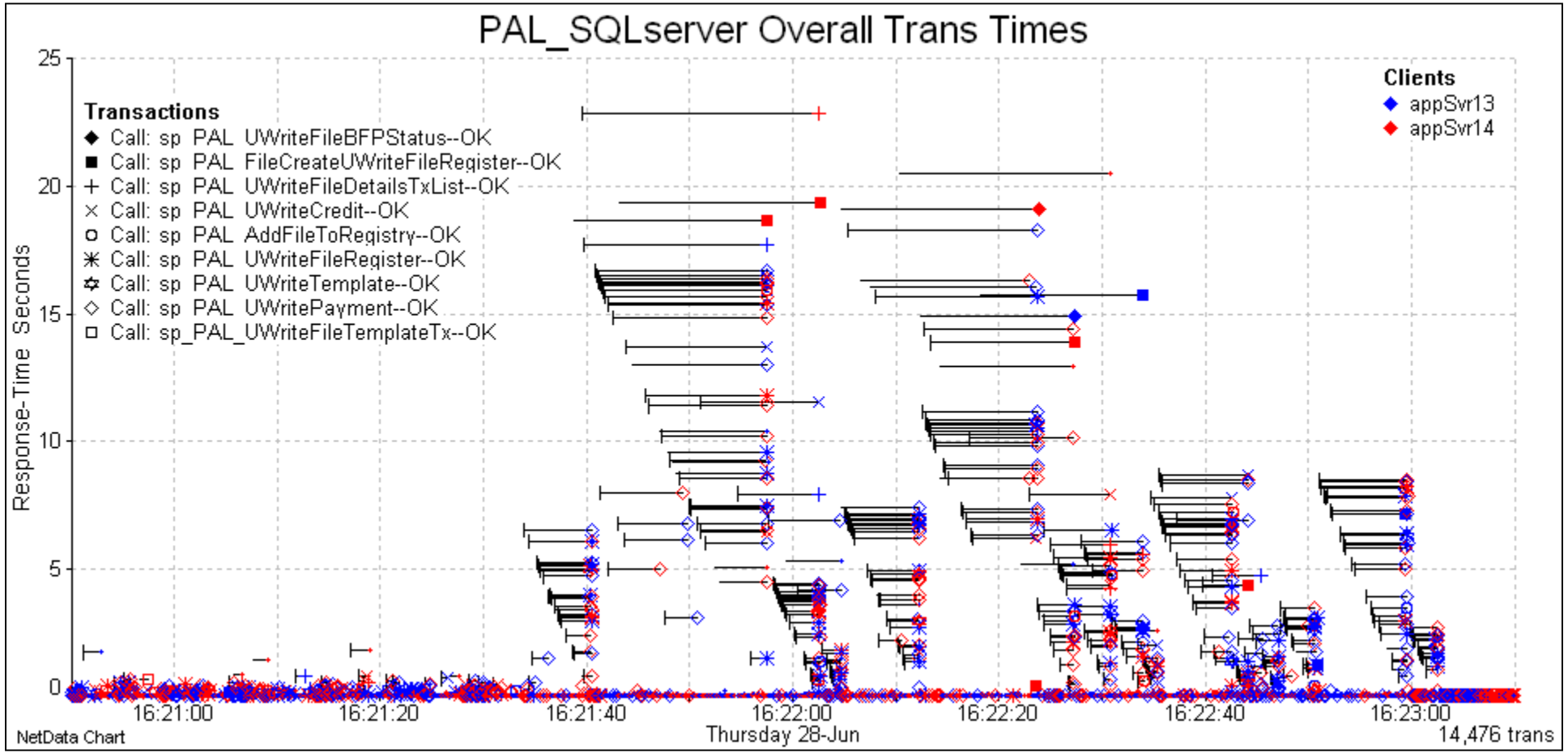
To locate periods of abnormal performance among millions of transactions, this chart plots the high-water mark for **queue-lengths in successive intervals throughout the whole capture period**.



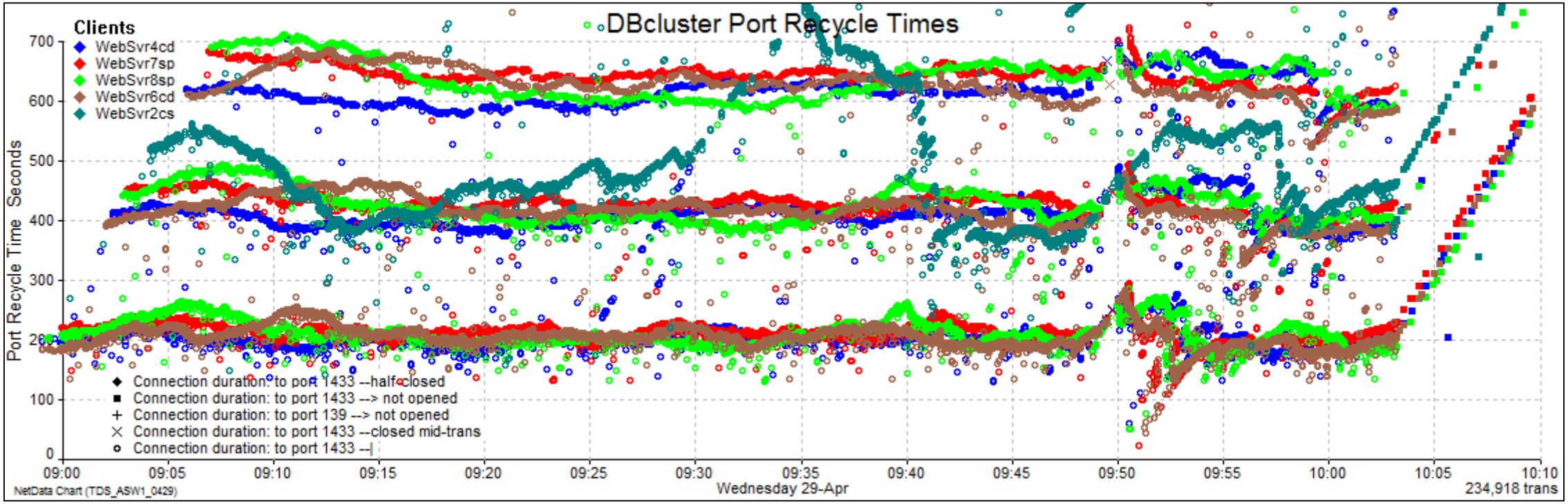
Graphs of transaction queue lengths (transactions in progress) illuminate several aspects of system behaviour. The appearance of abnormally long queues makes it easy to **find a bottleneck**, even in large online systems with hundreds of web servers, app servers, backend servers, load balancers, firewalls and routers.

Here, gradual rises in queue lengths to abnormal heights, and sudden queue clearances, **indicate blockages**.

Graphs of the transaction rates (resp/sec) from two app servers to a database confirm satisfactory load balancing.



Towers of markers here indicate **database blockages**, and the types of transactions caught in the blockages indicate which tables were locked and what commands caused a blockage. Overlapping towers indicate that locks were placed only on parts of a table.



Markers here indicate **port-recycle times**, the intervals between a connection's closing and its re-opening (from the same port). If a port is re-used too quickly the server might have the connection still in its **Time-Wait state** and ignore new connection requests. For systems that don't use persistent connections in the backend this mechanism creates a **bottleneck** that can be very difficult to identify because transaction delays and failures have a random nature.

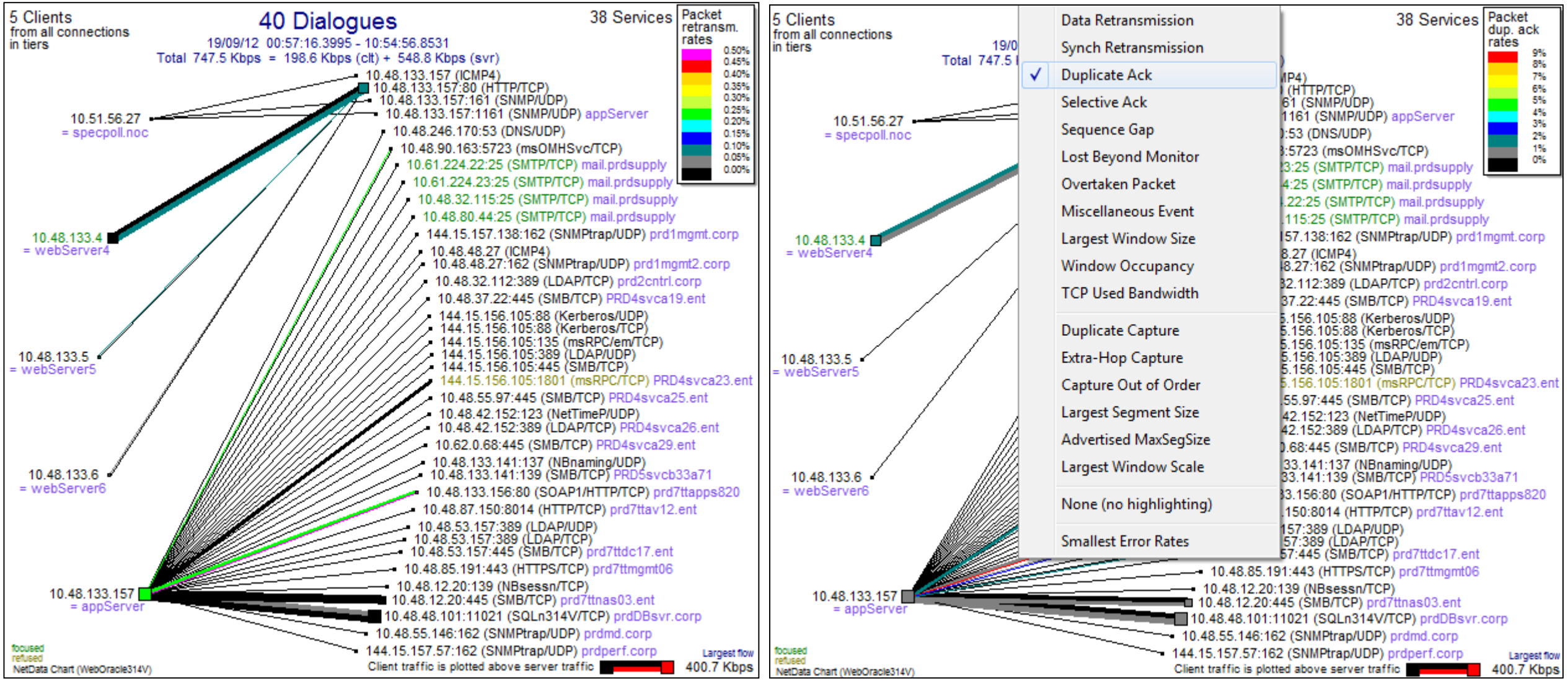
The markers fall into bands at heights that indicate the time a client machine takes to cycle through its range of ephemeral ports. Bands at different heights occur when a port isn't directed to the same server in every cycle; markers scatter below their bands when connections are open for longer times. This chart indicates a likely problem at 09:51 when many recycle times were less than a server's default Time-Wait time of 120 seconds. Recycle times rose after 10:03 when the server ignored all new connection requests while restarting.

Dialogues and Network Abnormalities

Usually the first chart to be examined after analysing capture files, a dialogue chart shows:

- all the captured dialogues;
- the servers, their services and application protocols;
- their clients;
- the top talkers; and
- the presence of any types of network abnormality.

A dialogue chart can give the **network a clean bill of health** at a glance.



Dialogue charts not only **distinguish clients from servers** but also identify individual ports and their protocols.
They highlight dialogues with the largest error rates and other statistics.
Every network abnormality can be investigated with a packet-timing chart.

Timing and Waterfalls

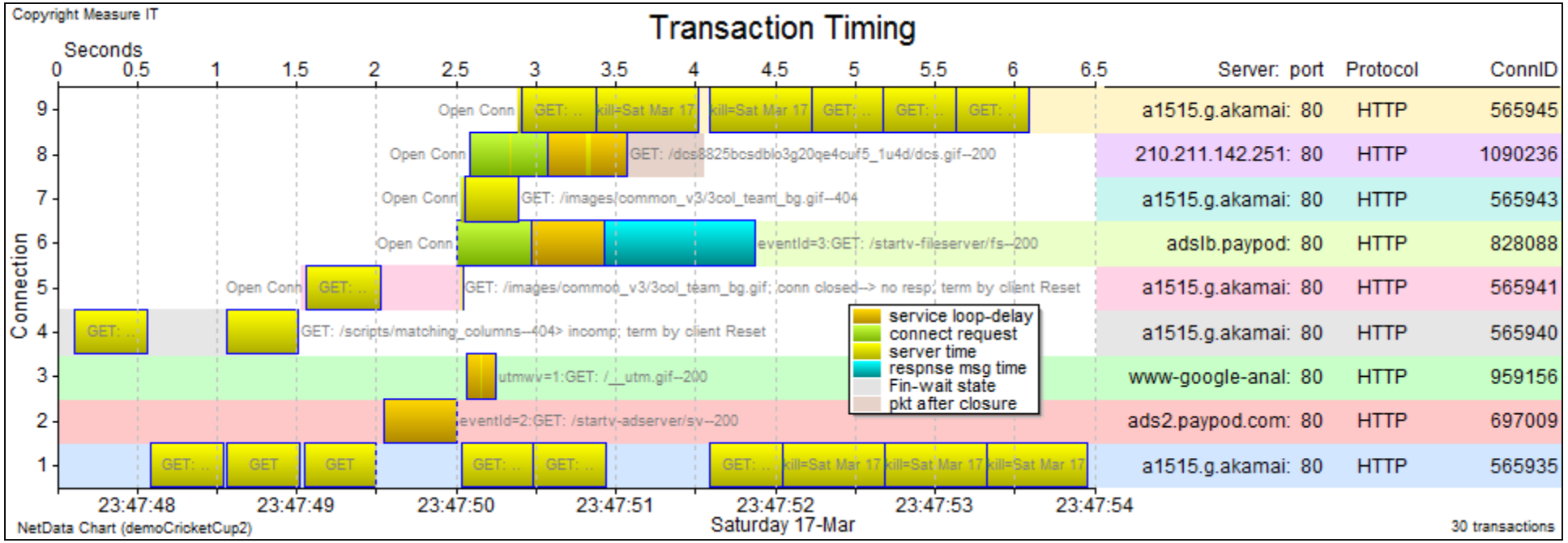
NetData's timing chart presents the **transaction activity of concurrent connections**, and because each connection is usually controlled by an application thread we gain insight into the behaviour of those threads, singly and collectively. We can usually see where a thread switches attention from one backend server to another, and we can see whether abnormal times are spent waiting on a backend server or digesting a response.

Transaction bars can be laid out in **four different forms of waterfall** to document all the round-trips involved in a user transaction or to reveal more about the structure of an application program. One form focuses on the different types of backend transactions, and for database applications this can reveal the execution of **nested program loops**.

If we overlay **markers for all the packets** of a connection we can see when and how network abnormalities introduce delays.

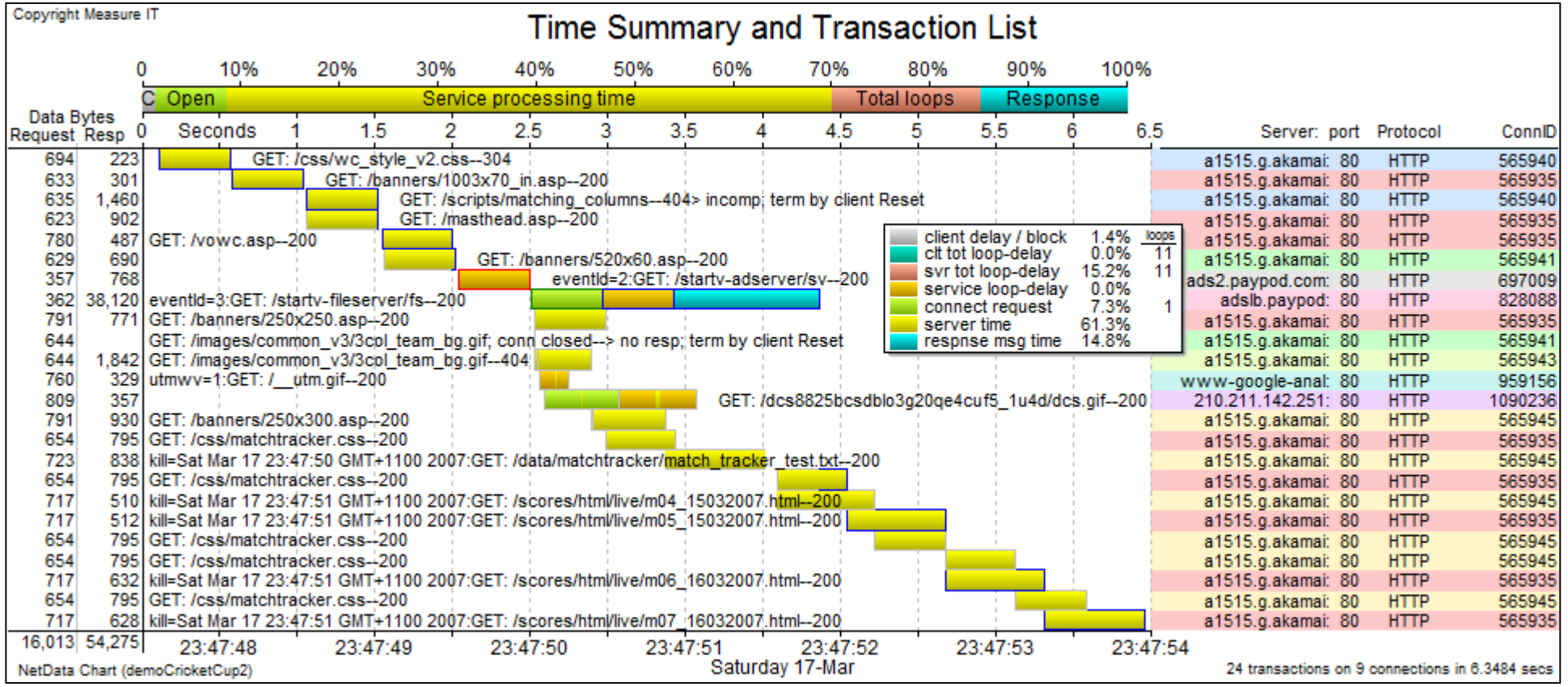
Any connection displayed on a packet-timing chart can be chosen to display its **TCP sliding windows**, the composition and handling of its micro-bursts, **window occupancy** and **round-trip times** on a slave chart.

A timing chart thus becomes the focus for an extraordinary range of performance insights, from application program structure to the effects of network errors and the operation of TCP flow control.



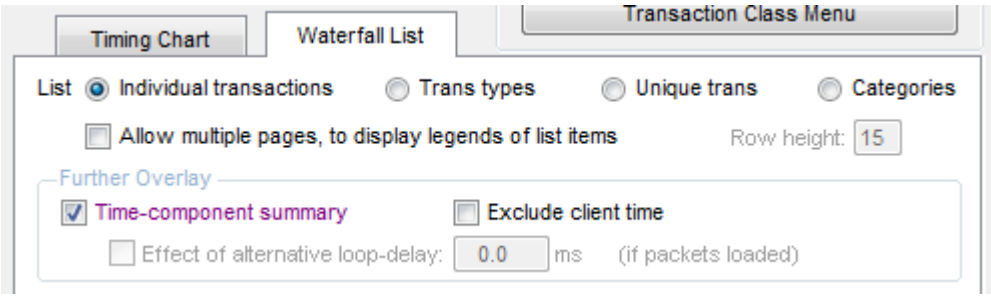
A ***transaction-timing chart*** shows the transaction activity on every connection, here loading a web page that involved 30 round-trips to 5 different servers.

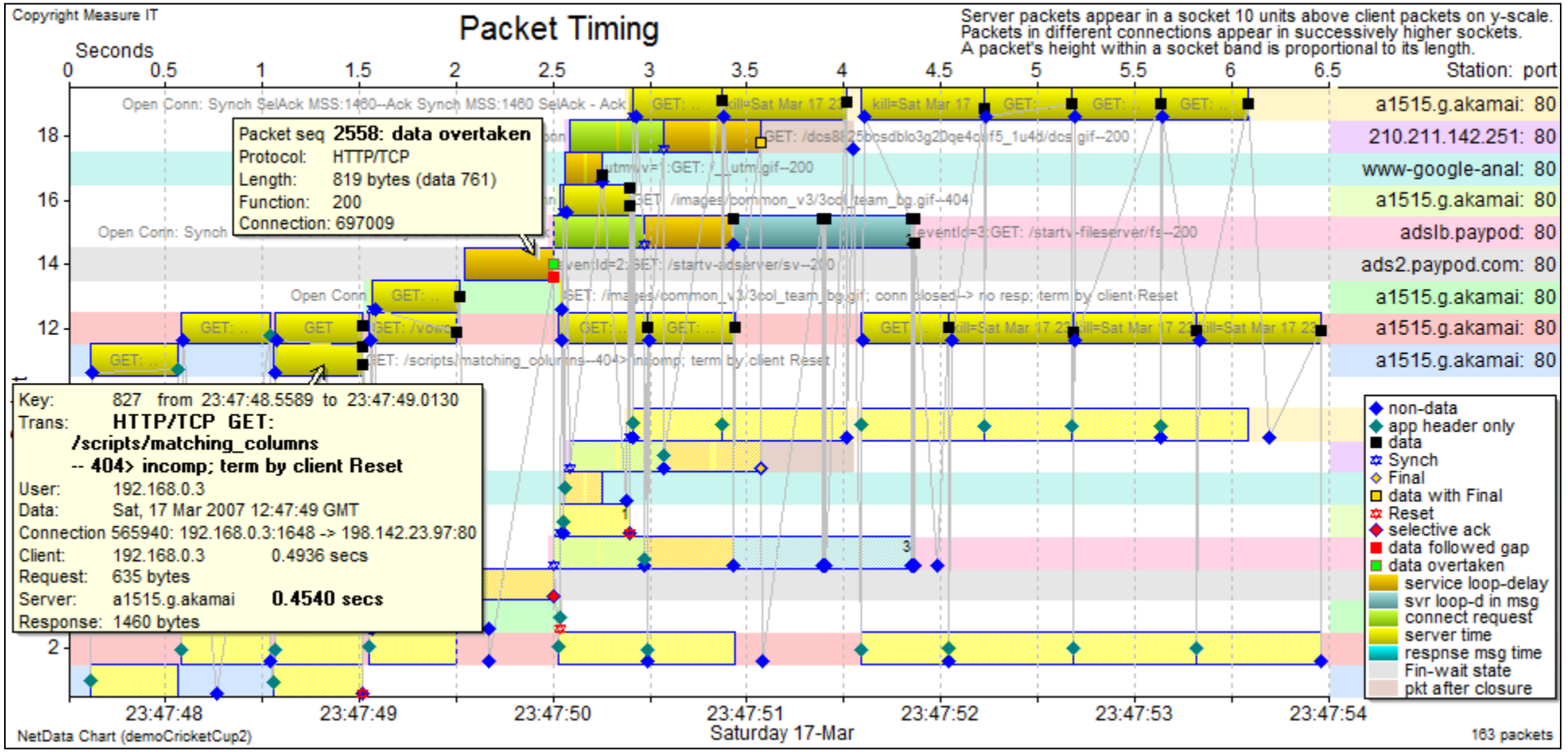
The pale-coloured bands underneath the transaction bars indicate the life-time of each connection, including Fin-Wait time.



Transaction bars can also be plotted on a ***waterfall chart***, with an optional time summary above. There are four different types of waterfall.

NetData's waterfall charts can combine the transactions of any protocol, and can be zoomed and adjusted in many ways. They can also show the **modelling of overall response time** when a data centre is moved and a path's loop-delay (propagation delay) changes.



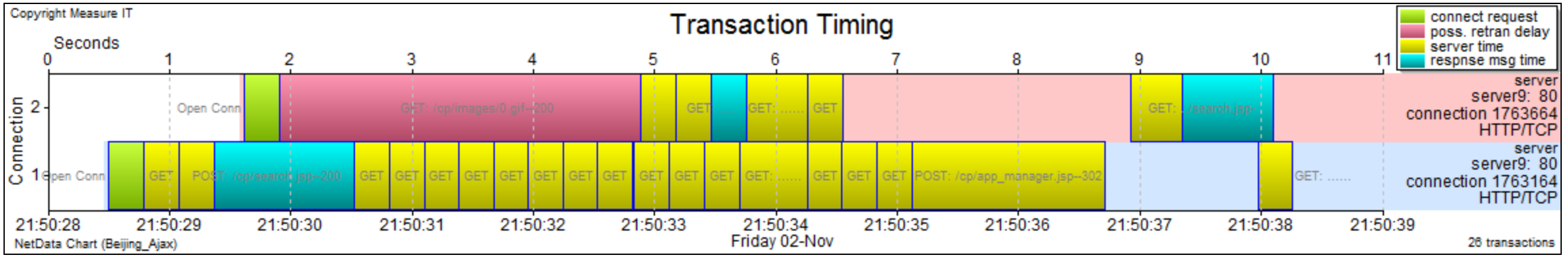


A transaction-timing chart can be overlaid with markers for every packet, using a pair of socket bands to separately carry the client and server packets of each connection. The resulting ***packet-timing chart*** shows any network errors and abnormal network delays.

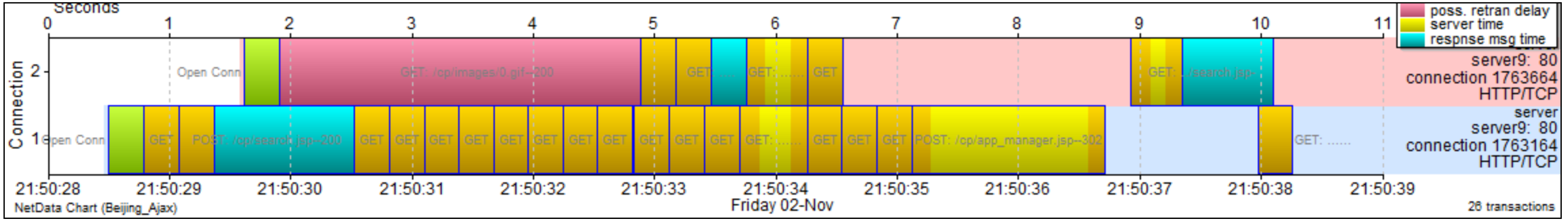
Pop-ups for individual transactions and packets provide additional detail.

Orange and grey-blue bars show how much time is spent in **signal propagation (loop-delay)** rather than server processing (yellow) or transmission (blue).

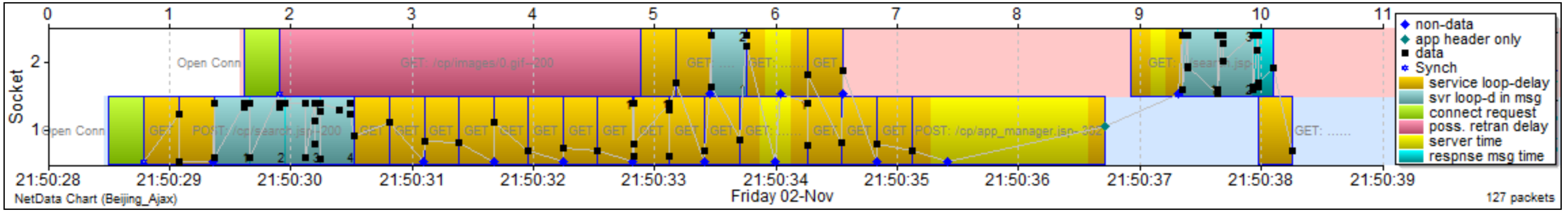
1



2



3



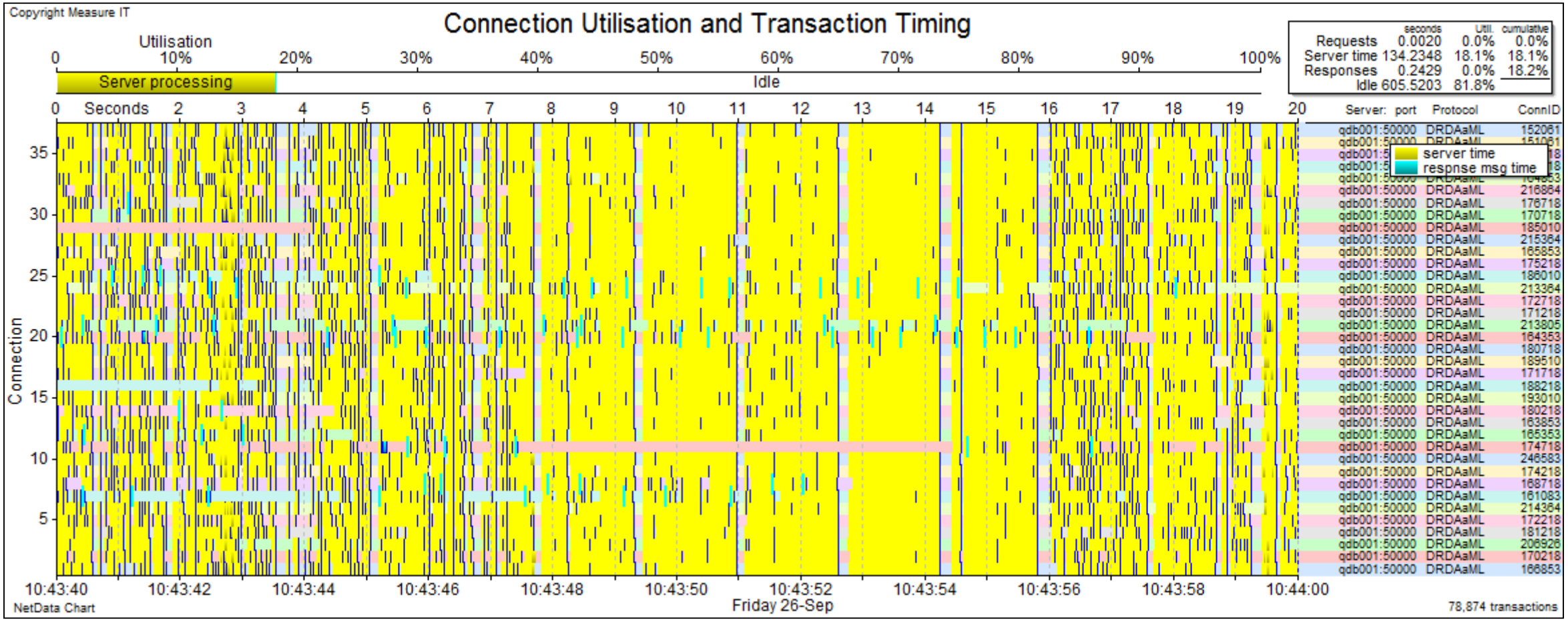
Three views of a browser requesting files for a web page using only two connections:

1. Apart from a **retransmission timeout**, most of the delay appears to be **server processing** (yellow) and **message transfer** (blue)
2. NetData has measured the minimum RTT and when it plots the time attributed to **propagation delay** we see there was little processing time, and the alignment of the few yellow bars in both connections is an indication of **garbage collection** time
3. When server-packet markers are overlaid, NetData also shows **propagation delay during message transfers**

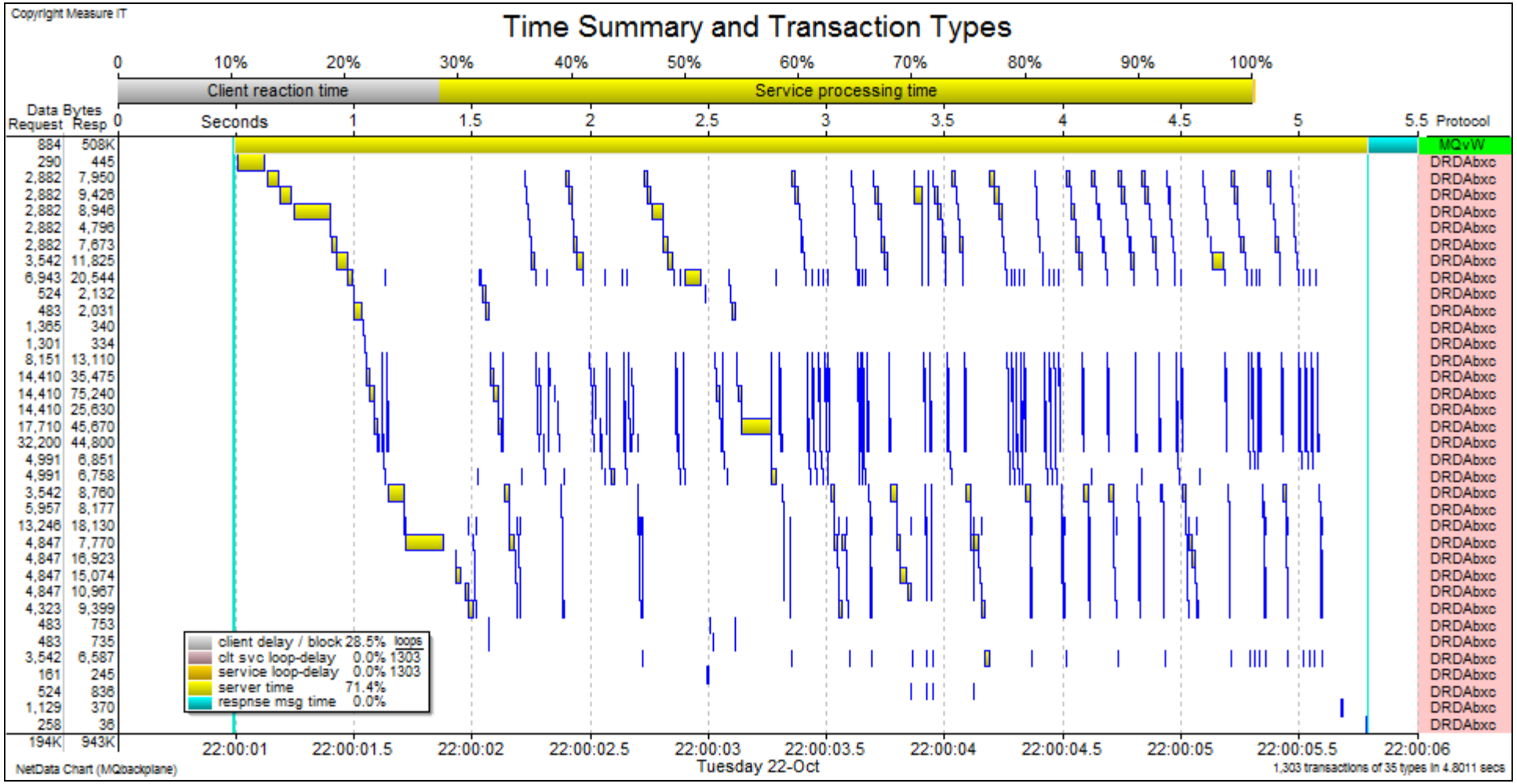
Inside Application Software

NetData here reaches inside the application servers of two systems, providing insight to software organisation, programming inefficiencies, and bottlenecks.

It is unlikely that such comprehensive and detailed views can be obtained any other way.



The database sessions of 37 **app-server threads**. In the central 8-second period all threads were continuously busy except when the **JVM stopped to collect garbage**. The short light-blue vertical lines mark Commit commands that separate the bursts of database activity generated by **different user transactions**. Were there enough threads to handle the load? Although the database appears to occupy most of a thread's time, the utilisation stats indicate that two thirds was spent in the app server, processing query *responses*. In fact the app server was starved of processing power.

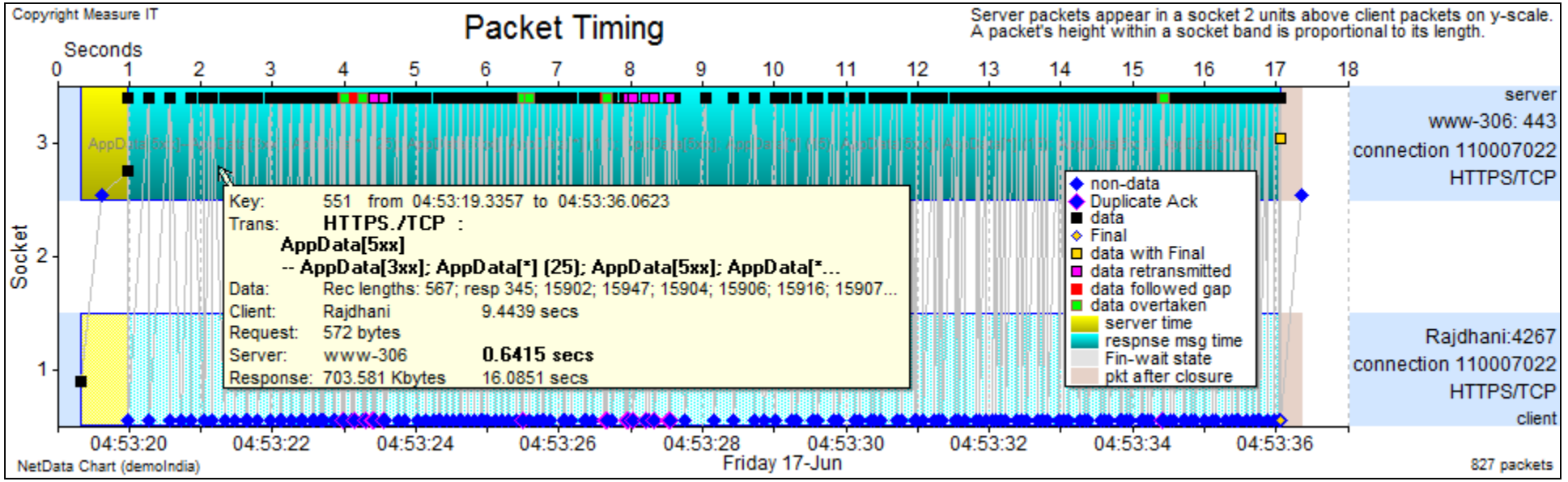


This form of waterfall chart characterises all the DB2 **database activity generated by a single user transaction** (conveyed by WebSphere MQ). All the bars plotted on an individual row represent transactions of the same type, and most are so short that they have collapsed to a single blue vertical line.

This user transaction needed 1303 queries but only 35 *types* of query, and the pattern reveals that most queries were executed in a large program loop containing an inner loop that executed up to 8 queries. This type of chart identifies those queries needing a long time in the database or client, the **redundant queries, and other inefficiencies**.

Improving Data Throughput

What is the most common cause of slow data transfers over a WAN, and how do you recognise it?

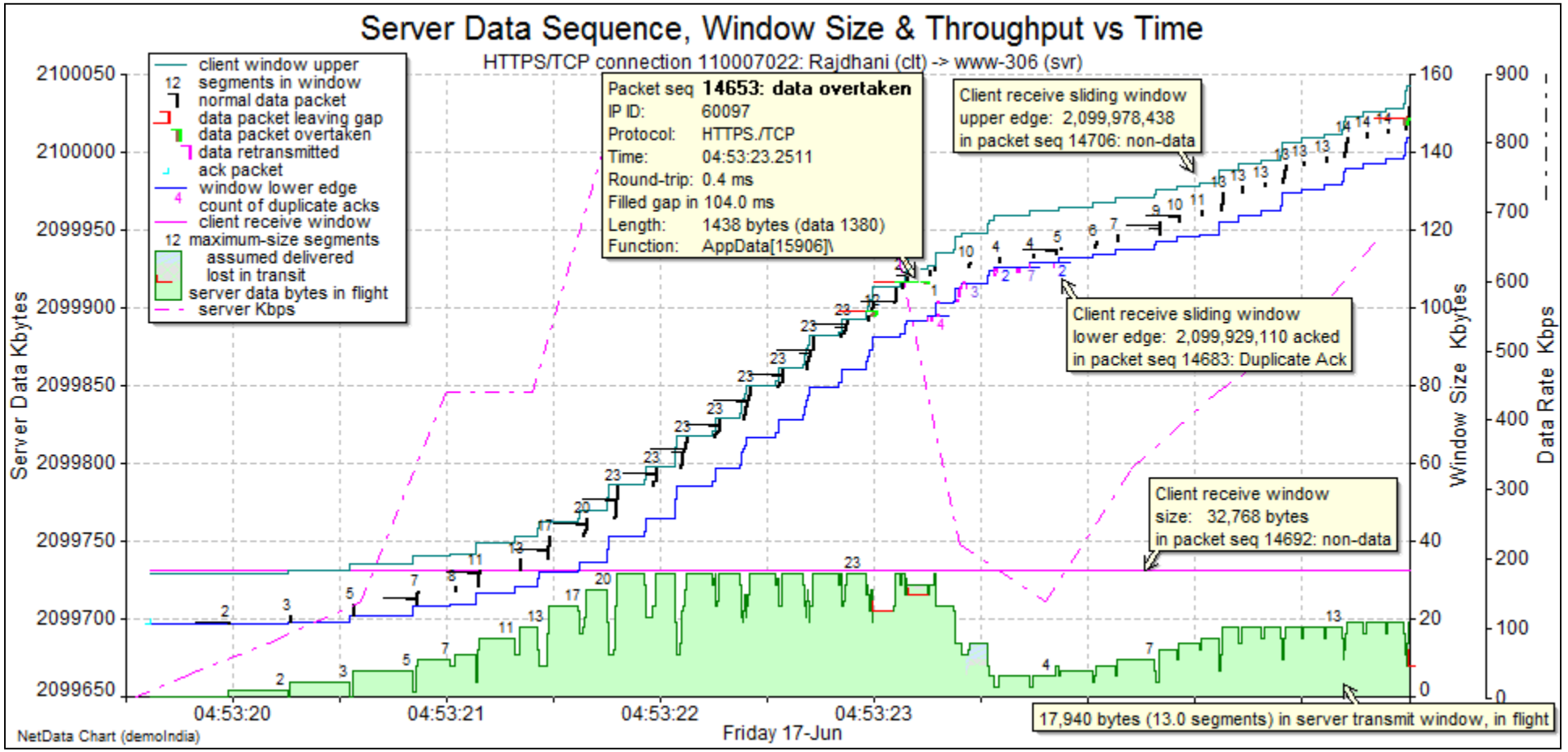


Modern networks frequently transfer very large files – some for web pages – and throughput is often much lower than the bandwidth would suggest. This packet-timing chart reveals a transfer of 703 Kbytes that took 17 seconds over a WAN; the many retransmissions, and widely varying intervals between packets, suggest there was heavy congestion and packet loss. But there are **many possible causes of delay**:

- Too small a TCP receive window for the bandwidth-delay product
- Too small a TCP transmit window
- Too little send-buffer space
- Heavy congestion (queuing delays)

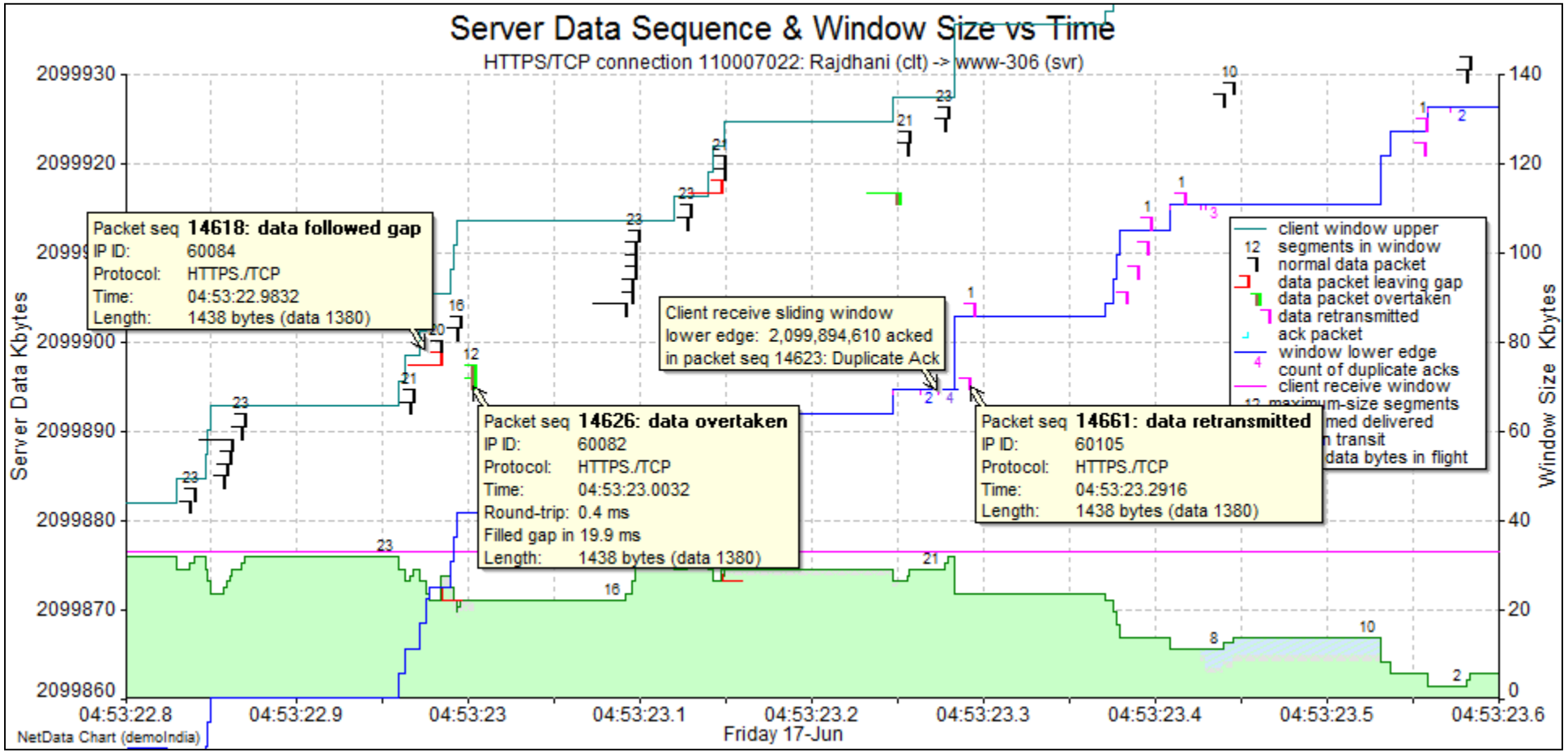
- Retransmission timeouts for packet loss caused by congestion, buffer shortage, or poor media
- Packets delivered out of order
- Bottleneck in receiver (TCP window closures)
- Bottleneck in sender (pauses in output)

NetData's charts identify all of these causes, in any combination.

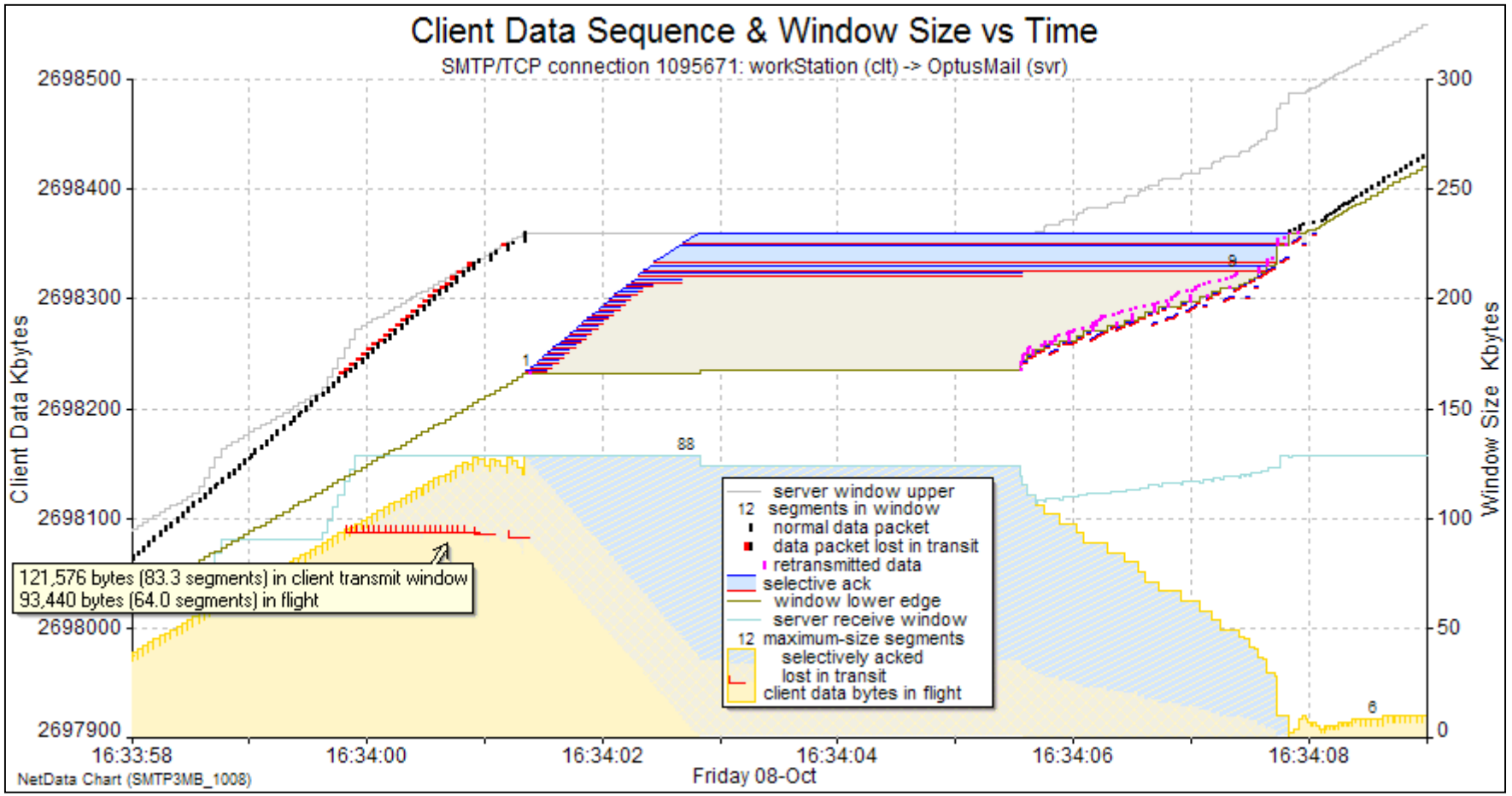


Data throughput is largely proportional to the size of the **send window (bytes in flight)** which is displayed by the green area at the bottom of this chart. Even though the traffic was captured at the receiving client, NetData presents **the sending server's view** and adjusts the graph as every data and ack packet is seen. It shows the initial slow-start opening of the send window; filling the receive window (pink line); halving the send window in reaction to assumed packet loss (i.e. congestion); slow start again; and gradual opening for congestion avoidance.

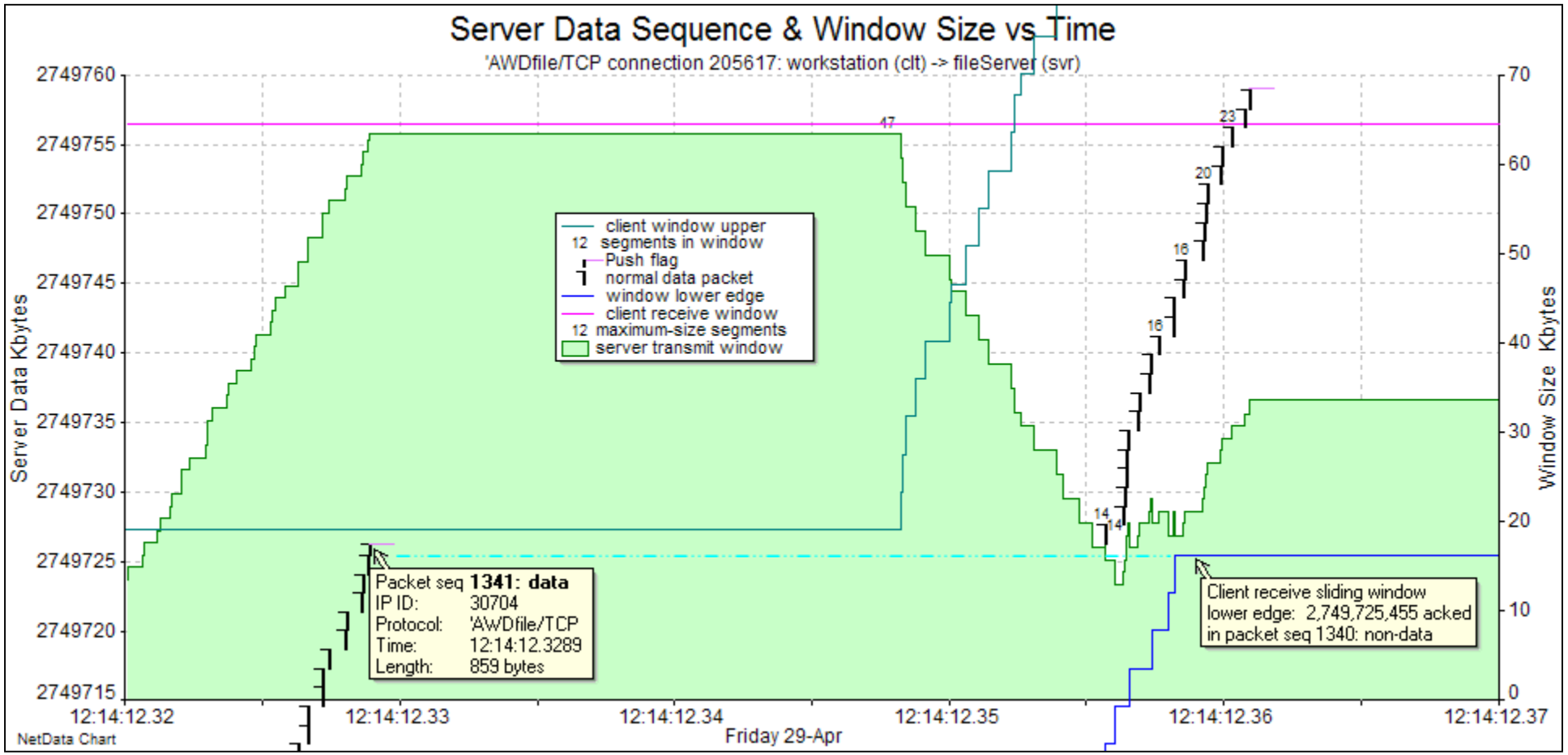
Tiny vertical strips represent **packets within the sliding window**, plotted against the data-sequence scale on the left.



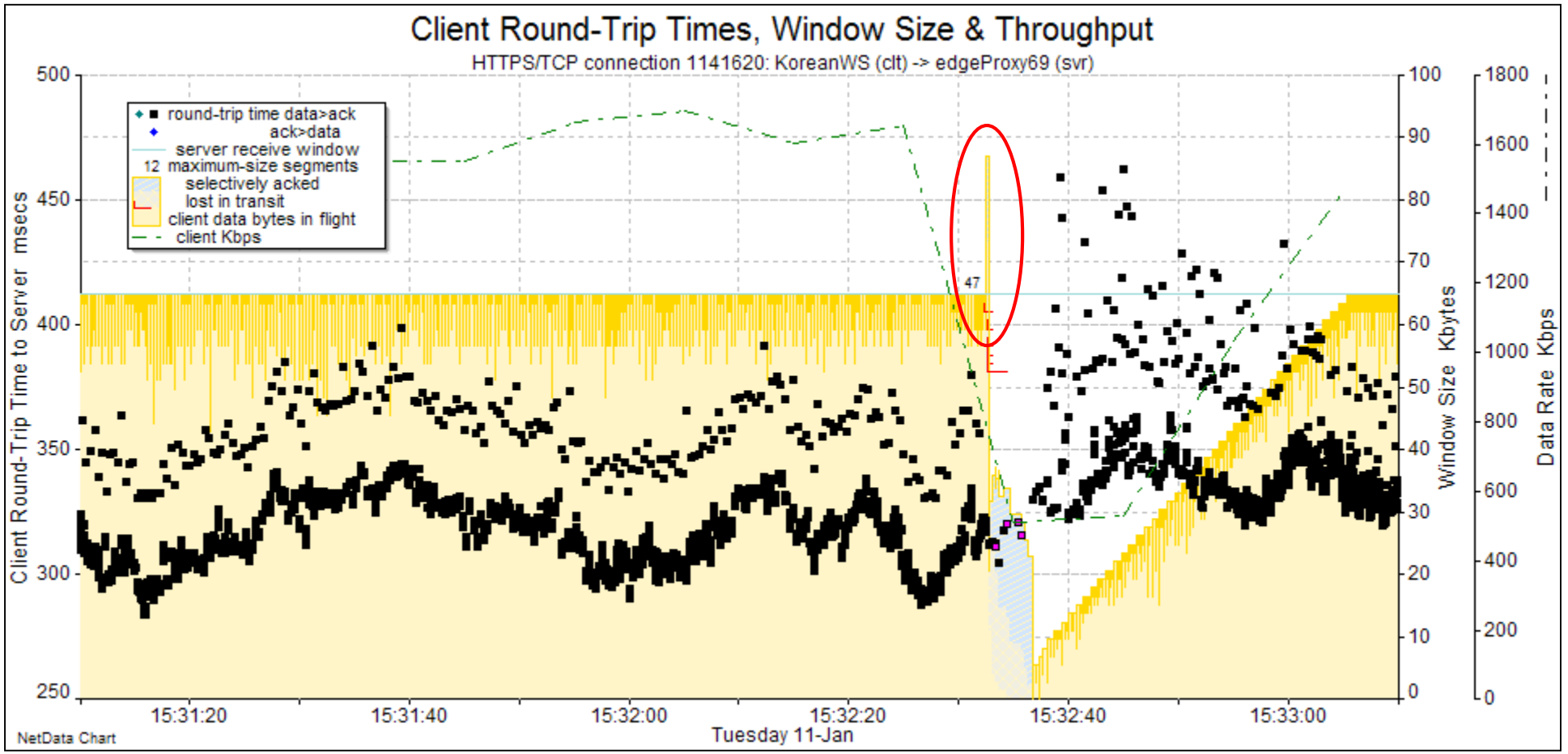
Close inspection of network abnormalities in **data-sequence charts** like this revealed the surprising result that this particular transfer was not affected by any packet loss but by packets arriving out of order (‘overtaken’) – they generated duplicate acks and in turn prompted retransmissions and frequent halving of the send window.



In this sliding window NetData has plotted the **information in all the selective acks**. The same information has also divided the contents of the cream **send window into three categories**: data selectively acknowledged; data lost in transit; and data still in flight. Although the traffic was captured at the sending client before packets were lost, NetData has been able to infer which packets were lost in transit (attaching red strips to their black vertical strips in the sliding window). The red L-shaped ticks in the cream send window indicate how many packets were in flight when each packet was lost, and here indicate that some **router could not buffer** more than 64 packets.



This chart shows the end of a burst with two send buffers (that almost fill the receive window), and the subsequent burst that has only one send buffer. The last packet in the first burst happens to make an odd number of packets, and, because the client issues an ack only after every second packet, it remains unacknowledged during the second burst. The server's TCP driver has been configured with only enough send-buffer space to fill the receive window, that is, for two send buffers of about 32 KB each, and, because it must retain a buffer until all its packets have been acknowledged, it can accept and transmit only *one* new send buffer from the application. In other words, flow is constrained primarily by a **shortage of send-buffer space**.



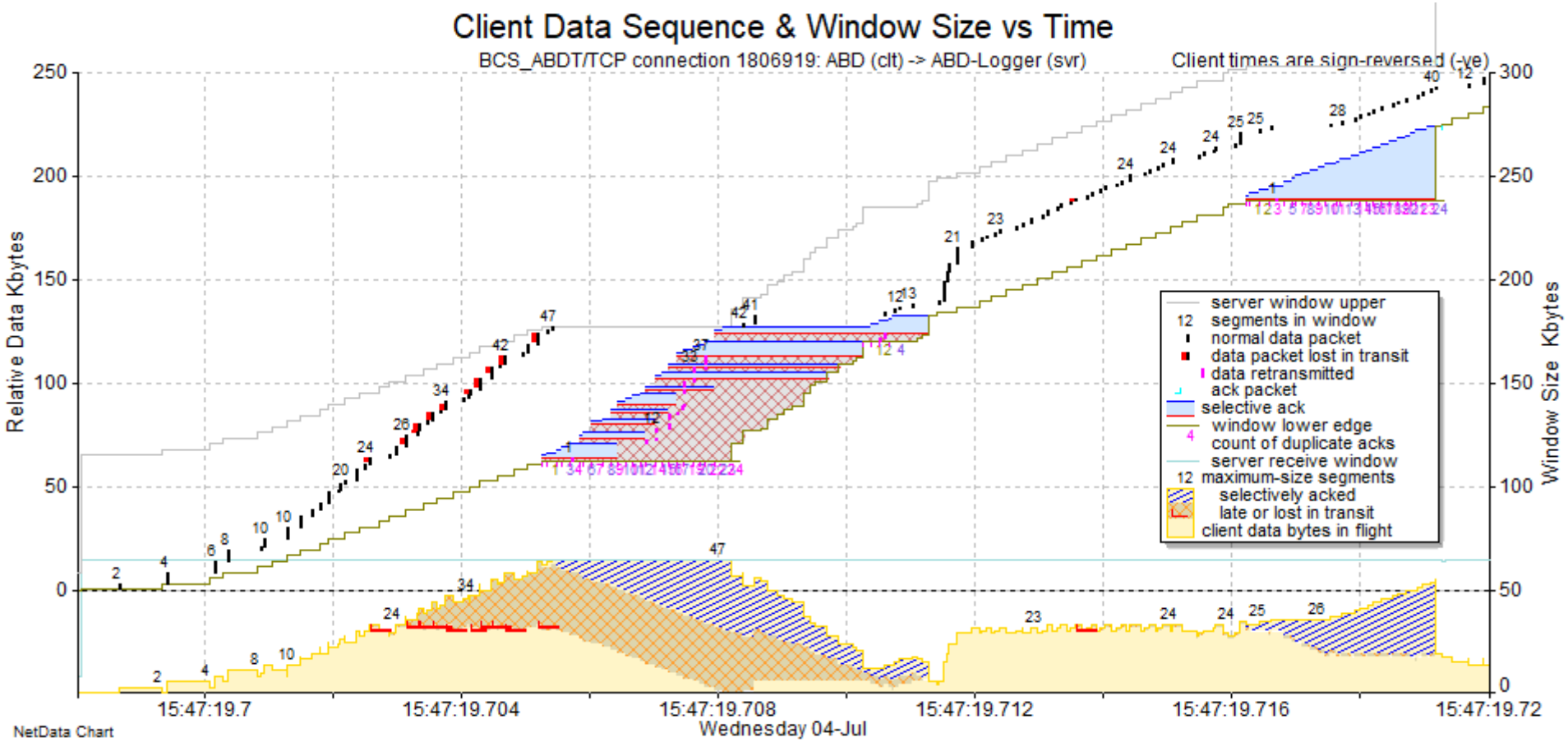
The speed of this file transfer was limited by a receive window that was too small for the loop-delay of 285 ms. The cyclic variation in round-trip times, mostly between 290 and 340 ms, reflects queuing delays within some router (i.e. **congestion**).

Throughput dropped to a third when first one packet was lost and the TCP driver subsequently **broke the fast-retransmit protocol** by transmitting data outside the window.

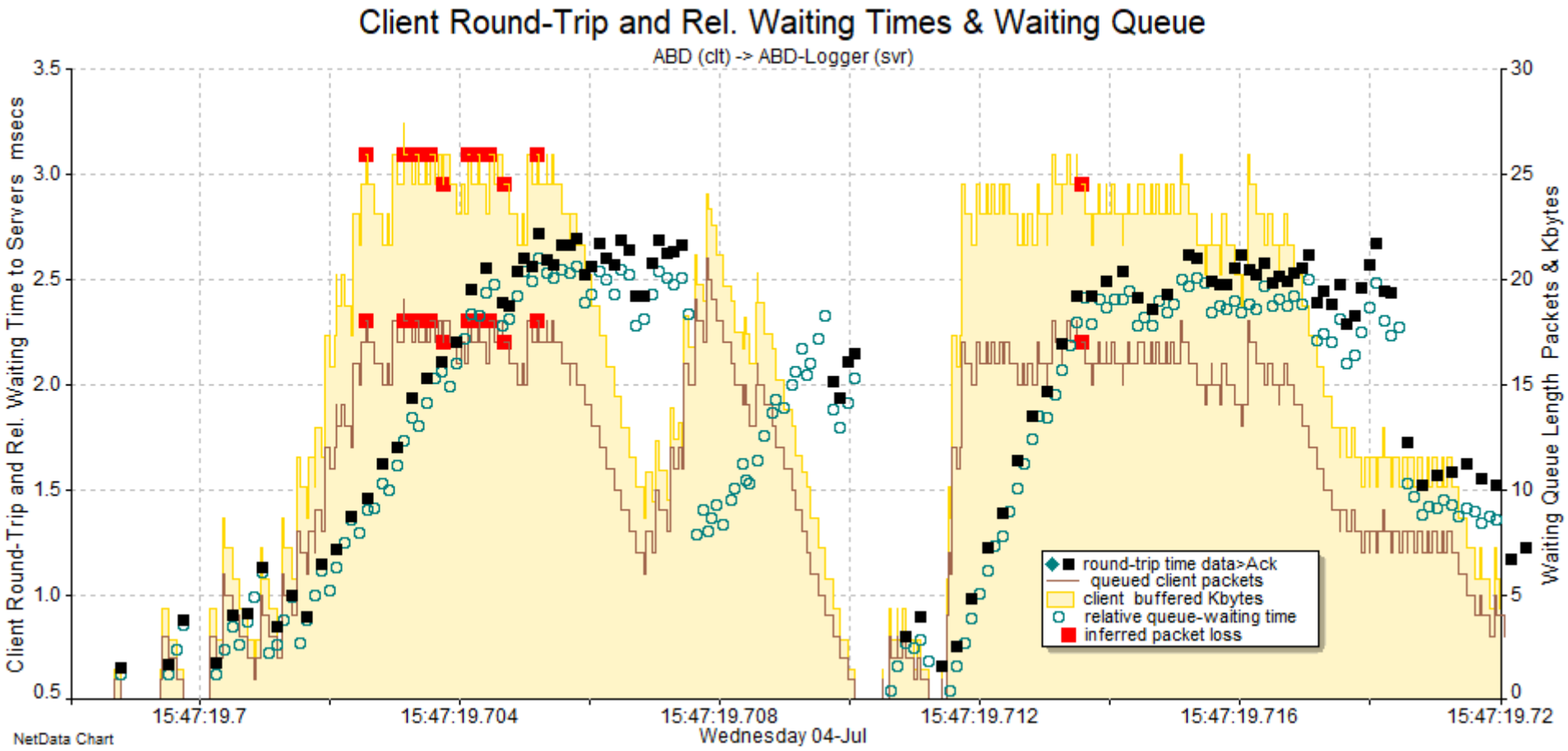
Causes of Packet Loss
Modelling Queues and Packet Shaping

Packet loss usually has a severe effect on data flow because congestion-avoidance schemes mandate an immediate reduction in the flow rate and cautious attempts to increase the subsequent rate. In modern networks most packet losses are caused by some form of buffer shortage – perhaps by an inability to marshal buffers in time, but more likely by a queue overflow or a traffic regulator known as a *packet shaper* or *packet policer*, with contracted *Committed* and *Peak* information rates.

NetData has extensive facilities to measure flow rates and is also able to determine the parameters that control packet queues, packet shapers and packet policers. It does this by modelling the behaviour of packet queues, token buckets and leaky buckets, and overlaying modelled performance on charts of measured performance.

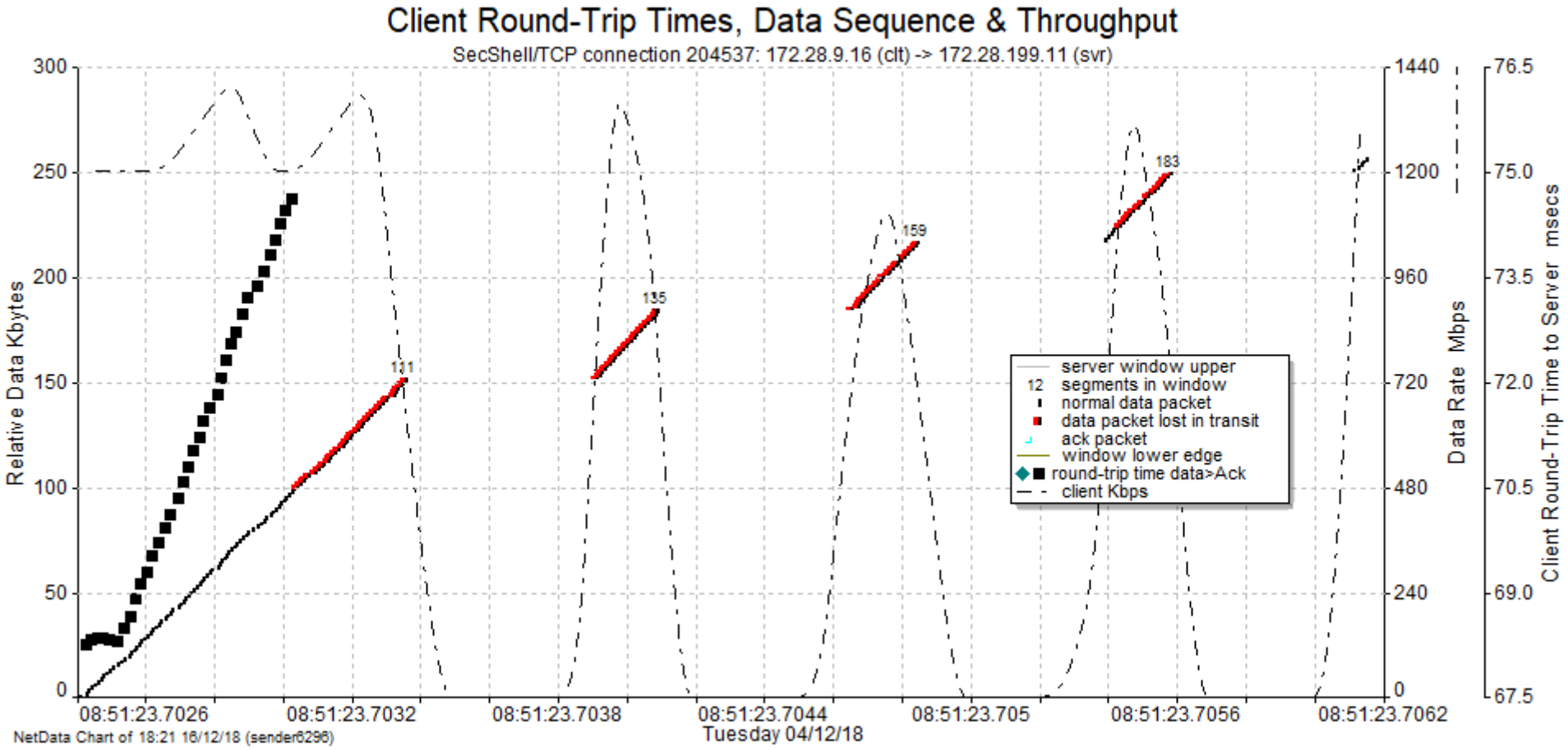


A file transfer was disrupted by frequent packet losses, indicated here by the selective-ack information and the red packet strips plotted on the sliding window. The TCP fast-recovery scheme prompted retransmissions with minimal delay but the bytes-in-flight area graph below the sliding window shows that the congestion-avoidance window was halved many times, severely reducing the data flow. Why were packets lost?

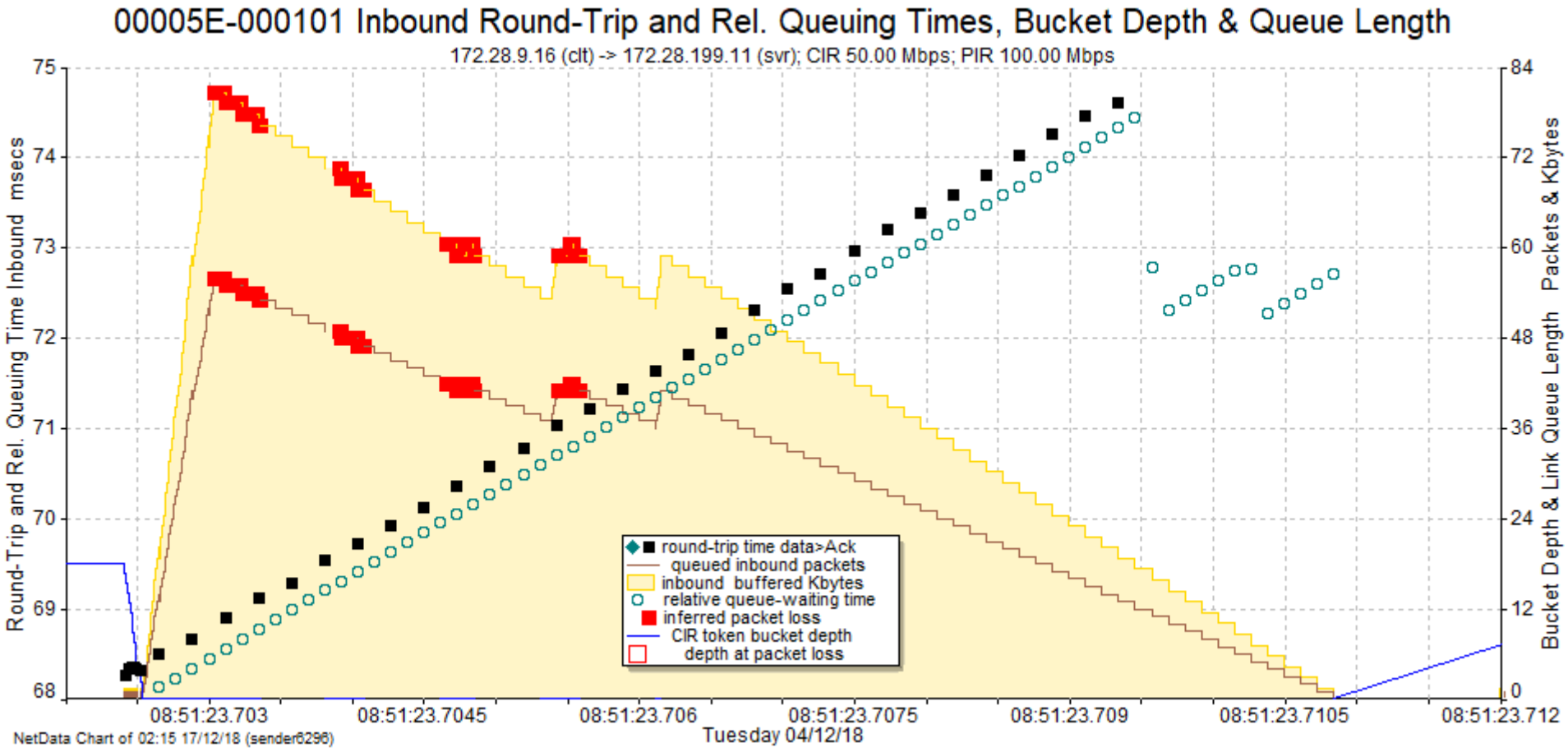


On an assumption that packets were overflowing a queue waiting for transmission over a 100 Mbps link, NetData modelled the behaviour of that queue with packets arriving and being dropped as indicated by the captured traffic. The cream area graph of queue length shows that packets were dropped (indicated by red squares) only when the queue size reached 25 Kbytes.

The model's validity is confirmed by the markers for observed round-trip times which increase by amounts which closely follow the modelled queue-waiting times (indicated by green circles). A subsequent examination of the network path revealed an old router with only 25 KB of buffer space.



This file transfer across another network was also subjected to severe packet loss (tiny red packet strips). The chart also shows that packets passed the sniffer at 1 Gbps and the first dozen packets participated in round-trips that were close to the minimum RTT of 68 ms. Subsequent packets, however, experienced increasing delays, indicating that they were held in a queue of increasing length. Switching the flow in and out of a queue is undeniable evidence of a packet shaper's operation.



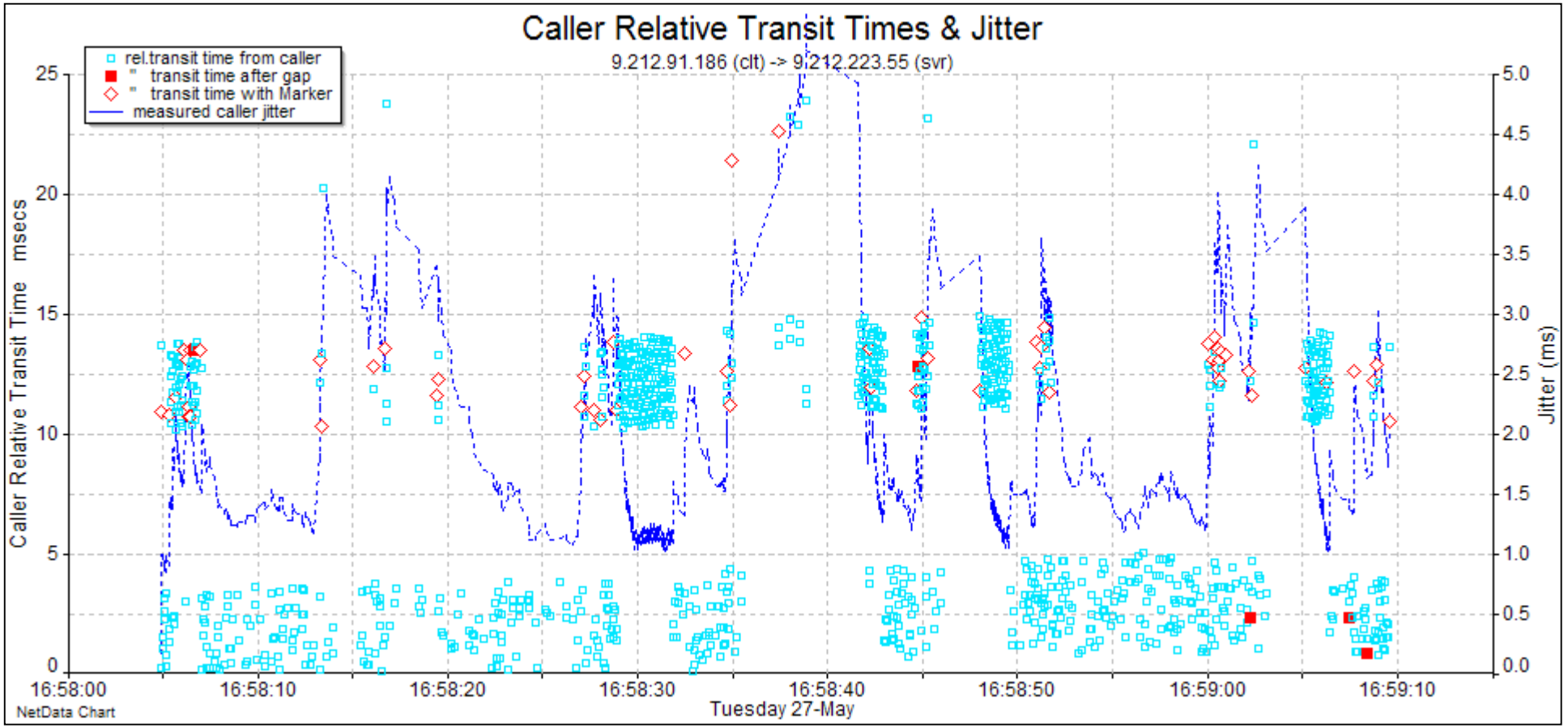
NetData modelled the network's packet handling very closely, assuming a packet shaper that had a token bucket with up to 18 KB of tokens and a Committed Information Rate (CIR) of 50 Mbps. When the token bucket (blue line) became empty, packets were queued in a leaky bucket with a Peak Information Rate (PIR) – the packet 'leaking' rate – of 100 Mbps. The PIR's validity is confirmed by the markers of round-trip and queue-waiting times that track each other very closely. Packets were dropped when the leaky bucket first filled to 80 KB, and subsequently when the bucket depth exceeded only 60 KB.

Knowledge of the packet-loss mechanism suggested ways of minimising the loss and increasing the flow. The buffer space allocated to the packet shaper could be increased, or the size of the packet bursts could be reduced by reducing the receive window or the allocated send-buffer space, below 95 KB.

VoIP Jitter

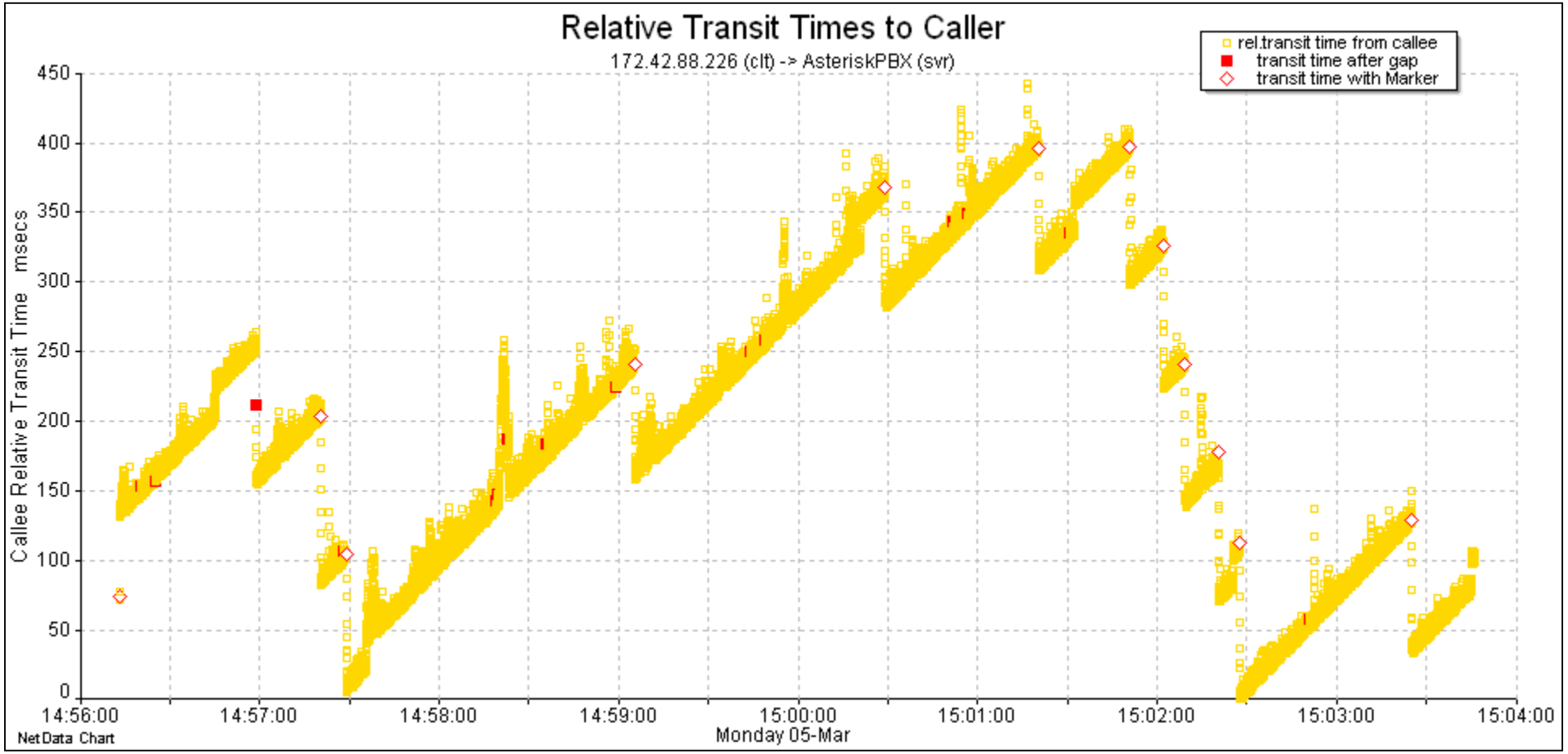
Jitter has a standardised measure for VoIP network quality but because it involves sampling and statistical smoothing it is virtually useless when looking for causes of poor voice quality.

On the other hand, patterns of **individual transit times** – the raw data from which jitter is calculated – provide valuable clues to the causes of jitter. Few if any other tools present this information that some say is impossible.



Light-blue squares mark the **relative transit times** of RTP audio packets from a VoIP caller. Red diamonds indicate packets with a Marker flag, and red squares indicate packets that followed a sequence gap.

When the packet rate was high, packets were given a different path that took an extra 10 ms, indicated by the middle band of light-blue markers. Irrespective of the path, packets were also subject to a random delay of up to 4 ms (the height of each band) – a second source of jitter involving a **polling delay** that could be produced by the cyclic attention to packet queues in some firewalls. A third cause of jitter is attributed to occasional extra delays producing relative transit times of nearly 25 ms.



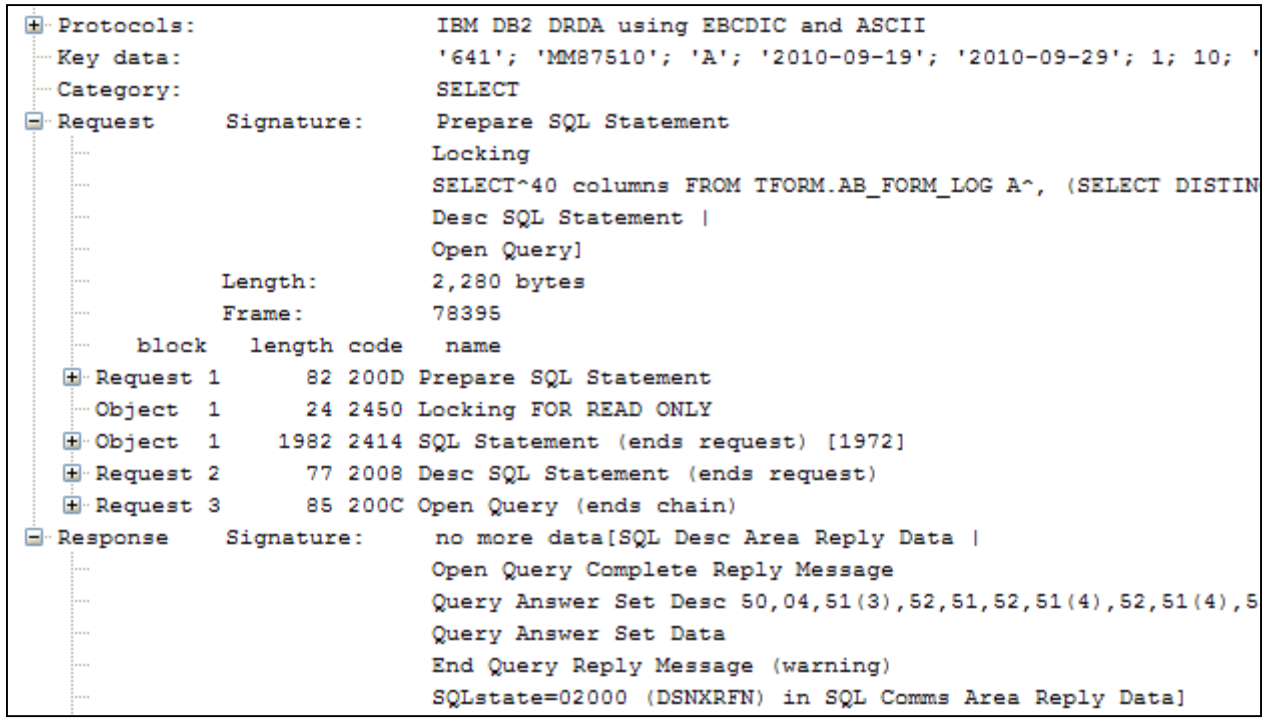
The rising slope in these bands of relative transit-time markers is due to a decoding clock in the callee's VoIP phone that was faster than the output clock of a PBX. This difference between input and output speeds caused transit times to gradually increase and produced frequent over-runs in the PBX de-jitter buffer. The PBX reacted to each over-run by suddenly reducing transit times by 100 ms.

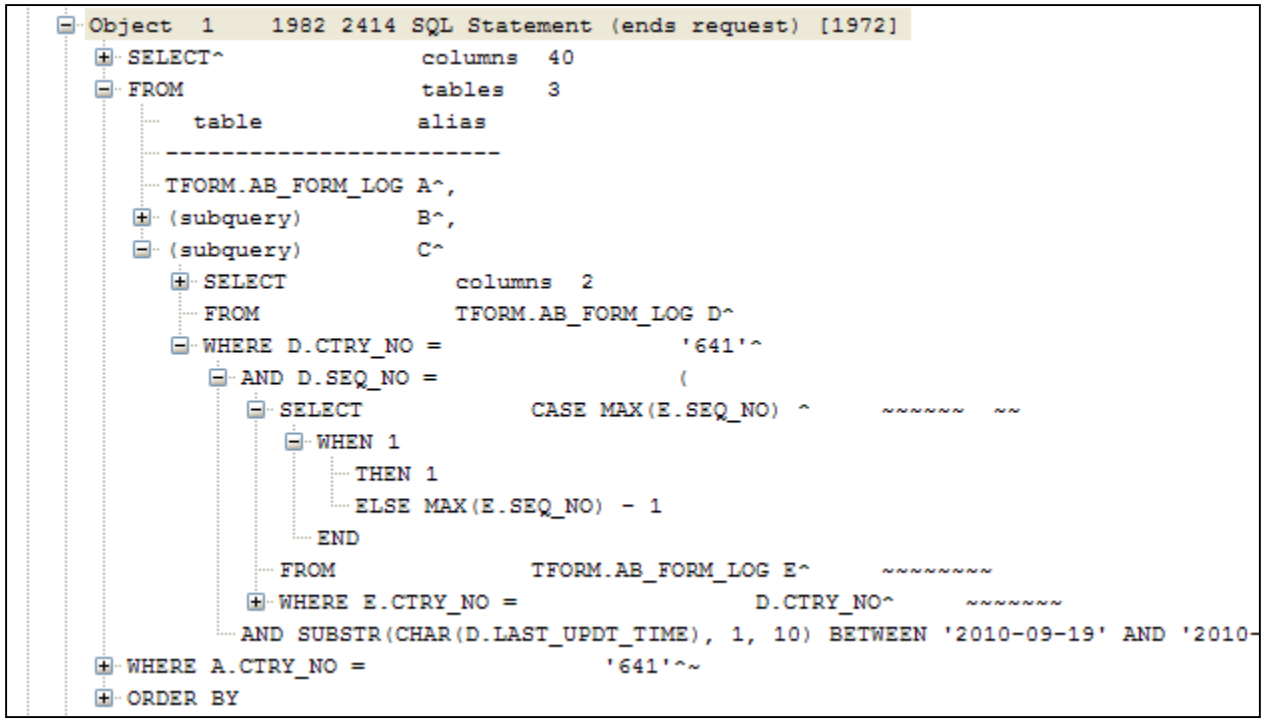
Understanding the Application

NetData exposes slow transactions, failed transactions, network abnormalities and many signs of stress. But are they relevant to the main problem?

We often have to work with minimal system documentation. One of NetData's important functions is to present the most detailed and clear descriptions of transaction messages and help build a **high-resolution picture of how an application works**. Only then can we decide what behaviour is relevant.

NetData has **decoders for a very large range of application protocols** (and their underlying network protocols). The decoders reconstruct transactions even when transactions are **multiplexed over single connections** (as for SMB) or involve several connections (as for IBM MQ), and are able to parse much if not all of the transaction messages to **extract critical elements such as error codes**.

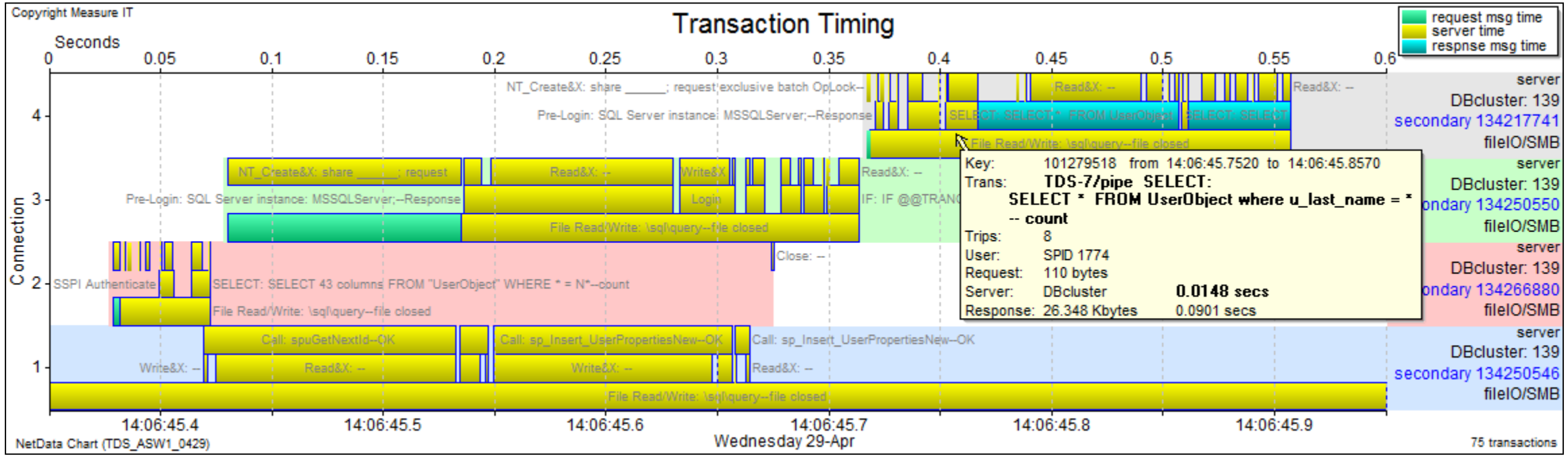




NetData thoroughly decodes the messages of **all the major database protocols** and can describe their contents in a tree structure, as in this excerpt of a SQL statement for a DB2 query.

SQL, XML and MIME statements can be displayed in exactly the same format in which they are transmitted, or in a form that reveals their structure and allows any parts to be explored in detail.

The above DB2 query, for example, accessed one table and two sub-queries. The second sub-query had a third sub-query in its Where clause.



NetData's timing charts not only show transaction activity within connections, but they also unscramble different forms of multiplexing and display transactions in different protocol layers.

This chart reveals **multiplexed database transactions** on just *one* TCP connection. The SQL Server protocol, TDS-7, here runs in named pipes implemented with SMB transactions. NetData has split the activity into four bands that it regards as ***secondary connections*** but in this context are different **named pipes**, uniquely identified by their SMB tree- and file-IDs. The stack of three layers of transaction bars in each band relate to three protocol layers: the SMB transactions for opening and closing files (or pipes), and for writing and reading data; the TDS database transactions; and bars for pseudo transactions that indicate the **life-times of the pipes** (file reading and writing).

The grey band reveals two identical queries each requiring 8 round-trips because their response messages of 27 KB required seven SMB Read&X commands that each read no more than 4 KB. Write&X commands conveyed the queries.

To investigate delays, this chart could be overlaid with markers for all the packets that carried these transactions.

Monitoring and Analysis

- Most tools are little more than fuel gauges:
  - They collect and integrate event counters with measures of bandwidth usage, memory usage, CPU utilisation and the like.
- Very few tools analyse sequences of events and present results graphically to characterise system behaviour:
  - Most servers and other devices record significant events in log files and can yield crucial clues, but
  - log files can't match traffic recordings for either their **broad view** or **forensic detail**.
- There is a huge wealth of diagnostic information in capture files:
  - Traffic recorders are as valuable to IT teams as black-box flight-data recorders are to aviation.
  - Effective traffic recorders (sniffers) can be assembled with free software (Wireshark) and any laptop, PC or server; but
  - very few analysis tools can extract the important diagnostic information.

NetData Objectives

- Extract all the diagnostic information in capture files.
- Record all analysis results in a database for quick and selective retrieval.
- As far as possible present information graphically, and
  - expose all forms of unhealthy behaviour and stress
  - allow multiple chart overlays to test correlations
  - pop-up descriptions when cursor rests on any chart object
  - give quick access to supporting tables
  - permit verification of analysis by drilling down to relevant packets
  - provide alternative views of behaviour to corroborate findings
  - present publication-quality charts and tables

All the charts in this overview are either entirely unique to NetData, or
provide a much deeper analysis than other tools; yet
they barely scratch the surface of NetData's capabilities

New Diagnostic Methods

"Why can't you look at the packets with Wireshark?"

With NetData the analyst can contemplate a much larger list of failure mechanisms that are virtually impossible to check with any other tool, such as:

- Ceilings for threads, connections, router buffering
- Port-recycle times
- Polling delays
- Limited send-buffer space
- Inappropriate, obscure TCP parameters
- Queues for different transaction types
- Transaction redundancies
- Database abuse with unnecessary round-trips
- Database blockages; missing table indexes
- Single-threading blockages

- Short-term bottlenecks at different points in system
- Obscure bugs in protocol drivers
- WAN accelerator faults
- Application coding inefficiencies
- Absence of buffering for application output
- Changes in transaction mix
- Mismatched timeouts across system
- Connection-pool misconfiguration
- Insufficient packet-queue buffer space
- Inappropriate packet-shaper and –policer parameters

In fact an experienced NetData user adopts a **quite new approach to problems.**

You have to use NetData to appreciate its power and ease of use.

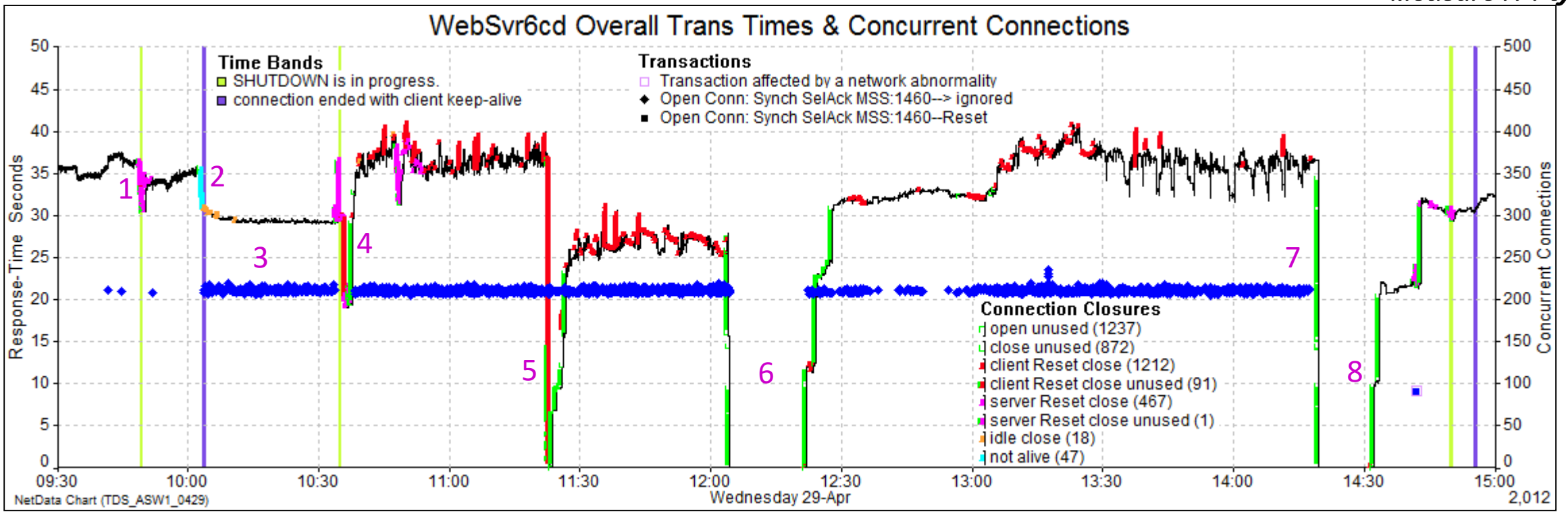
Sequence of Events
Which is the Root Cause?

Finding the root cause is often as sacred as a search for the holy grail because only when the real cause is found can we be confident about the fix for a problem. It is very easy to jump to the wrong conclusion, wasting considerable resources such as added bandwidth and more computing power that have no effect on a problem.

But modern IT systems have such considerable redundancy and sophisticated recovery routines that the more serious problems generally arise from a combination of faults and weaknesses, and often depend on a sequence of events that may take hours to unfold. In such a case, which is the root cause?

As the following example shows, the only answer is to understand as much as possible – all the forms of unhealthy behaviour, inefficiencies, weaknesses and faults. Only then can we choose the best options for avoiding or fixing the problem.

The following chart summarises a 5-hour outage. It identifies and characterises the significant events through which the problem unfolded and, because it documents the times of the events, it correlates with and confirms operator records.



A complete system outage as documented by the traffic between one web server and its database server, with eight events:

1. The database closed about 50 connections with Resets (pink markers on concurrent-connections graph) and responded to other commands with “SHUTDOWN in progress” (pale-green vertical time bands).

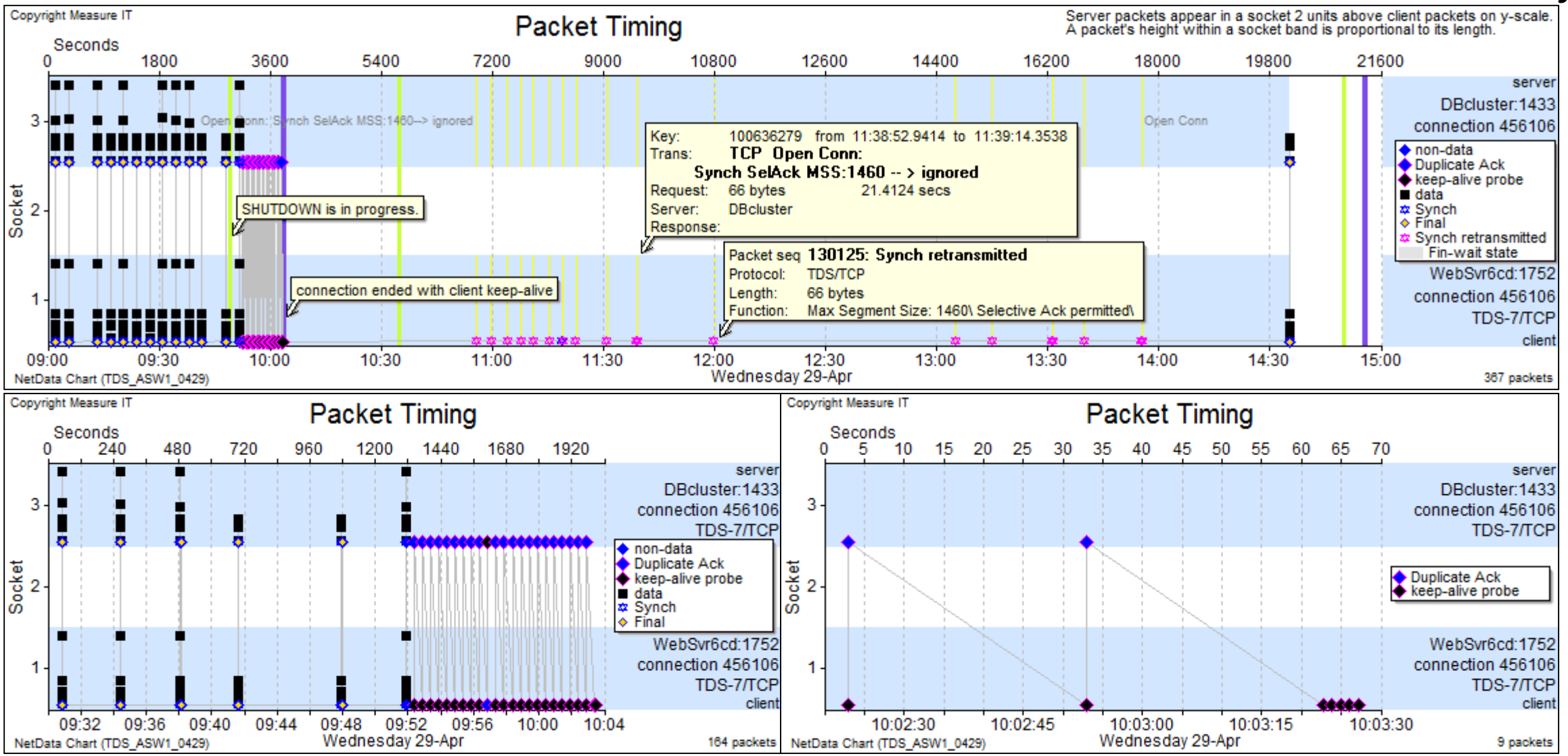
2. While the database server restarted it ignored connection keep-alive probes in 47 connections (light blue markers) and those connections ended as ‘not alive’ (purple time bands) without Final or Reset flags, effectively making these client ports ‘toxic’.

3. Subsequent attempts to open new connections from these toxic ports failed in spite of Syn packet retransmissions over 21 seconds (blue diamond markers) because a firewall believed that the old connections were still open.

4. Part of web server was shut down and restarted, closing a pool of 92 unused connections with Resets (red markers) and re-opening.

5, **6** and **8**. Whole web server shut down and restarted.

7. Firewall restarted to remove toxic state of all connections.



Five hours in the life of one web-server port and its connection with the database. For 50 minutes it was opened frequently to conduct a single database operation and then closed. At 09:52 it was left open, activated by client keep-alive probes. At 10:03:25 the keep-alive probes were ignored and the web server considered it closed. Until 14:35, soon after the firewall had been restarted, all attempts to open a connection were ignored because the firewall believed the connection to be still open.

In diagnosing this outage the preceding charts were supported by many others that identified several system weaknesses:

1. A commerce software package in the web server that opened pools of 92 connections that were never used.
2. Database sessions opened for only a single database operation – query or update – to handle a user login.
3. Database sessions left idle while their application threads waited for a new login, and then closed as a consequence of a new program object being created and its database connection being opened.
4. Connection opening being handled by a single thread, with the result that while a new connection request was ignored, retried and timed out over 21 seconds, all web-server login activity became frozen, and the whole system became virtually unusable.
5. During a system outage dozens of ephemeral ports in each web server were toxic and froze their server for 21 seconds each time the server tried to re-open their database connections.
6. Connection requests from toxic ports were dropped by a firewall.

The immediate evidence of the problem found in log files was that the web servers had trouble connecting with the database, and the first finding from traffic captures was that the network was dropping Syn packets, but the chain of events began with a database restart. There would have been no problem if connection pools were used as intended; application threads re-used their connections or closed them immediately after they were used; the web servers didn't issue keep-alive probes; connection opening wasn't single threaded; the firewall had an idle-connection timeout much shorter than its default of one hour; or the firewall was reset when the database was restarted. Which was the root cause? Removing any one of these weaknesses would fix the problem.

What other tools might have helped with this diagnosis? Wireshark and NetMon could show only that Syn packets were being lost in the network – where in the network but not why. Charts of top talkers, showing when traffic stopped and started, would not help. Some tools can measure web response times but they would only confirm there was a problem, as would better tools that present a pie chart showing where a transaction's time was spent. No bounce chart, wizard, automated analyst, 'doctor' or canned report could do more than confirm the packet loss. Tools for Application Performance Management (APM) which analyse trends and help manage capacity issues are not relevant. No performance instrumentation in application code – nor any combination of monitoring agents in servers – could illuminate the important aspects of this problem.

Even when there is no serious fault in the network the answers lie in the network traffic. Those with access to the traffic must be able to explore all types of theories and all aspects of system behaviour, with tools that are flexible, thorough and easy to use.

Who was responsible for the 5-hour outage of this online banking system? The software vendor that delivered ineffective connection pooling? The application developer unaware that connections were not being used as intended? The system builder that specified client keep-alive probes? The system designer that hadn't specified appropriate timeouts throughout the system? The support team that didn't understand what was required to recover a system after an outage? The system architect?

Can this responsibility be left to a data-centre or cloud operator?

If you are responsible for only application development or infrastructure support, let alone a whole IT system, you must have an effective diagnostic capability either in house or on call. You must be able to see how your system works – and you must look.

Unique NetData Benefits

1. See **how systems really work**, not just what the designer intended
2. Give invaluable **feedback to application developers**
3. Characterise transactions for **capacity planning**
4. **Optimise configuration parameters** before conducting expensive load tests
5. **Check whether tests are invalidated** by the intrusion of uncontrolled variables
6. **Find bottlenecks** during load tests and in production
7. Proactively **identify problems** before they become serious
8. **Diagnose the most complex problems** quickly, avoiding prolonged and expensive critical situations
9. Gather data **non-intrusively**

NetData Assignments

- NetData has played a key role in cracking innumerable IBM *crit sits* for customer accounts and IBM corporate systems around the world over the last 15 years – all without visiting an equipment site – including
  - a data-centre problem of an IBM service provider in Zurich; and
  - a connectivity problem between web and app servers in an IBM data centre in Atlanta
- Measure IT has diagnosed critical performance problems in a very wide variety of systems, including those of
  - Australian Commonwealth government departments
  - Banks in Australia and other countries
  - Software vendors
  - Online shopping web sites
  - Electricity generation firms
- NetData has been licensed by
  - Westpac
  - NAB
  - CSIRO (government research)
  - Department of Health
  - IBM on several customer accounts and for the corporate network in the Asia Pacific region.

"Without NetData, you're just floundering in the dark"

It is impossible to appreciate NetData's productivity tools, and the ease with which charts can be generated and modified, without watching an experienced user, or using NetData yourself.

Ask for a demonstration analysis of a current problem, or a trial licence subscription

Contact:

Bob Brownell BSc BE PhD

Director, Measure IT

ph. +61 2 9144 3151

Bob@netdata-pro.com