What Do TCP Selective Acknowledgements (SACKs) Look Like?

Part 1: Investigating Simple Selective Acknowledgements

#1 in the Series: Understanding Wireshark Outputs with NetData Charts.

Phil Storey

PacketLife.Net Blog

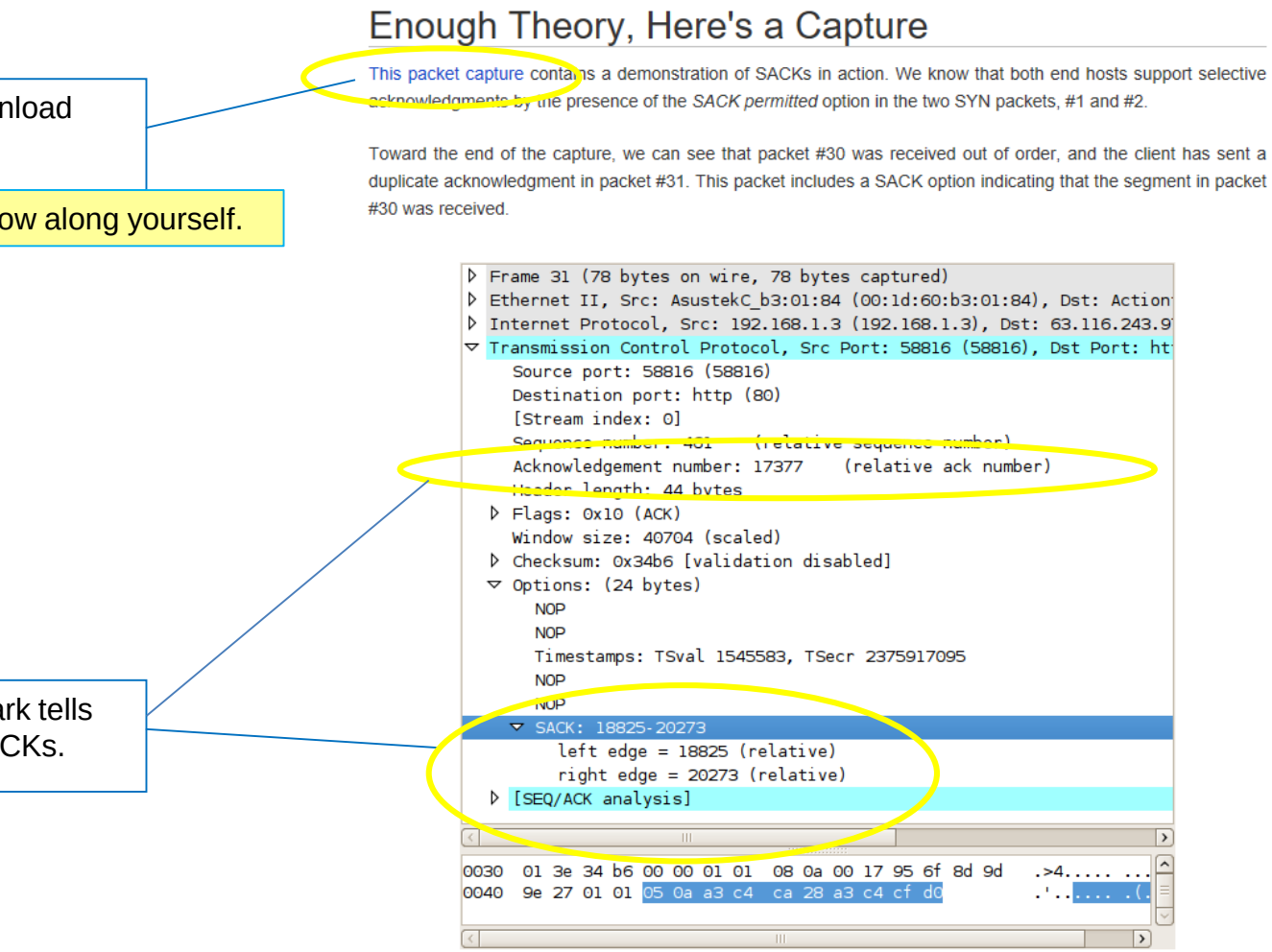
The capture file, “TCP_SACK.CAP”, used in this video came from a PacketLife.Net blog by Jeremy Stretch.

http://packetlife.net/blog/2010/jun/17/tcp-selective-acknowledgments-sack/

Click this link to download “TCP_SACK.cap”

Download the CAP file and follow along yourself.

This is what Wireshark tells us about Selective ACKs.



Wireshark Packet List

Here Wireshark tells us the story:

#1, #2, #3 are a 3-way handshake. RTT = 22ms. **Both sides support the SACK Option.**
#4 is a HTTP GET from the client. #5 is the server's ACK to that GET.
#6 - #28 are server data packets, with corresponding normal ACKs.

#29 is the last normal ACK before a "gap".
#30 is a data packet that followed a "gap". There should have been a data packet before #30.
#31, #33, #35 and #37 are SACKs that followed data packets #30, #32, #34 and #36. A SACK also acts as Dup-ACK.
#38 is a retransmission of data that "fills the gap". This data should have been before #30 but was lost in the network.
#39 is a normal ACK that acknowledges all the data so far (effectively just acknowledging the retransmission, #38).

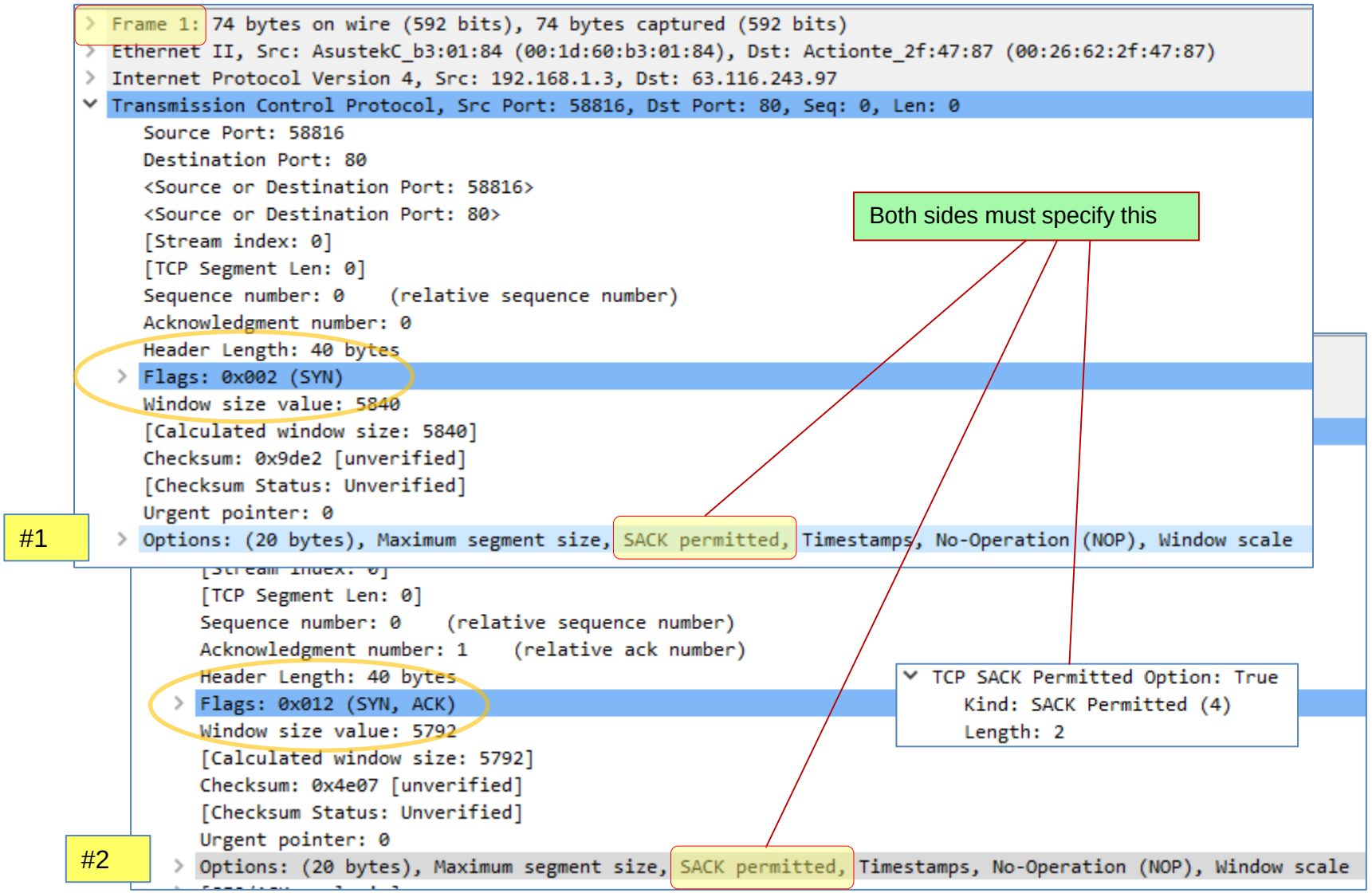
This 19ms Data-ACK is the shortest RTT?

No.	Time	Source	Destination	Protocol	Length	Info
1	0.000	192.168.1.3	63.116.243.97	TCP	74	58816→80 [SYN] Seq=0 Win=5840 Len=0 MSS=1460 SACK_PERM=1 TSval=1545573 TSecr=0 WS=128
2	0.022	63.116.243.97	192.168.1.3	TCP	74	80→58816 [SYN, ACK] Seq=0 Ack=1 Win=5792 Len=0 MSS=1460 SACK_PERM=1 TSval=2375917050 T
3	0.000	192.168.1.3	63.116.243.97	TCP	66	58816→80 [ACK] Seq=1 Ack=1 Win=5888 Len=0 TSval=1545575 TSecr=2375917050
4	0.000	192.168.1.3	63.116.243.97	HTTP	526	GET /service/gremlin/js/files/facebooklike,slideshow,itemSlider,schedule,textmessage,m
5	0.019	63.116.243.97	192.168.1.3	TCP	66	80→58816 [ACK] Seq=1 Ack=461 Win=6880 Len=0 TSval=2375917073 TSecr=1545575
6	0.002	63.116.243.97	192.168.1.3	TCP	1514	[TCP segment of a reassembled PDU]
7	0.000	192.168.1.3	63.116.243.97	TCP	66	58816→80 [ACK] Seq=461 Ack=1449 Win=8832 Len=0 TSval=1545577 TSecr=2375917073
8	0.000	63.116.243.97	192.168.1.3	TCP	1514	[TCP segment of a reassembled PDU]
9	0.000	192.168.1.3	63.116.243.97	TCP	66	58816→80 [ACK] Seq=461 Ack=2897 Win=11648 Len=0 TSval=1545577 TSecr=2375917073
10	0.000	63.116.243.97	192.168.1.3	TCP	1514	[TCP segment of a reassembled PDU]
11	0.000	192.168.1.3	63.116.243.97	TCP	66	58816→80 [ACK] Seq=461 Ack=4345 Win=14592 Len=0 TSval=1545577 TSecr=2375917073
12	0.000	63.116.243.97	192.168.1.3	TCP	1514	[TCP segment of a reassembled PDU]
13	0.000	192.168.1.3	63.116.243.97	TCP	66	58816→80 [ACK] Seq=461 Ack=5793 Win=17536 Len=0 TSval=1545577 TSecr=2375917073

No.	Time	Source	Destination	Protocol	Length	Info
27	0.000	192.168.1.3	63.116.243.97	TCP	66	58816→80 [ACK] Seq=461 Ack=15929 Win=37760 Len=0 TSval=1545580 TSecr=2375917095
28	0.001	63.116.243.97	192.168.1.3	TCP	1514	[TCP segment of a reassembled PDU]
29	0.000	192.168.1.3	63.116.243.97	TCP	66	58816→80 [ACK] Seq=461 Ack=17377 Win=40704 Len=0 TSval=1545580 TSecr=2375917095
30	0.027	63.116.243.97	192.168.1.3	TCP	1514	[TCP Previous segment not captured] [TCP segment of a reassembled PDU]
31	0.000	192.168.1.3	63.116.243.97	TCP	78	[TCP Dup ACK 29#1] 58816→80 [ACK] Seq=461 Ack=17377 Win=40704 Len=0 TSval=154558
32	0.000	63.116.243.97	192.168.1.3	TCP	1514	[TCP segment of a reassembled PDU]
33	0.000	192.168.1.3	63.116.243.97	TCP	78	[TCP Dup ACK 29#2] 58816→80 [ACK] Seq=461 Ack=17377 Win=40704 Len=0 TSval=154558
34	0.001	63.116.243.97	192.168.1.3	TCP	1514	[TCP segment of a reassembled PDU]
35	0.000	192.168.1.3	63.116.243.97	TCP	78	[TCP Dup ACK 29#3] 58816→80 [ACK] Seq=461 Ack=17377 Win=40704 Len=0 TSval=154558
36	0.000	63.116.243.97	192.168.1.3	TCP	1288	[TCP segment of a reassembled PDU]
37	0.000	192.168.1.3	63.116.243.97	TCP	78	[TCP Dup ACK 29#4] 58816→80 [ACK] Seq=461 Ack=17377 Win=40704 Len=0 TSval=154558
38	0.030	63.116.243.97	192.168.1.3	TCP	1514	[TCP Retransmission] 80→58816 [ACK] Seq=17377 Ack=461 Win=6880 Len=1448 TSval=237
39	0.000	192.168.1.3	63.116.243.97	TCP	66	58816→80 [ACK] Seq=461 Ack=24391 Win=43520 Len=0 TSval=1545586 TSecr=2375917152

Wireshark SYN / SYN-ACK

SACKs are a TCP Option and cannot be used within a TCP connection unless both ends agree in the TCP Header Options fields within the 3-way handshake. Here is the Wireshark information from the first two packets in our capture.



Wireshark ACKs and SACKS

The 6 boxes here show the TCP sequence number information contained within the last 6 ACKs and SACKs in the capture. The “left” edge of the first SACK (18825) is 1448 larger than the 17377 in the prior normal ACK. This tells us that the “gap” was size 1448. The SACK “right” edges increment by 1448 but the “left” remains unchanged at 18825 until the retransmitted data packet (remember #38) fills that 1448 sized “gap”. A final normal ACK ends the SACK sequence by acknowledging all data.

#29

```
Frame 29: 66 bytes on wire (528 bits), 66 bytes captured (528 bits)
et II, Src: AsustekC_b3:01:84 (00:1d:60:b3:01:84), Dst: Actionte_2f:47:87 (00:26:62:2
et Protocol Version 4, Src: 192.168.1.3, Dst: 63.116.243.97
Transmission Control Protocol, Src Port: 58816, Dst Port: 80, Seq: 461, Ack: 17377, Len: 0
```

Size of “gap” is
#31-#29: 18825 – 17377 = 1448

#31

```
Transmission Control Protocol, Src Port: 58816, Dst Port: 80, Seq: 461, Ack: 17377, Len: 0
SACK: 18825-20273
    Kind: SACK (5)
    Length: 10
    left edge = 18825 (relative)
    right edge = 20273 (relative)
    [TCP SACK Count: 1]
```

#33

```
Seq: 461, Ack: 17377, Len: 0
SACK: 18825-21721
    Kind: SACK (5)
    Length: 10
    left edge = 18825 (relative)
    right edge = 21721 (relative)
    [TCP SACK Count: 1]
```

#35

```
Seq: 461, Ack: 17377, Len: 0
SACK: 18825-23169
    Kind: SACK (5)
    Length: 10
    left edge = 18825 (relative)
    right edge = 23169 (relative)
    [TCP SACK Count: 1]
```

#37

```
Seq: 461, Ack: 17377, Len: 0
SACK: 18825-24391
    Kind: SACK (5)
    Length: 10
    left edge = 18825 (relative)
    right edge = 24391 (relative)
    [TCP SACK Count: 1]
```

Note the unchanging “left” edge – but incrementing “right” edge.
#31 : 20273 – 18825 = 1448
#33-#31: 21721 – 20273 = 1448
#35-#33: 23169 – 21721 = 1448
#37-#35: 24391 – 23169 = 1222

#37-#31: 24391 – 18825 = 5566

Wouldn’t it be nice if Wireshark subtracted these two numbers for us!

#39

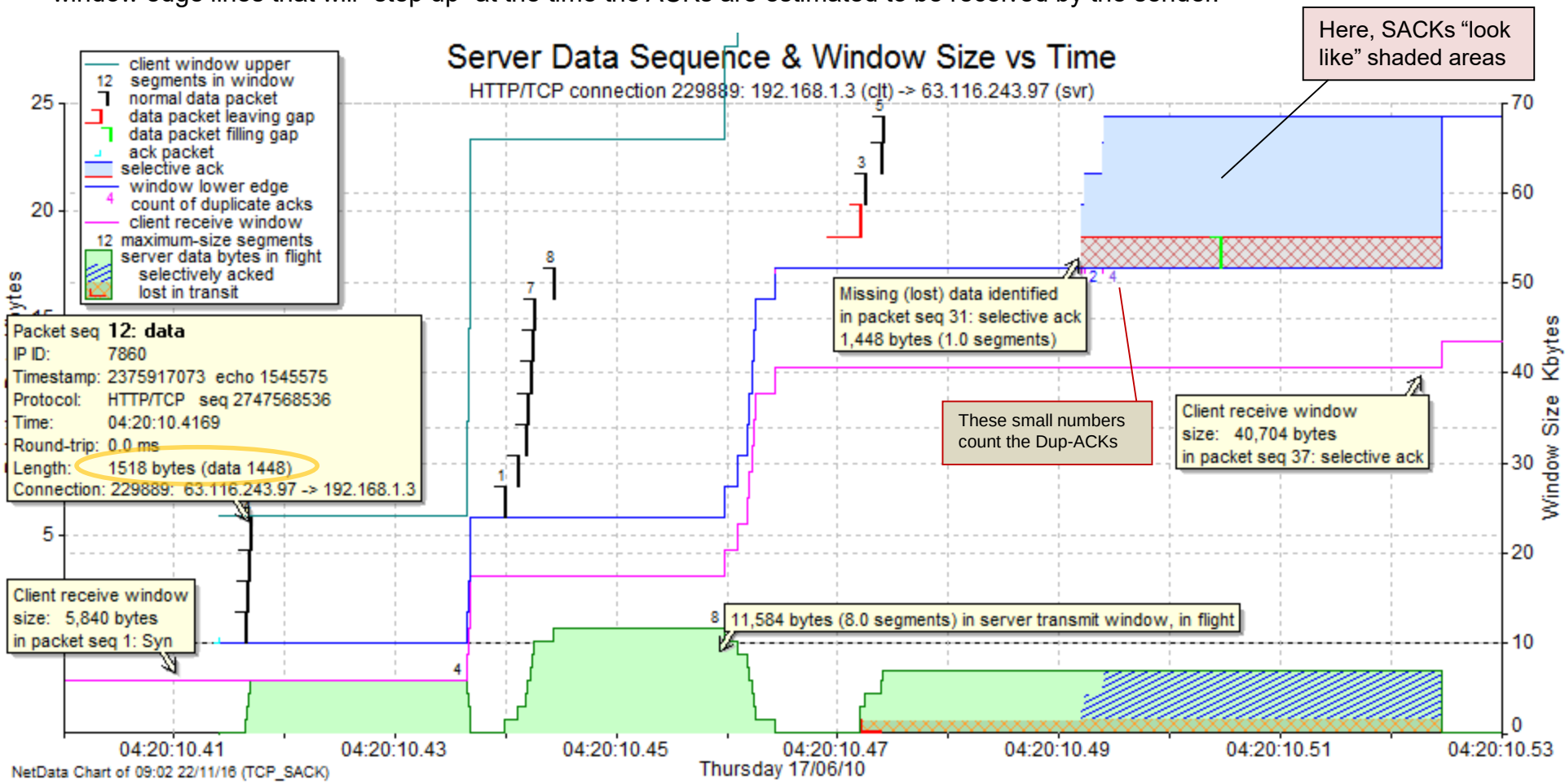
```
Frame 39: 66 bytes on wire (528 bits), 66 bytes captured (528 bits)
et II, Src: AsustekC_b3:01:84 (00:1d:60:b3:01:84), Dst: Actionte_2f:47:87 (00:26:62:2
et Protocol Version 4, Src: 192.168.1.3, Dst: 63.116.243.97
Transmission Control Protocol, Src Port: 58816, Dst Port: 80, Seq: 461, Ack: 24391, Len: 0
```

24391 represents all the data so far. This last normal ACK brings us to a fully ACKed state.

NetData – Packet Flow Chart

Packets (segments) are now vertical strips, with height representing TCP payload size.
Observe that our “full” payload sizes here are 1448 instead of the usual 1460 because TCP Timestamps are enabled.
The left y-axis measures bytes transferred (TCP Sequence Numbers) and the x-axis is time of day. This means that normal flows will work up from the bottom left to the top right of the chart. Chart items related to the various TCP windows use the right y-axis.

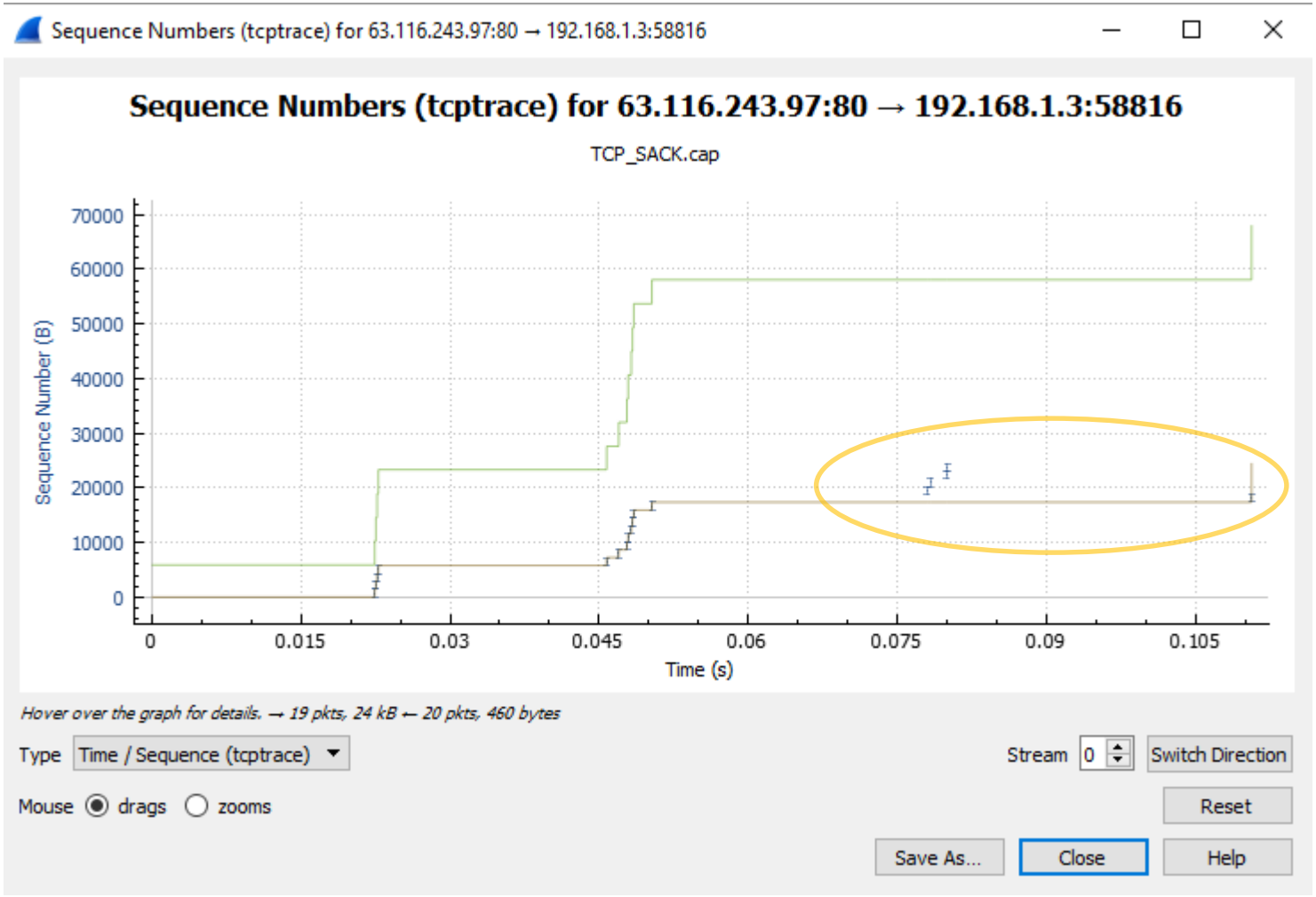
Shapes, colours and position are used to present different packet types and sizes. TCP acknowledgements are displayed as window edge lines that will “step up” at the time the ACKs are estimated to be received by the sender.



Wireshark: TCP-Trace Chart

Packets (segments) are vertical strips, with height representing TCP payload size.
We can see our block of 4, block of 8, block of 4 (with one missing) then the retransmission of 1 to fill the gap.
The grey line under the packets is the ACK line (similar to NetData) but it is drawn as we'd see normal ACKS.
There is no indication or hint of SACK behaviour here though.

Wireshark developers please take note – could you make Wireshark display SACK information?



SACK Fun Facts!

RFC 2018: https://tools.ietf.org/html/rfc2018

SACKs are a TCP Option – they can't be used unless both sides agree during the 3-way handshake.

SACK left/right sequence information goes in the TCP Options extended header.

When any optional TCP headers are used, the TCP data payload is reduced accordingly*.

One left/right pair of sequence numbers acknowledges one contiguous data sequence.
- By inference, each pair of sequence numbers announces the existence of one sequence gap.

All SACKs also contain the "normal" data acknowledgement sequence number in the usual "TCP Ack" field.
- SACKS are therefore also considered to be Duplicate-ACKs.

The TCP Options extended header has a maximum total of 40 bytes.
- Bytes consumed for a SACK pair = 2 + (8 per pair).
- 4 pairs = 2 + (8 x 4) = **34**.
- Thus, there can only be no more than 4 x SACK sequence pairs. (Implying 4 x non-contiguous "gaps").
- If other TCP Options (such as Timestamps) are enabled, must have fewer SACK pairs.

SACKs are not "legally binding" – the sender of a SACK can renege later.

- Normal ACKs are the only method to officially acknowledge data.
- SACKs should have no effect on the receive window.

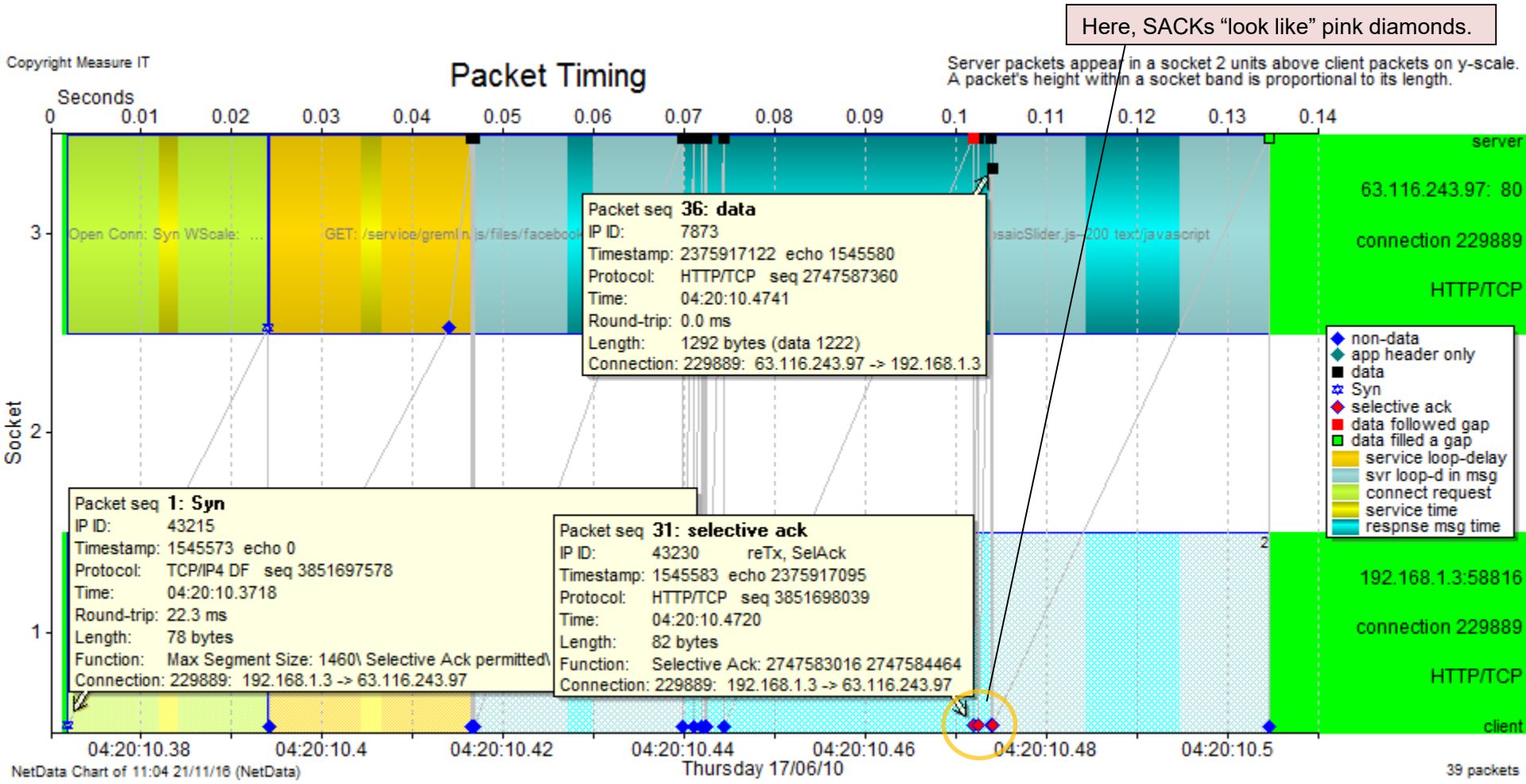
* In seven years of performing packet analysis, the author has seen only one example of SACK information included in a data packet. This was from a NetApp File Server. It seems that common practice is to include SACK information only in acknowledgement packets.

NetData – Packet Timing Chart

Client and server packets are displayed in their own different bands. Horizontal axis is time of day.

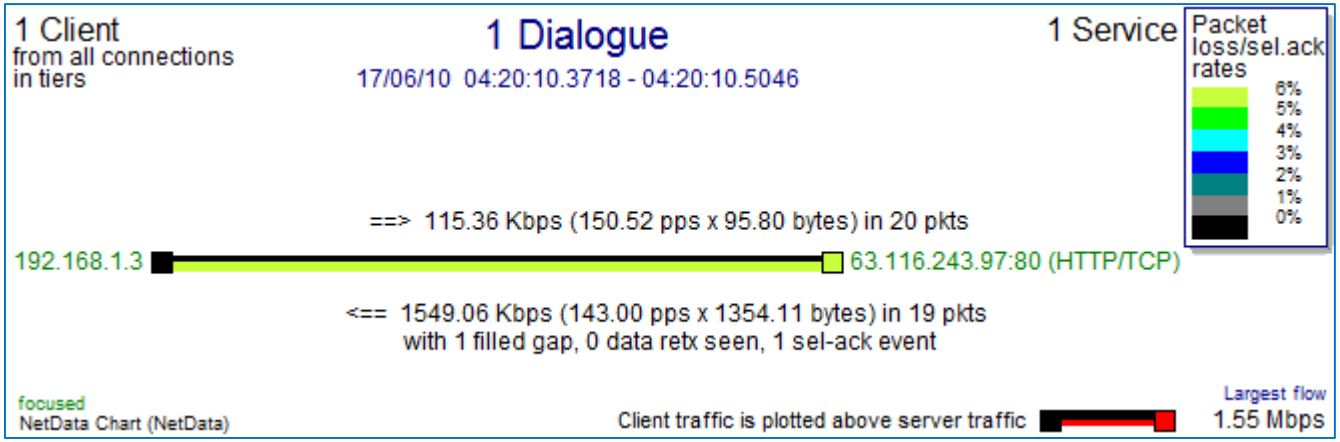
Shapes, colours and position are used to present different packet types and sizes. All items in the legend will appear somewhere on the chart.

The colours for the various transaction timings highlight the respective portions of the packet flow – and the background transaction descriptions help us to keep track of what we're looking at.



NetData – Dialogue Chart (TCP_SACK)

The Dialogue Chart (with Selective ACK chosen as the highlight) shows some summary information for this capture file. Further below are some table views for Transactions, Connections and Events.



Transactions of Project NetData

Plot All | Plot Only Selected | Deselect All | Addresses | Columns | Reset | Copy | Export | ? | Close

	Trn Key	Request Strt	Resp End	Type	Description	Rqst Dur	Strt Rsp	End Rsp	Resp Dur	ConnID	Client	Server	Data	LRqst	LResp	Frame
■	1	04:20:10.371775	04:20:10.394075	TCP	Open Conn: Syn WScale: 7 SelAck MSS:1460--Syn-Ack MSS:1460 SelAck WScale: 5 - Ack		0.0223	0.0223	0.0000	229889	192.168.1.3	63.116.243.97	WScale: 7 SelAck...	78	78	1
◆	2	04:20:10.394093	04:20:10.504636	HTTP/TCP	GET: /service/gremlin/js/files/facebooklike_slideshow_itemSlider_schedule_textmessage_mosaic_mosaicSlider.js--200 text/javascript		0.0224	0.1105	0.0881	229889	192.168.1.3	63.116.243.97		460	24390	4

Connections or Streams

Re-plot | Sessions | Addresses | Columns | Reset | Copy | Export | Close

ConnID	UserID	Type	Client (Caller)	cPort	Server (Callee)	sPort	First Packet	Closing	Closure	Total sec	Trips	Clt Pkts	ReTx	Kbps	Svr Pkts	ReTx	Kbps
229889		HTTP/TCP	192.168.1.3	58816	63.116.243.97	80	opn04:20:10.3718	> 04:20:10.5046		0.1329	1	20		15.328	19		205.824

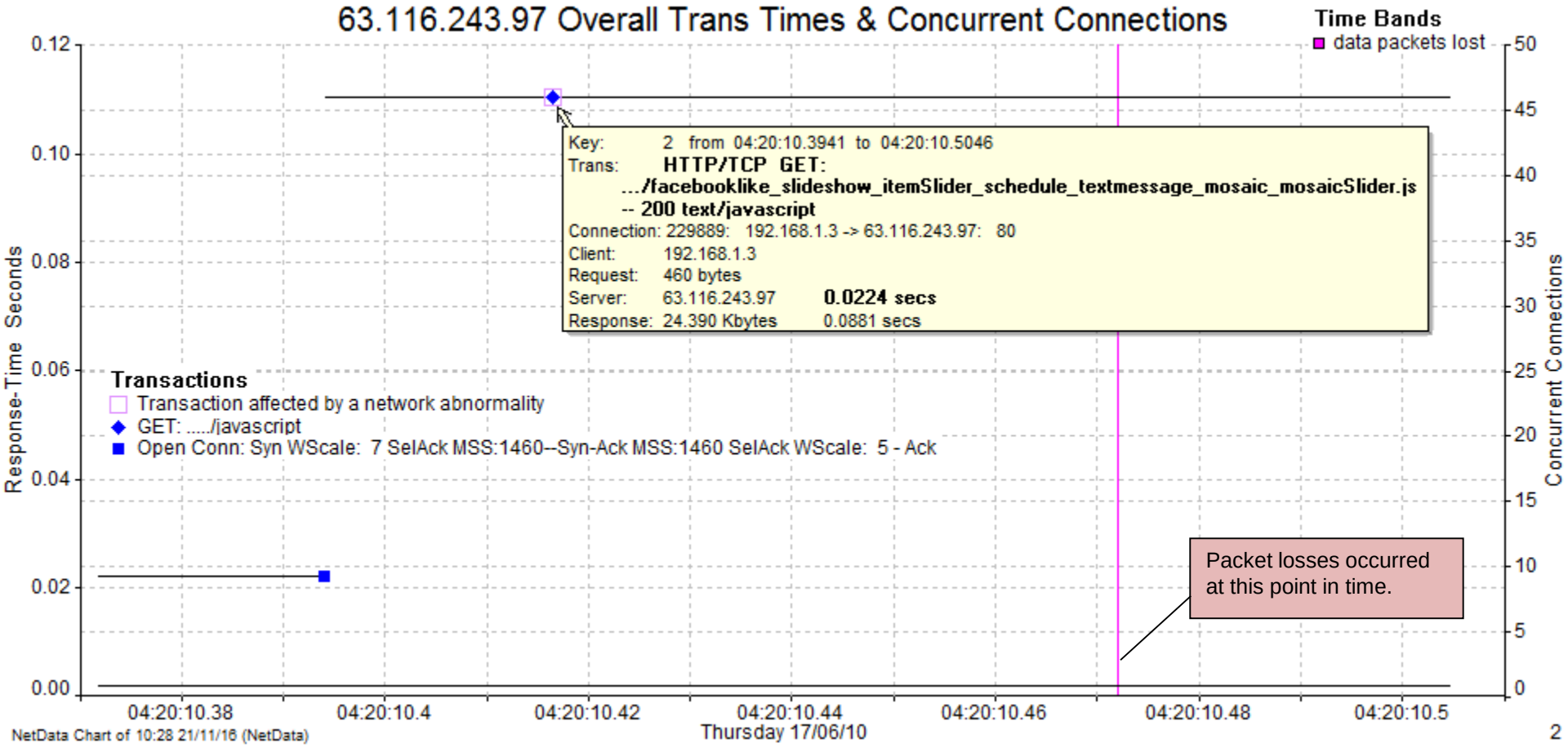
Events and Warnings

Filter | Stats | Plot Selected Events | Addresses | Columns | Reset | Copy | Export | Close

Start	End	Category	Description	Type	Duration	Client/Expected	Server	Conn/Sample	Frame
04:20:10.472026	04:20:10.504636	Pkt Loss	server filled all gaps after 0.0326 secs	HTTP	0.0326	192.168.1.3	63.116.243.97: 80	229889	38
04:20:10.472035	04:20:10.504645	TCP SAck	client acknowledged data in all gaps after 0.0326 secs	HTTP	0.0326	192.168.1.3	63.116.243.97: 80	229889	39

NetData - Performance Chart (Transactions)

Note the horizontal axis is “time of day” and the left vertical axis is transaction response time.
The CAP file contains one TCP connection, with a 22 ms 3-way handshake followed by a HTTP GET that took an overall time of around 110 ms. The delivery of the 24.4 KB response packets took 88 ms – and there were lost packets towards the end of the transaction.



NetData – Packet Table

NetData also provides various tables – that can easily be sorted and exported to CSV. This Packet Table lists all 39 packets that make up the "TCP_SACK.cap" file. This is sometimes useful to sort/track TTLs or IP IDs, sort by packet size, examine TCP Flags, etc.

The symbols used on the Packet Timing chart are also shown in the Packet Table for cross-referencing.

Time Of Day	Seq	Source	Destination	Len	H.	Net	TOS/TTL	IP ID	Tspt	Flags	ConnID	AppType	Data	Function	Content	Comment
04:20:10.371775	1	192.168.1.3:58816	63.116.243.97: 80	78	78	IP4 DF	TTL 64	43215	TCP	S	229889				Max Segment Size: 1460\ Selective Ack permitted\ Window Scale: 7	
04:20:10.39406	2	63.116.243.97: 80	192.168.1.3:58816	78	78	IP4 DF	TTL 56	0	TCP	A S	229889				Max Segment Size: 1460\ Selective Ack permitted\ Window Scale: 5	
04:20:10.394075	3	192.168.1.3:58816	63.116.243.97: 80	70	70	IP4 DF	TTL 64	43216	TCP	A	229889					Trans: Open Conn in 0.0223 secs
04:20:10.394093	4	192.168.1.3:58816	63.116.243.97: 80	530	70	IP4 DF	TTL 64	43217	TCP	AP	229889	HTTP	460	GET	GET /service/gremlin/js/files/facebooklike,slideshow,itemSlider,sche...	
04:20:10.414067	5	63.116.243.97: 80	192.168.1.3:58816	70	70	IP4 DF	TTL 56	7856	TCP	A	229889	HTTP				
04:20:10.416489	6	63.116.243.97: 80	192.168.1.3:58816	1518	70	IP4 DF	TTL 56	7857	TCP	A	229889	HTTP	1448	200 text/ja...	HTTP/1.1 200 OK\ Content-Length: 23858\^Content-Type: text/java...	
04:20:10.4165	7	192.168.1.3:58816	63.116.243.97: 80	70	70	IP4 DF	TTL 64	43218	TCP	A	229889	HTTP				
04:20:10.41661	8	63.116.243.97: 80	192.168.1.3:58816	1518	70	IP4 DF	TTL 56	7858	TCP	A	229889	HTTP	1448			
04:20:10.416616	9	192.168.1.3:58816	63.116.243.97: 80	70	70	IP4 DF	TTL 64	43219	TCP	A	229889	HTTP				
04:20:10.416733	10	63.116.243.97: 80	192.168.1.3:58816	1518	70	IP4 DF	TTL 56	7859	TCP	AP	229889	HTTP	1448			
04:20:10.416739	11	192.168.1.3:58816	63.116.243.97: 80	70	70	IP4 DF	TTL 64	43220	TCP	A	229889	HTTP				
04:20:10.416856	12	63.116.243.97: 80	192.168.1.3:58816	1518	70	IP4 DF	TTL 56	7860	TCP	A	229889	HTTP	1448			
04:20:10.416861	13	192.168.1.3:58816	63.116.243.97: 80	70	70	IP4 DF	TTL 64	43221	TCP	A	229889	HTTP				
04:20:10.439808	14	63.116.243.97: 80	192.168.1.3:58816	1518	70	IP4 DF	TTL 56	7861	TCP	A	229889	HTTP	1448			
04:20:10.439815	15	192.168.1.3:58816	63.116.243.97: 80	70	70	IP4 DF	TTL 64	43222	TCP	A	229889	HTTP				
04:20:10.441014	16	63.116.243.97: 80	192.168.1.3:58816	1518	70	IP4 DF	TTL 56	7862	TCP	AP	229889	HTTP	1448			
04:20:10.441024	17	192.168.1.3:58816	63.116.243.97: 80	70	70	IP4 DF	TTL 64	43223	TCP	A	229889	HTTP				
04:20:10.441846	18	63.116.243.97: 80	192.168.1.3:58816	1518	70	IP4 DF	TTL 56	7863	TCP	A	229889	HTTP	1448			
04:20:10.441852	19	192.168.1.3:58816	63.116.243.97: 80	70	70	IP4 DF	TTL 64	43224	TCP	A	229889	HTTP				
04:20:10.441968	20	63.116.243.97: 80	192.168.1.3:58816	1518	70	IP4 DF	TTL 56	7864	TCP	A	229889	HTTP	1448			
04:20:10.441974	21	192.168.1.3:58816	63.116.243.97: 80	70	70	IP4 DF	TTL 64	43225	TCP	A	229889	HTTP				
04:20:10.442285	22	63.116.243.97: 80	192.168.1.3:58816	1518	70	IP4 DF	TTL 56	7865	TCP	AP	229889	HTTP	1448			
04:20:10.442291	23	192.168.1.3:58816	63.116.243.97: 80	70	70	IP4 DF	TTL 64	43226	TCP	A	229889	HTTP				
04:20:10.442408	24	63.116.243.97: 80	192.168.1.3:58816	1518	70	IP4 DF	TTL 56	7866	TCP	A	229889	HTTP	1448			
04:20:10.442414	25	192.168.1.3:58816	63.116.243.97: 80	70	70	IP4 DF	TTL 64	43227	TCP	A	229889	HTTP				
04:20:10.442531	26	63.116.243.97: 80	192.168.1.3:58816	1518	70	IP4 DF	TTL 56	7867	TCP	A	229889	HTTP	1448			
04:20:10.442537	27	192.168.1.3:58816	63.116.243.97: 80	70	70	IP4 DF	TTL 64	43228	TCP	A	229889	HTTP				
04:20:10.444353	28	63.116.243.97: 80	192.168.1.3:58816	1518	70	IP4 DF	TTL 56	7868	TCP	A	229889	HTTP	1448			
04:20:10.444362	29	192.168.1.3:58816	63.116.243.97: 80	70	70	IP4 DF	TTL 64	43229	TCP	A	229889	HTTP				
04:20:10.47203	30	63.116.243.97: 80	192.168.1.3:58816	1518	70	IP4 DF	TTL 56	7870	TCP	AP	229889	HTTP	1448			
04:20:10.472037	31	192.168.1.3:58816	63.116.243.97: 80	82	82	IP4 DF	TTL 64	43230	TCP	A	229889	HTTP			Selective Ack: 2747583016 2747584464	
04:20:10.472411	32	63.116.243.97: 80	192.168.1.3:58816	1518	70	IP4 DF	TTL 56	7871	TCP	A	229889	HTTP	1448			1,448 bytes appended to saved fr...
04:20:10.472417	33	192.168.1.3:58816	63.116.243.97: 80	82	82	IP4 DF	TTL 64	43231	TCP	A	229889	HTTP			Selective Ack: 2747583016 2747585912	
04:20:10.473962	34	63.116.243.97: 80	192.168.1.3:58816	1518	70	IP4 DF	TTL 56	7872	TCP	A	229889	HTTP	1448			1,448 bytes appended to saved fr...
04:20:10.473968	35	192.168.1.3:58816	63.116.243.97: 80	82	82	IP4 DF	TTL 64	43232	TCP	A	229889	HTTP			Selective Ack: 2747583016 2747587360	
04:20:10.474064	36	63.116.243.97: 80	192.168.1.3:58816	1292	70	IP4 DF	TTL 56	7873	TCP	AP	229889	HTTP	1222			1,222 bytes appended to saved fr...
04:20:10.47407	37	192.168.1.3:58816	63.116.243.97: 80	82	82	IP4 DF	TTL 64	43233	TCP	A	229889	HTTP			Selective Ack: 2747583016 2747588582	
04:20:10.504636	38	63.116.243.97: 80	192.168.1.3:58816	1518	70	IP4 DF	TTL 56	7874	TCP	A	229889	HTTP	7014			Filled gap of 1,448 bytes precedin...
04:20:10.504645	39	192.168.1.3:58816	63.116.243.97: 80	70	70	IP4 DF	TTL 64	43234	TCP	A	229889	HTTP				

Phil Storey

www.NetworkDetective.com.au

au.linkedin.com/in/philipstorey3

@PhilStorey24

www.youtube.com/c/NetworkDetective