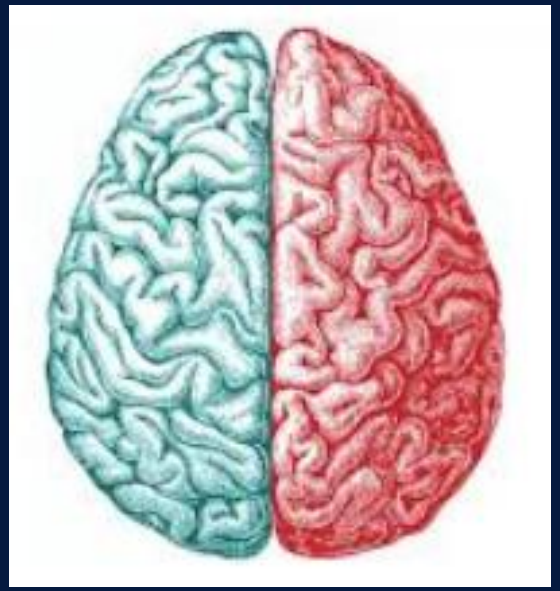
SharkFest '18 Europe

TCP Split Brain



Using Wireshark to Compare & Contrast behavior and TCP state of client vs. server

John Pittle

Riverbed Technologies
Performance Management Strategist

Premise: TCP Split Brain

- When troubleshooting TCP, you often have to consider both the sender's unique perspective and the receiver's unique perspective
- Both endpoints are independent, but at the same time, they do react to packets from the other end
- The joint behavior gets even more interesting when there's "high" latency in the path

Session Goals

- Compare and contrast TCP end point behavior
- Drill down into the “what is it doing?” and “why is it doing that?”
- Promote Wireshark Profiles Feature
- Share experience and ideas
- Expose you to visualizations that help reinforce the end point behavior we will be discussing

Session Resources

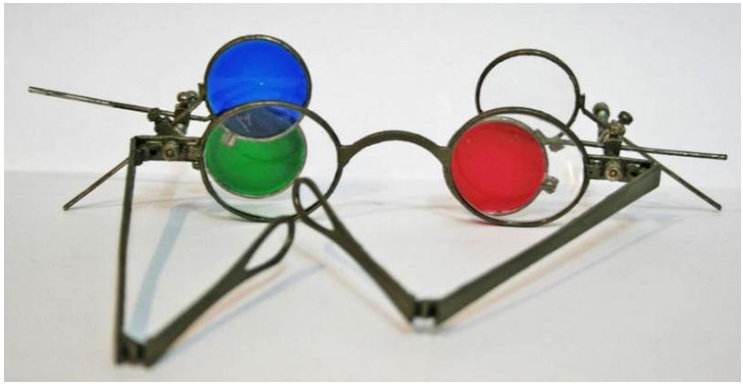
- Wireshark 2.6.3
- End point capture files for the TCP connection of interest
- Wireshark Profiles
- Riverbed Transaction Analyzer for Visualization

About me?

- Performance Engineering since 1980
- Protocol Analysis since 1991
- Professional Services with OPNET / Riverbed since 2005
- Love the mystery of a complicated performance issue
- Shaved off beard in 2003…

My Ask of You

- Engage
- Participate
- We have a lot of detailed material
- We will explore conflicting, contradictory, and possibly confusing details
- Ask Questions
- Question Answers




IT'S GO TIME

Application Scenario

- HTTPS Web Application
- Private key is not available
- Host based captures on web server and my laptop

Symptoms to Analyze

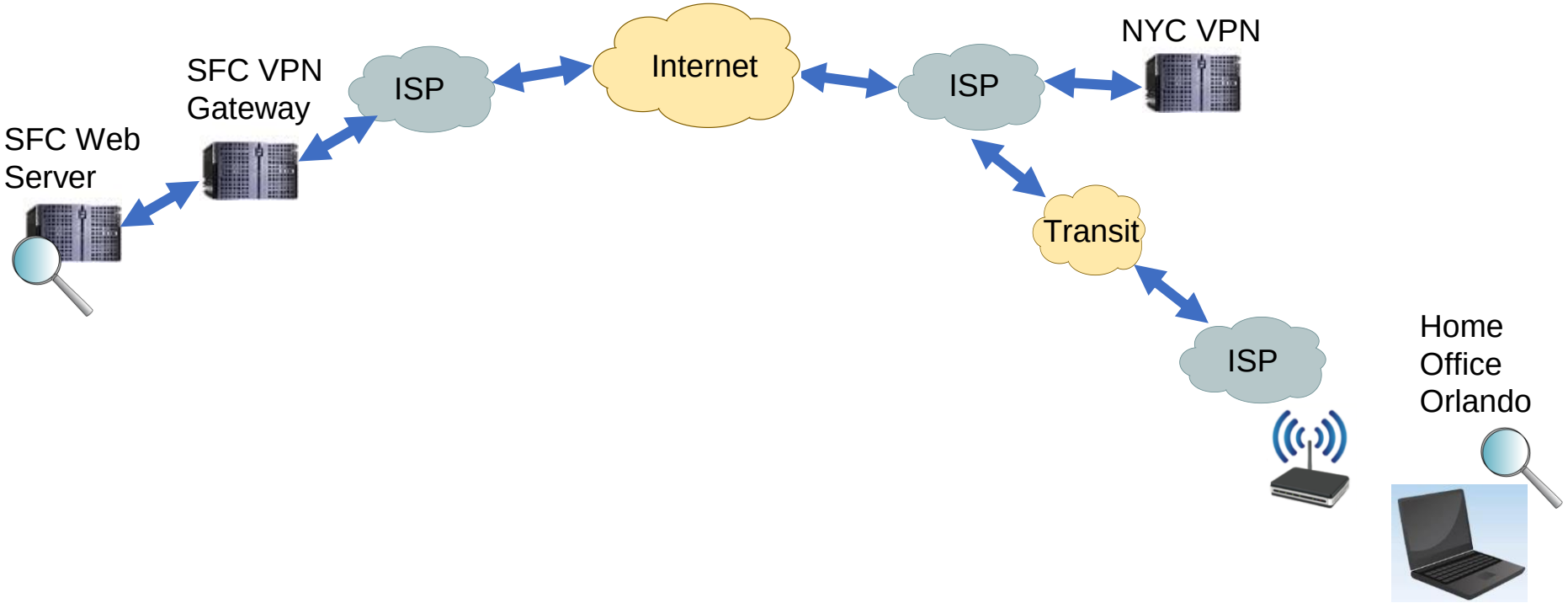
- Downloading files take *forever*
- 16 seconds to download a 1.4MB file
- One TCP connection has been isolated as the connection of interest – TCP/52942-443

Very Simple Topology

Web Server
San Francisco, CA

Home Office
Orlando, FL

Actual Topology

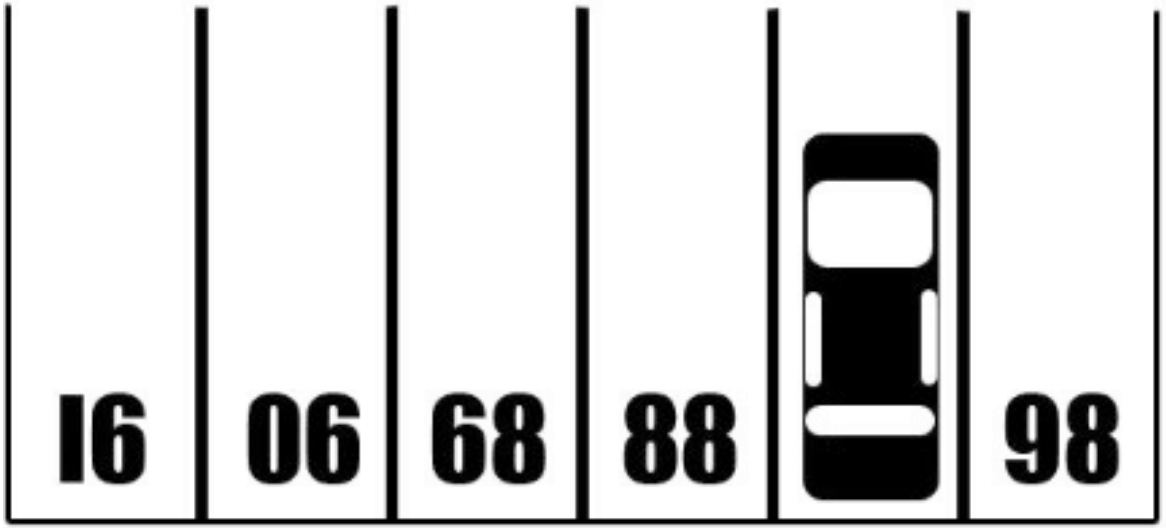


Pop Quiz #1

- Ready for our first Quiz?

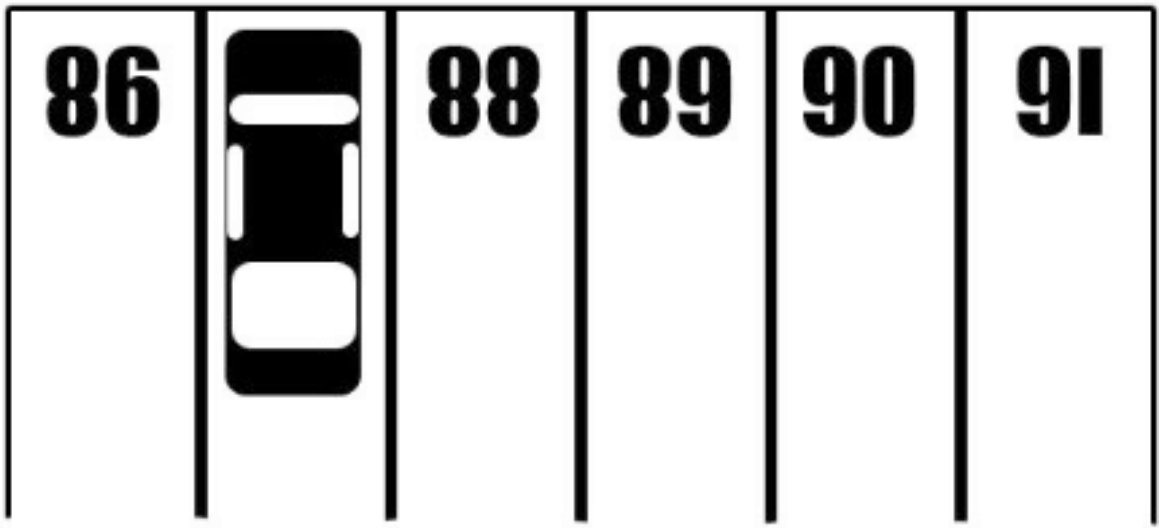
Pop Quiz #1

What parking slot # is the car in?
Can you do the math?



Perspective

How about now?

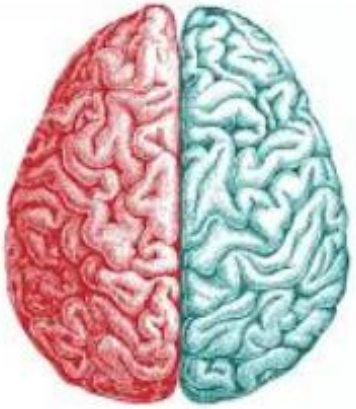


TCP End Point Behavior

- Each end point has a unique perspective
- Independent, yet influenced by the other
- Often during the traffic exchange, TCP stack decisions and actions can be occurring asynchronously on each host
- 3rd party actors can also have an affect on behavior

Comparisons

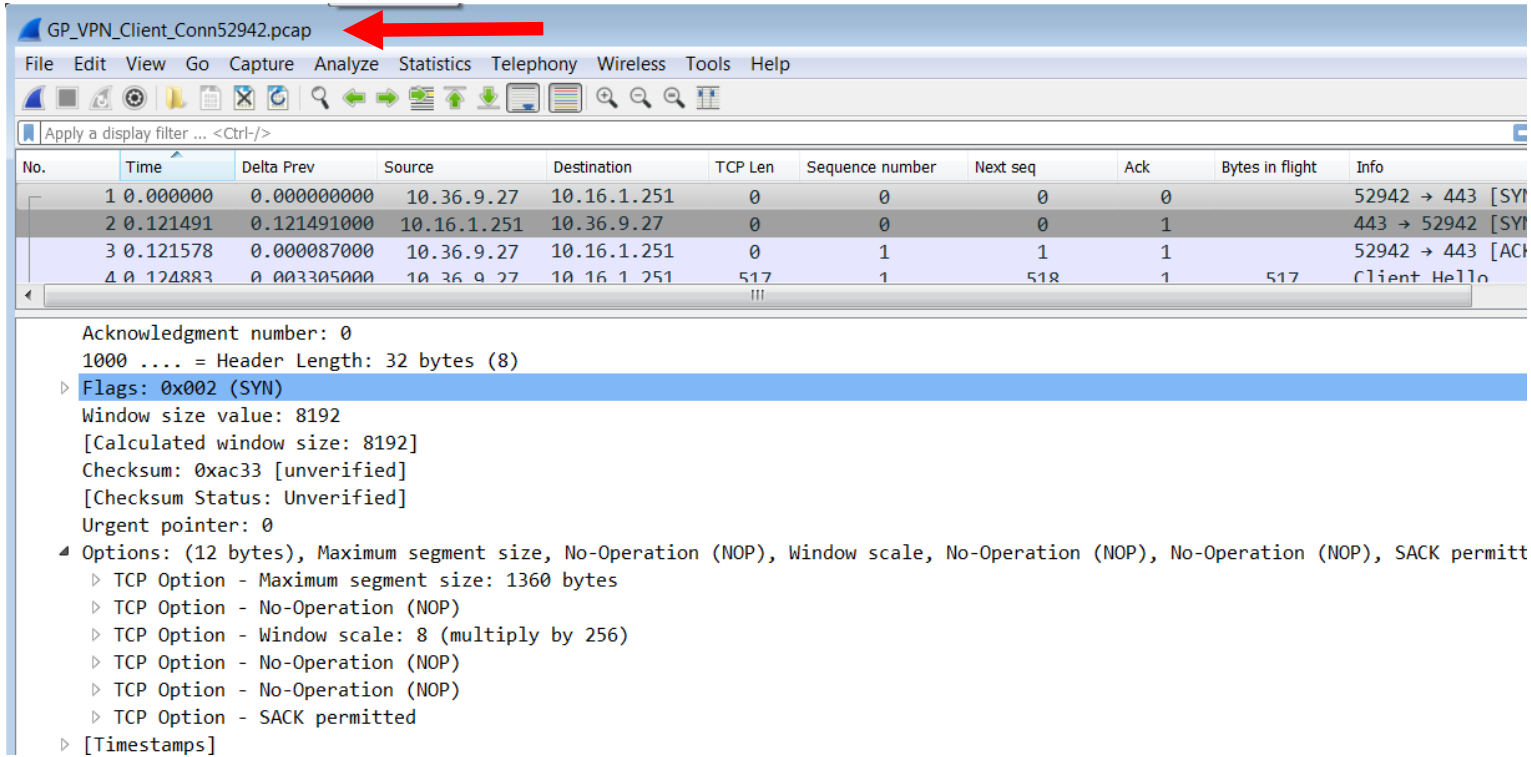
- 3-way handshake
- Latency
- Expert Info
- Fragment Overlaps / OOS / Packet Loss
- Bytes in Flight
- Congestion



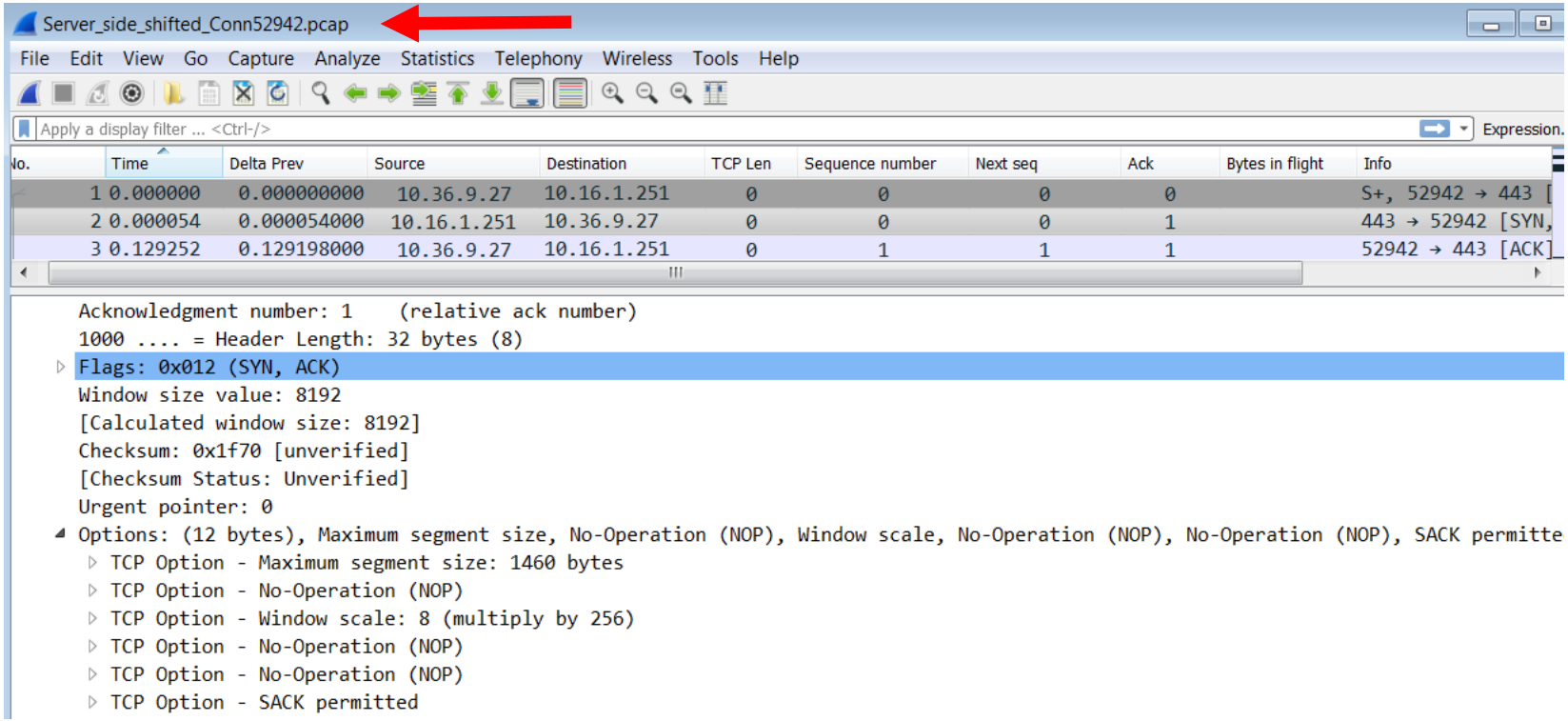
Split Brain Comparisons

- We'll start with the 3-way handshake signaling

3-Way Handshake - Client

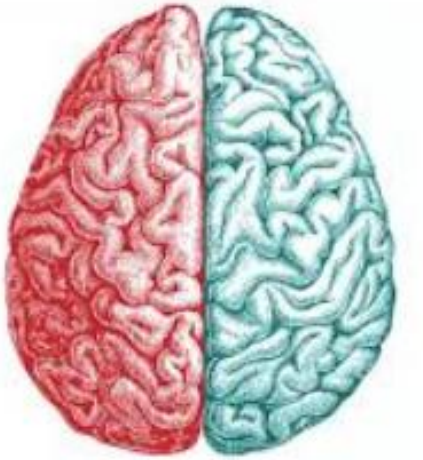


3-Way Handshake - Server



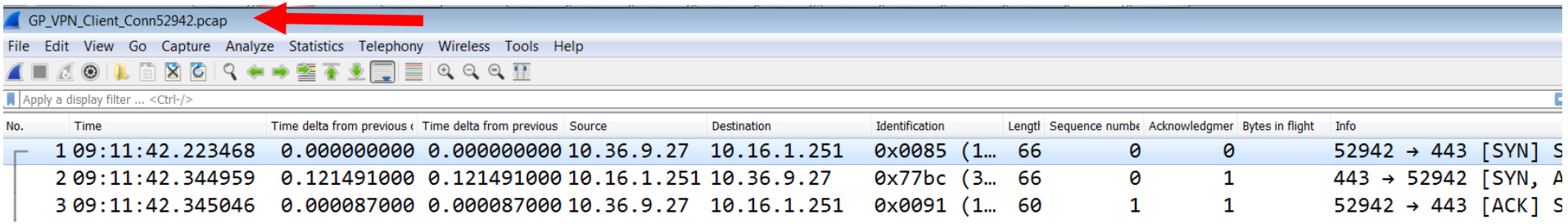
Split Brain Comparisons

- Latency Checks

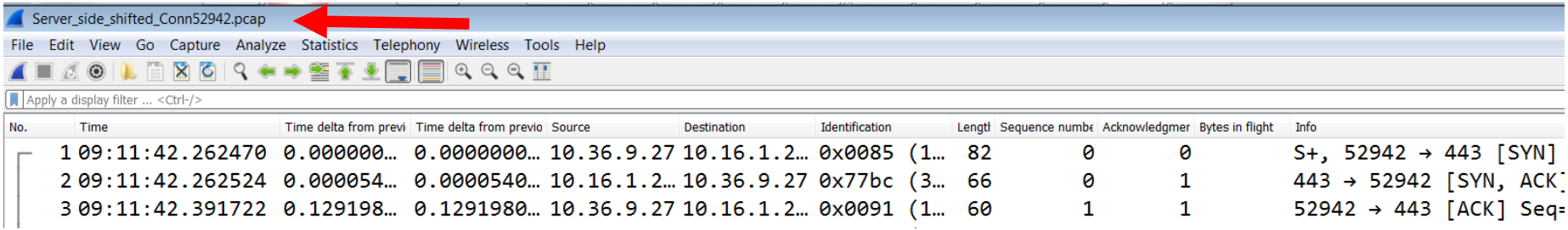


Latency Check

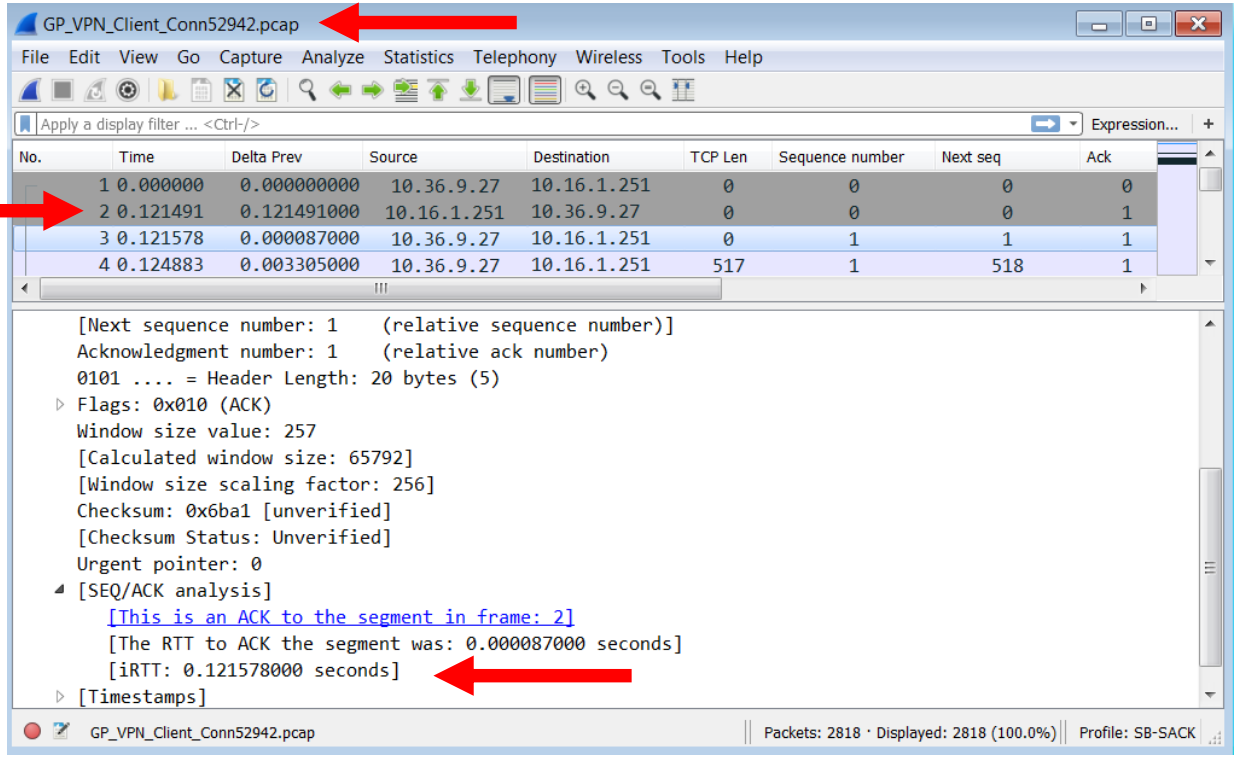
- Client Capture – 121ms



- Server Capture – 129ms



Wireshark i(nitial)RTT

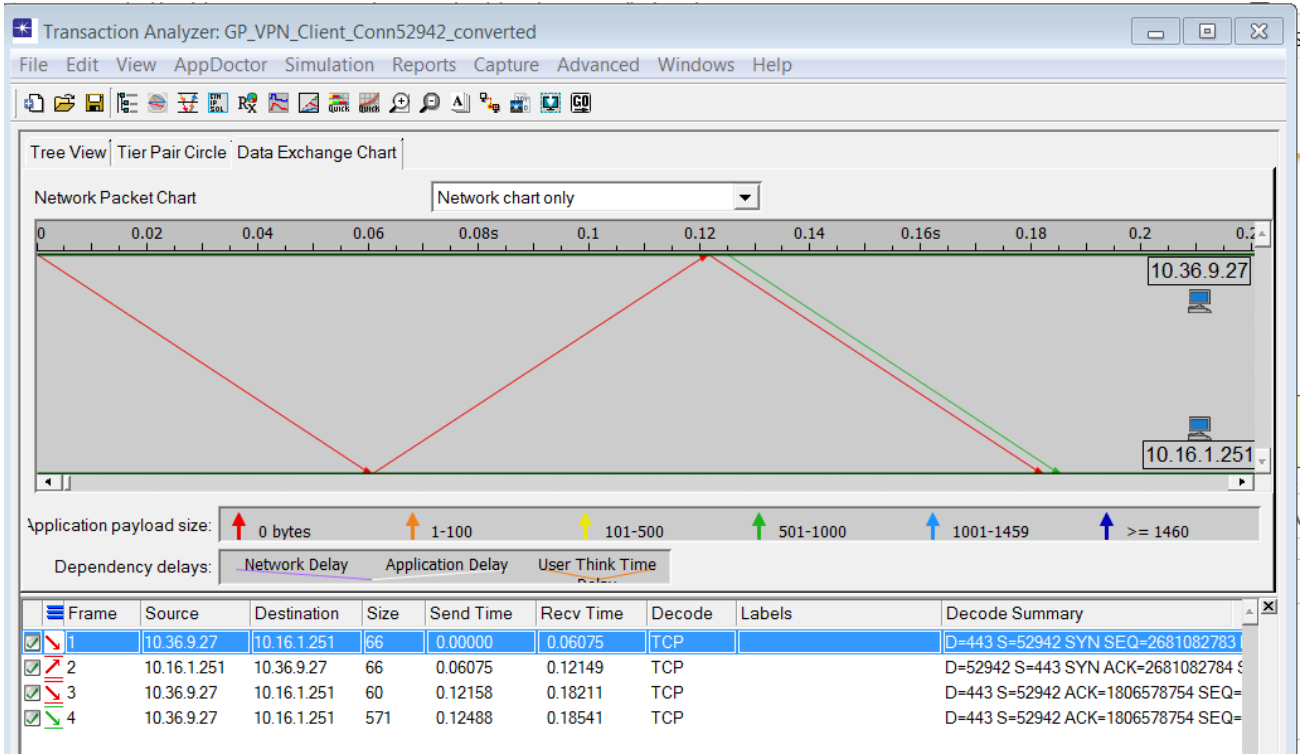


What if we could visualize the traffic exchanges?

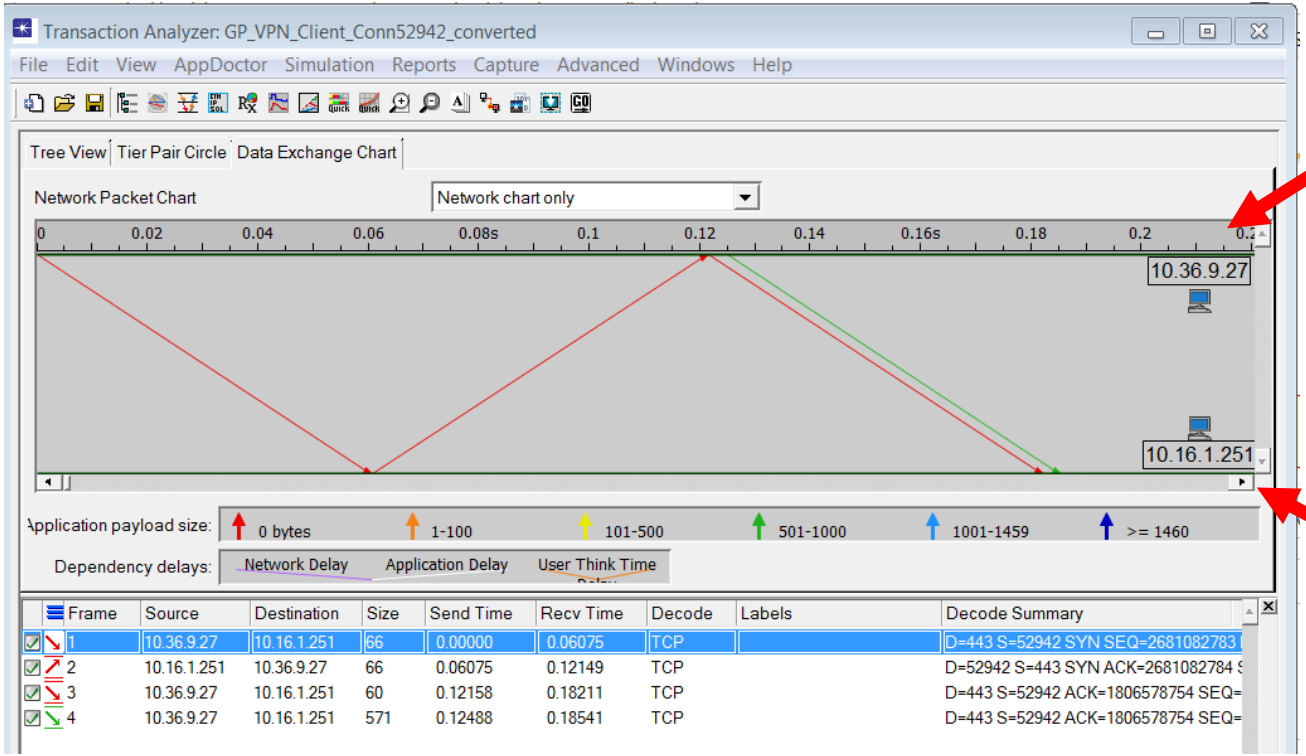
- Traffic bouncing between hosts
- Frequency, density, duration over time
- Expert analytics overlaid with the visualized traffic

- If only there were such a capability…

Visualize Traffic over Time



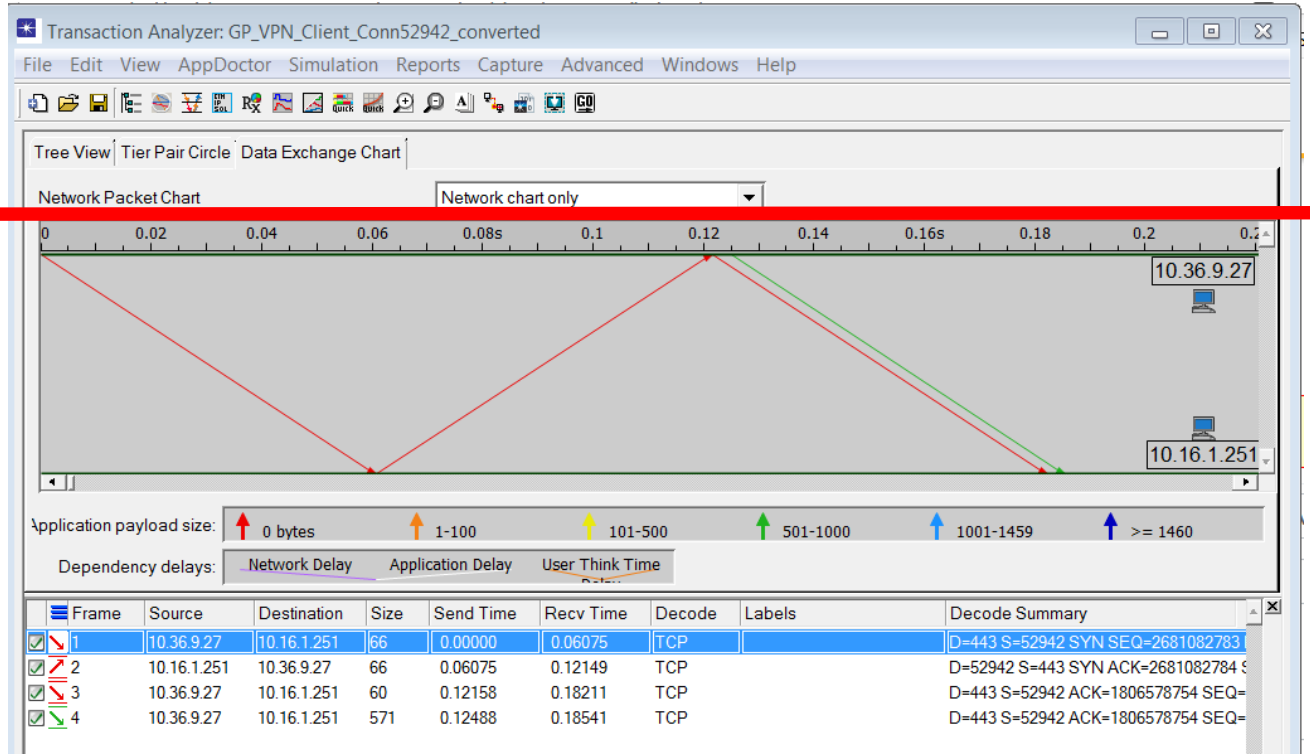
Top and Bottom of Swimlane



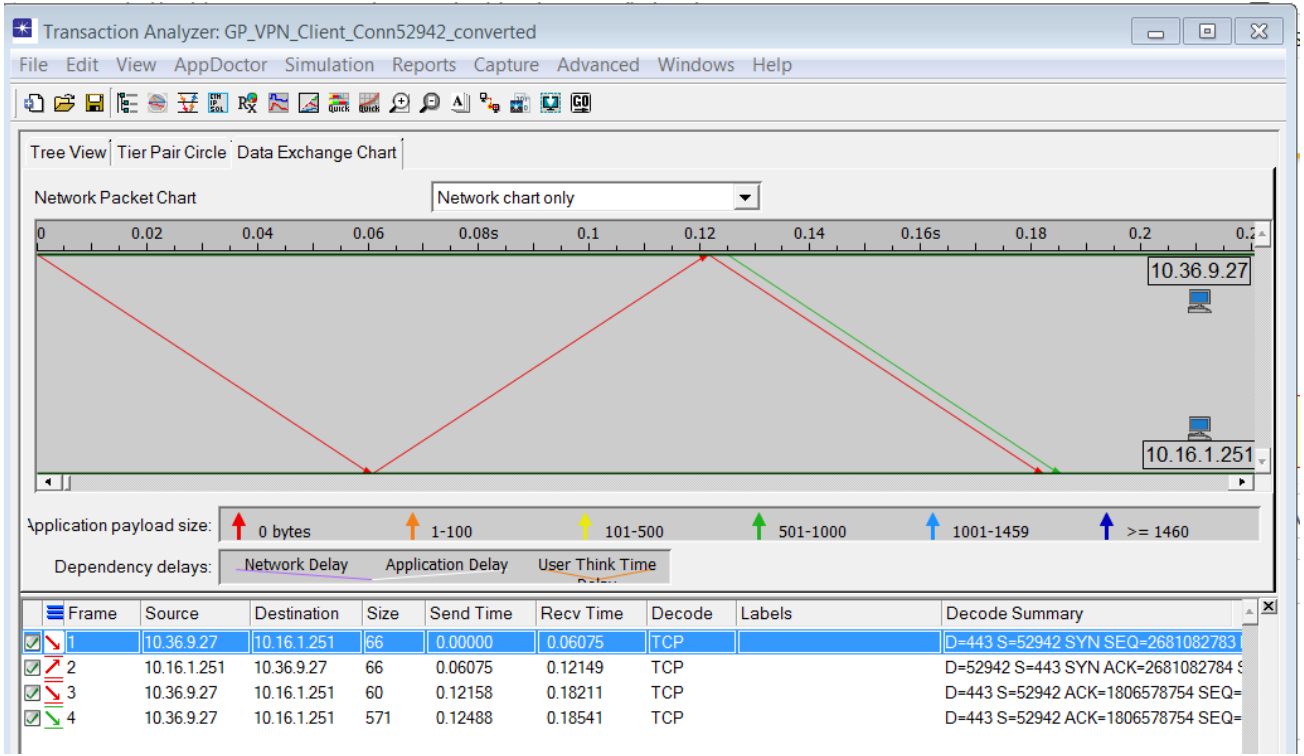
Client

Server

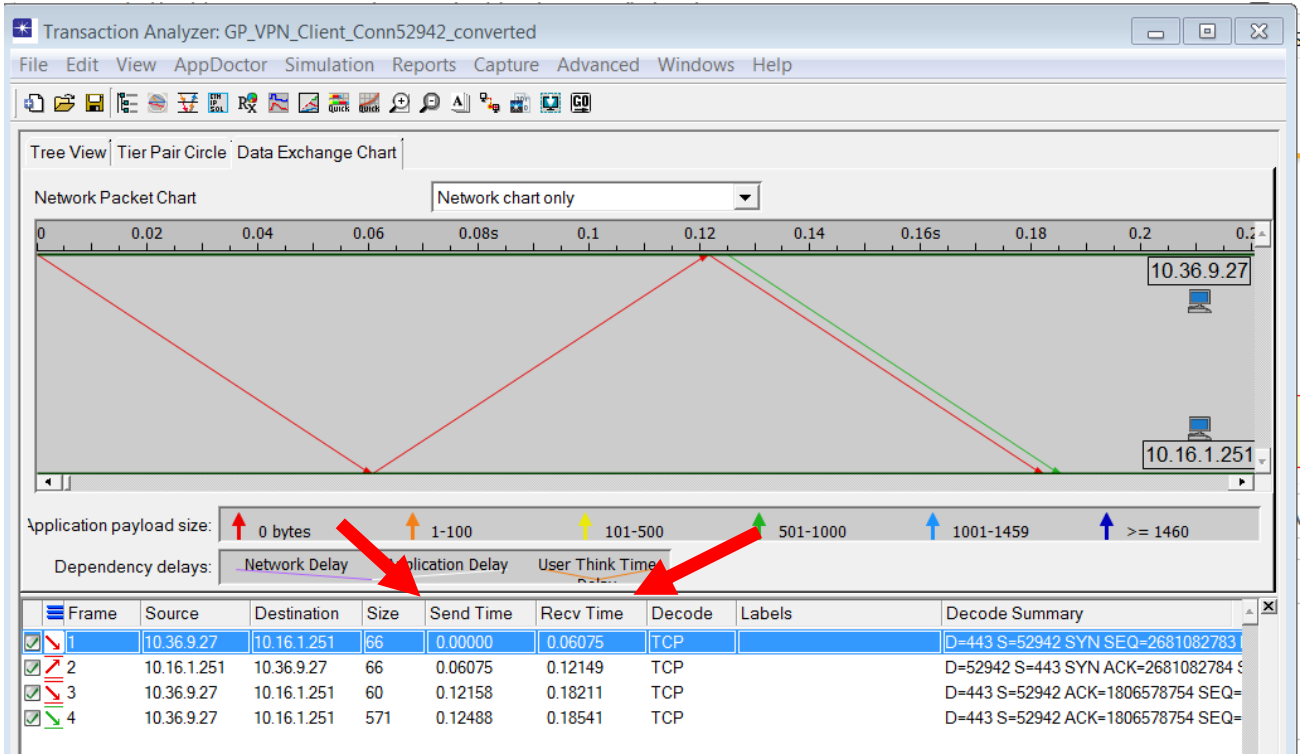
Time Moves Left to Right



Optional Summary Decodes



Send Time / Recv Time



Estimated vs. Actual Times

- Each capture has the actual times a packet is sent or received from / to that host

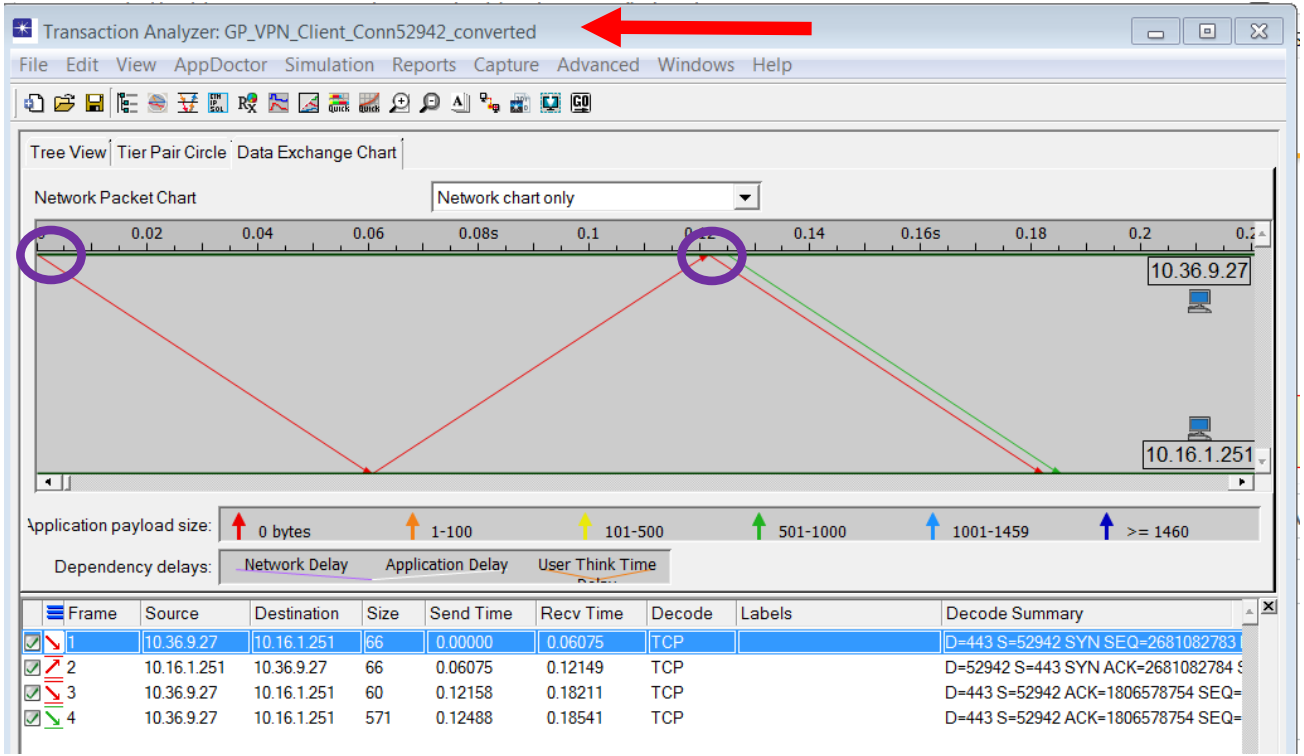
- Using what we do know, we can estimate what we don’t know

Estimated vs. Actual Times

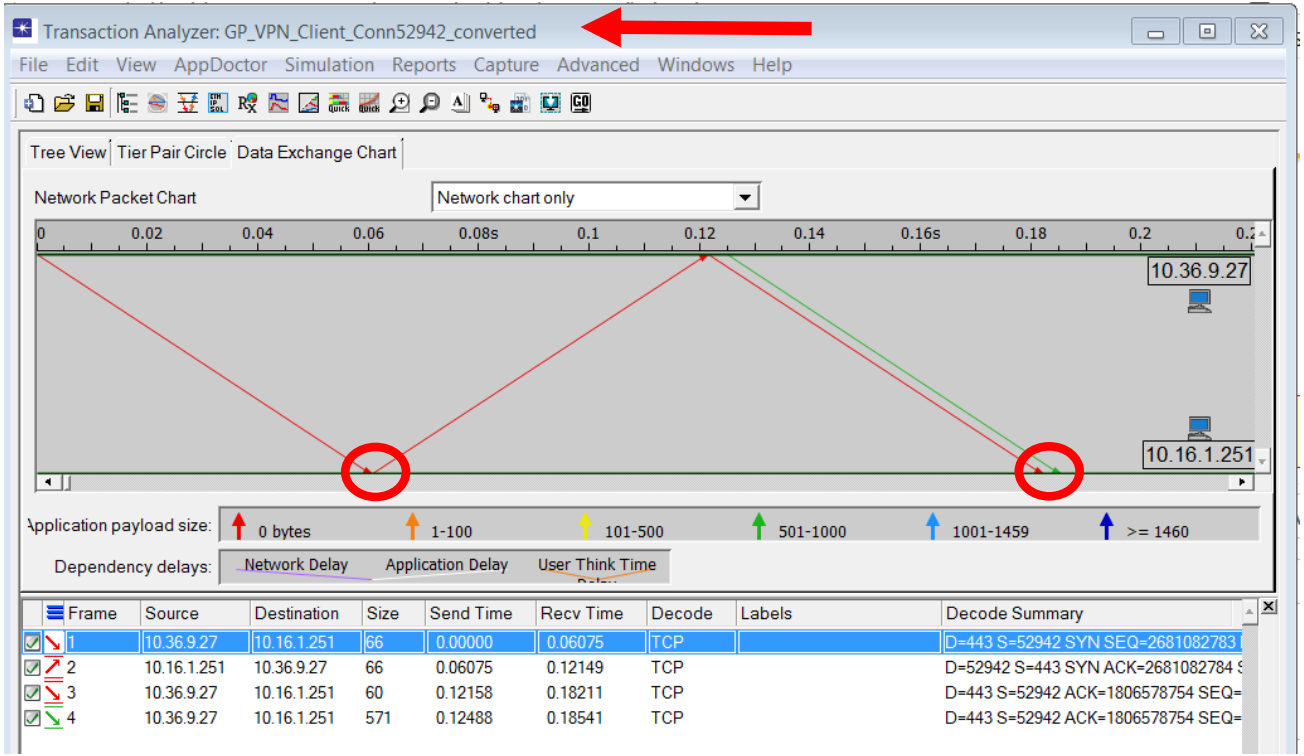
- If we know when we received a packet, we can make a reasonable estimate about the time it was transmitted

- If we know when a packet was transmitted, we can make a reasonable estimate about the time it was received

Here's what we know



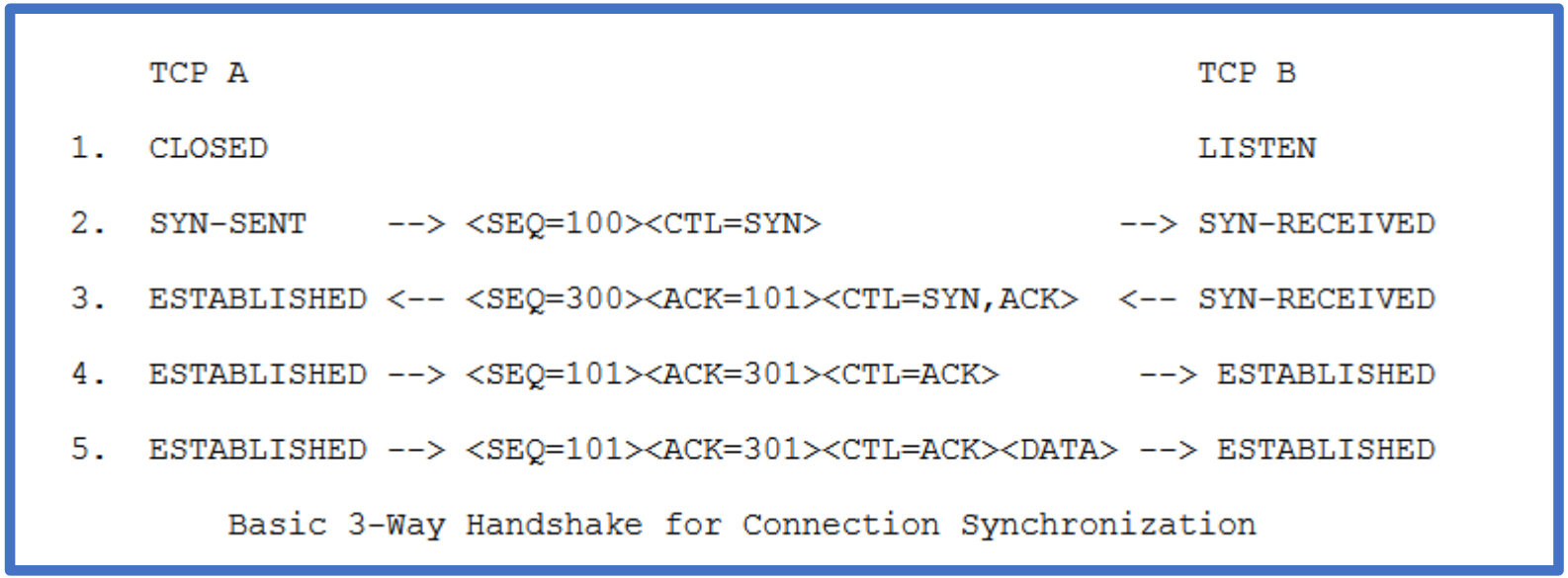
Here's what we can estimate



Affairs of State

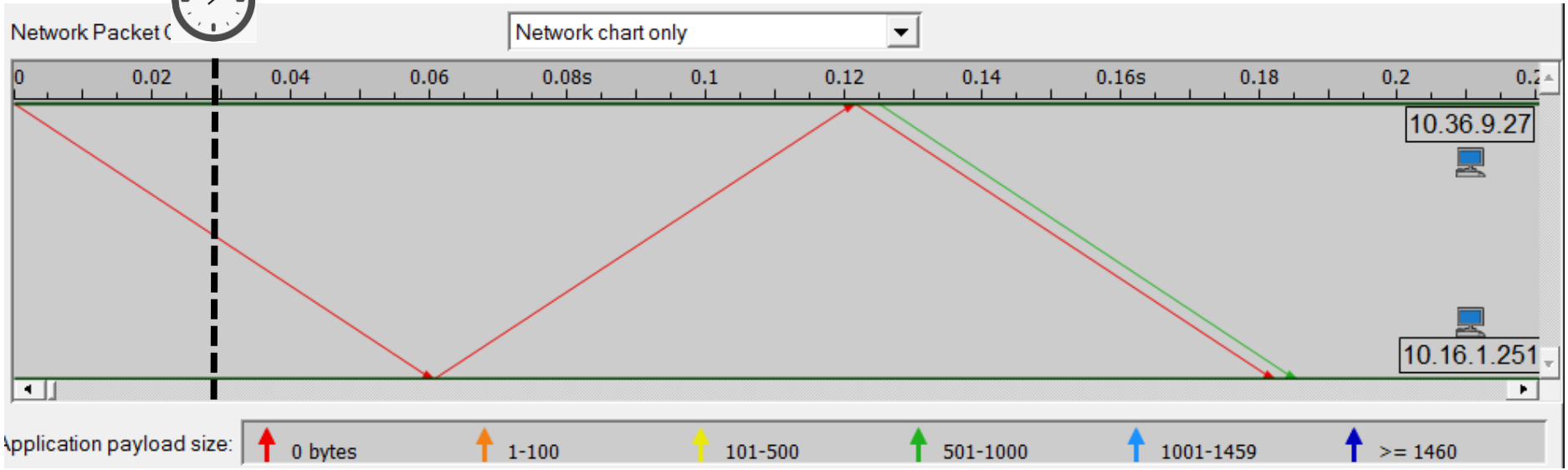
- Now let’s look at how we can use the visualization to understand the TCP state of each end point at a given point in time

RFC 793: TCP State Overview

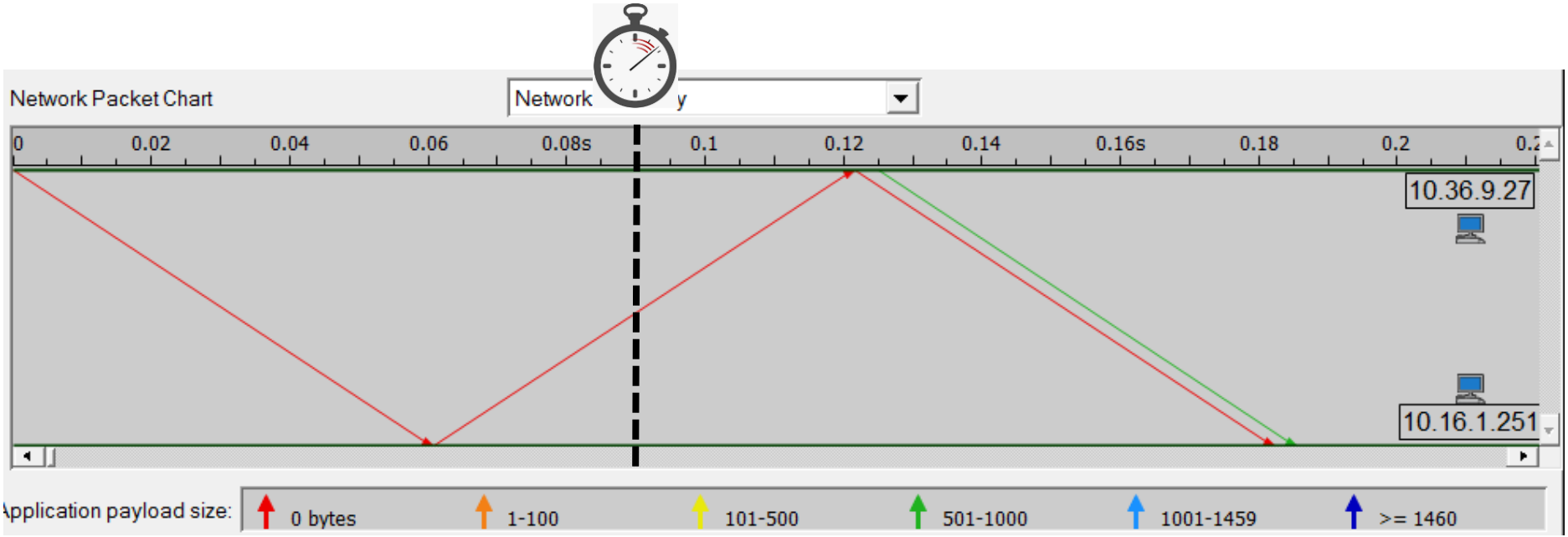


Split Brain – State Discussion

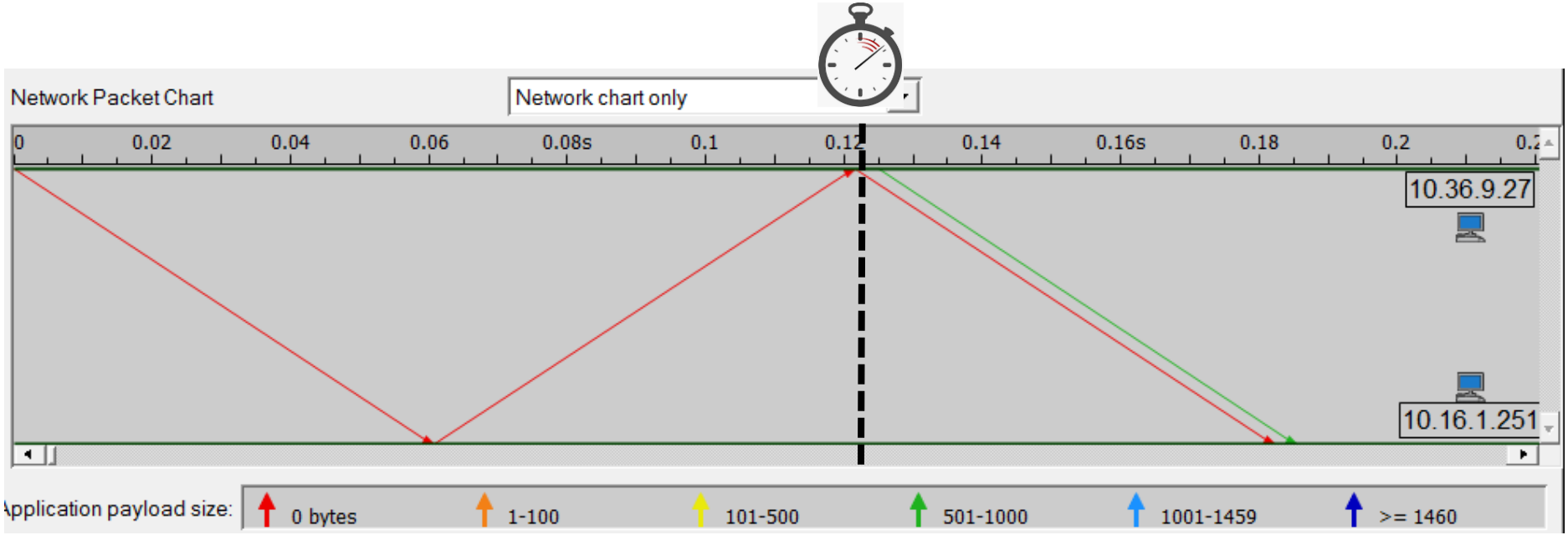
What is the TCP State of each host at this timepoint?



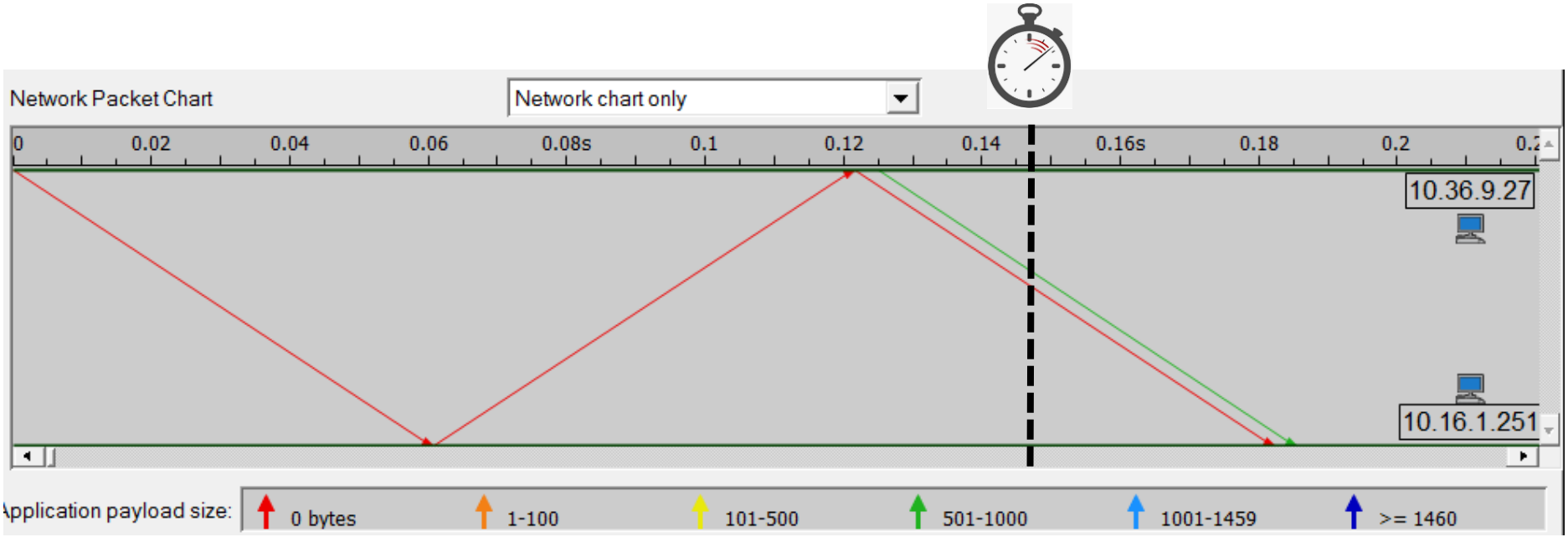
Split Brain – State Discussion



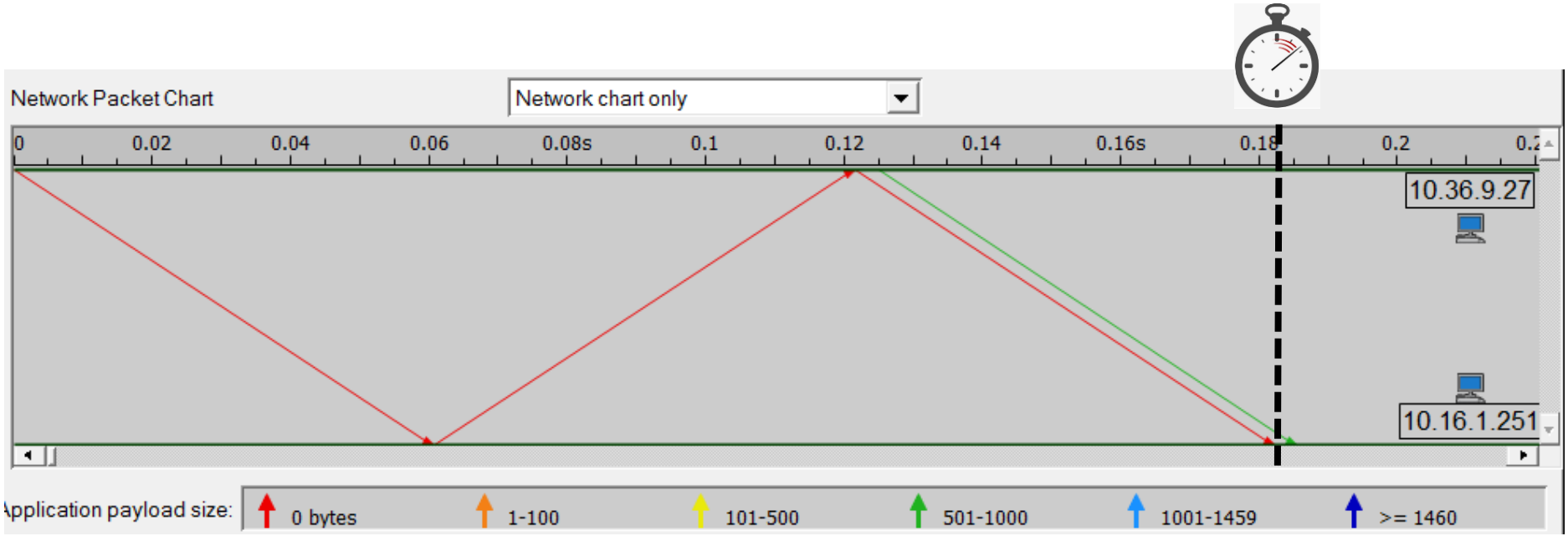
Split Brain – State Discussion



Split Brain – State Discussion



Split Brain – State Discussion

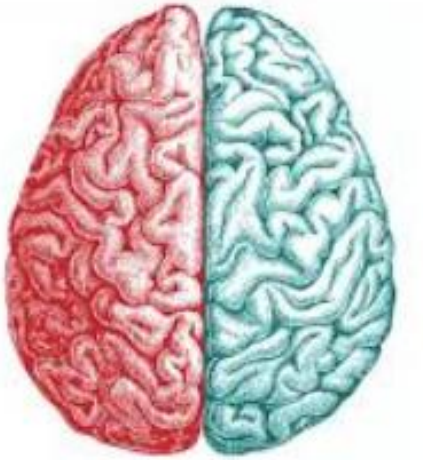


Discussion

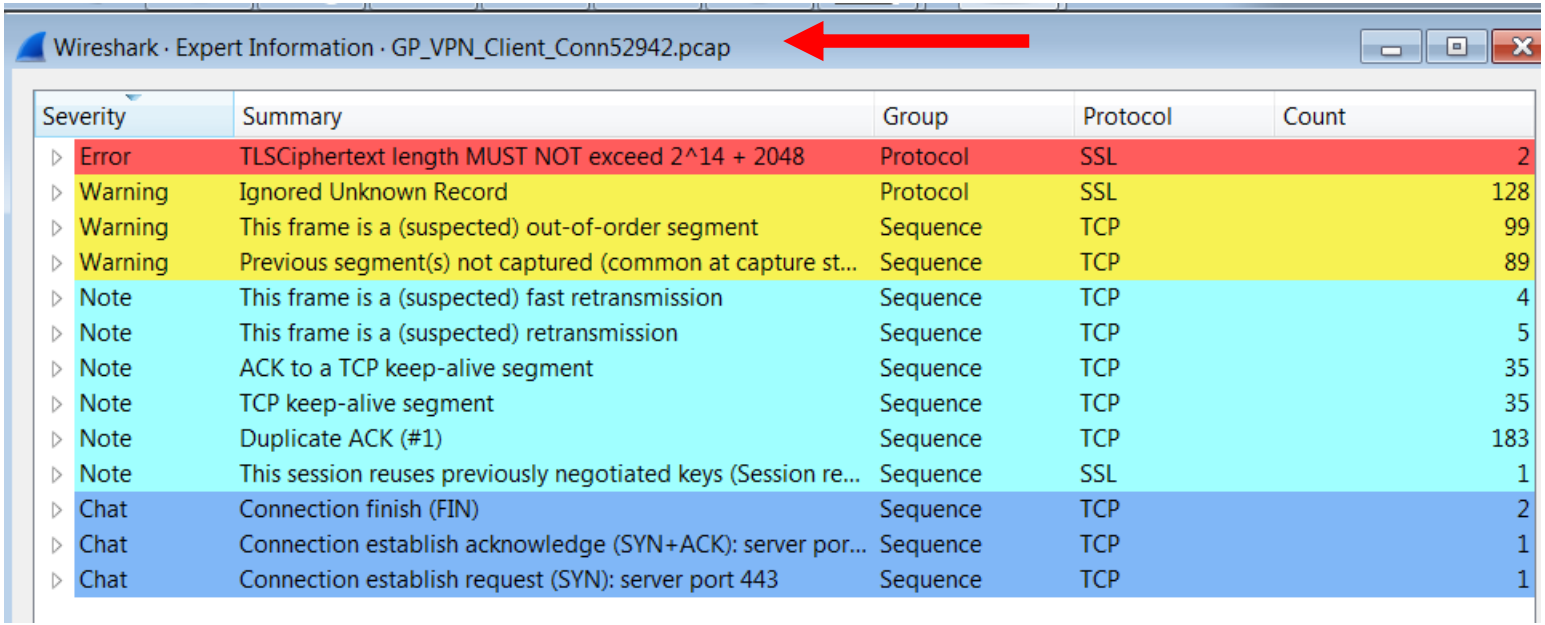
- Do you currently consider the send/receive time of the "other end point" when you do analysis?
- Do you think about each side's TCP state?
- We're just getting started on our journey…

Split Brain Comparisons

- Next we'll compare Wireshark Expert Info for server pcap vs. client pcap

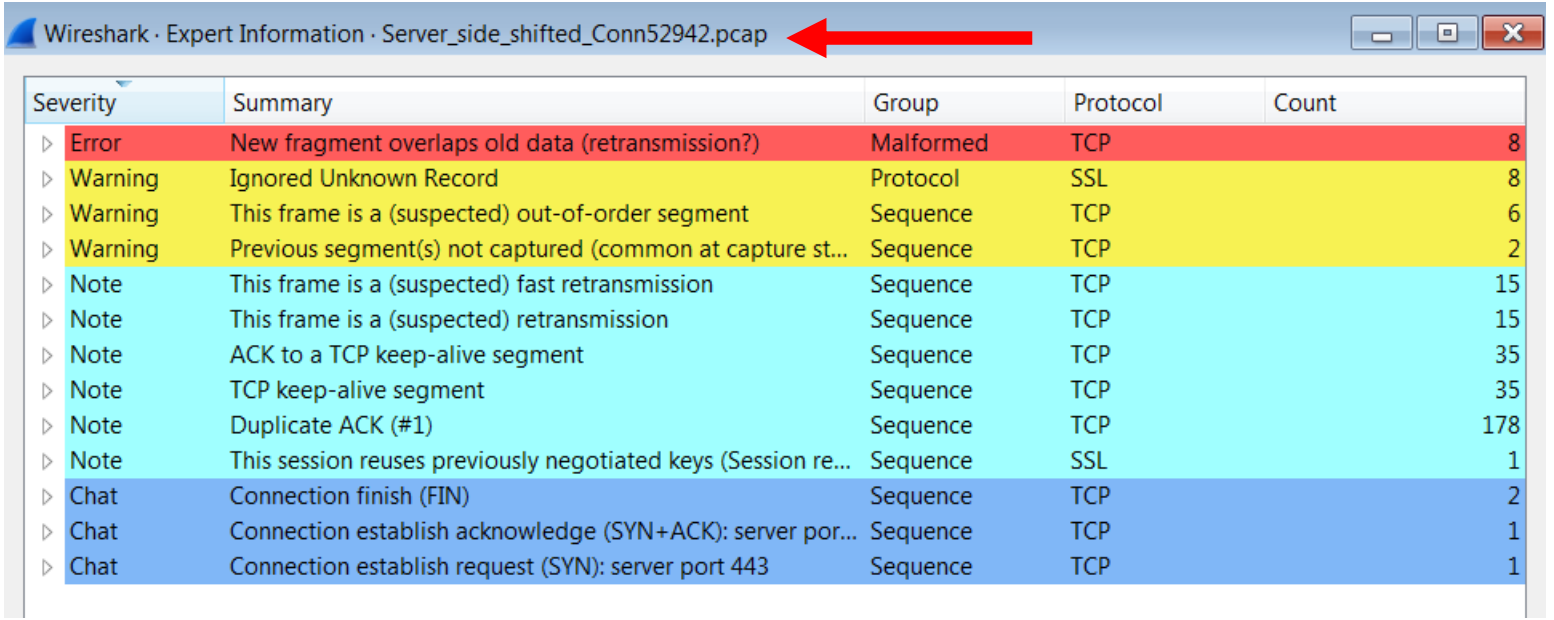


Client Stats Conn52942



Severity	Summary	Group	Protocol	Count
Error	TLSCiphertext length MUST NOT exceed 2^14 + 2048	Protocol	SSL	2
Warning	Ignored Unknown Record	Protocol	SSL	128
Warning	This frame is a (suspected) out-of-order segment	Sequence	TCP	99
Warning	Previous segment(s) not captured (common at capture st...	Sequence	TCP	89
Note	This frame is a (suspected) fast retransmission	Sequence	TCP	4
Note	This frame is a (suspected) retransmission	Sequence	TCP	5
Note	ACK to a TCP keep-alive segment	Sequence	TCP	35
Note	TCP keep-alive segment	Sequence	TCP	35
Note	Duplicate ACK (#1)	Sequence	TCP	183
Note	This session reuses previously negotiated keys (Session re...	Sequence	SSL	1
Chat	Connection finish (FIN)	Sequence	TCP	2
Chat	Connection establish acknowledge (SYN+ACK): server por...	Sequence	TCP	1
Chat	Connection establish request (SYN): server port 443	Sequence	TCP	1

Server Stats Conn52942



Severity	Summary	Group	Protocol	Count
Error	New fragment overlaps old data (retransmission?)	Malformed	TCP	8
Warning	Ignored Unknown Record	Protocol	SSL	8
Warning	This frame is a (suspected) out-of-order segment	Sequence	TCP	6
Warning	Previous segment(s) not captured (common at capture st...	Sequence	TCP	2
Note	This frame is a (suspected) fast retransmission	Sequence	TCP	15
Note	This frame is a (suspected) retransmission	Sequence	TCP	15
Note	ACK to a TCP keep-alive segment	Sequence	TCP	35
Note	TCP keep-alive segment	Sequence	TCP	35
Note	Duplicate ACK (#1)	Sequence	TCP	178
Note	This session reuses previously negotiated keys (Session re...	Sequence	SSL	1
Chat	Connection finish (FIN)	Sequence	TCP	2
Chat	Connection establish acknowledge (SYN+ACK): server por...	Sequence	TCP	1
Chat	Connection establish request (SYN): server port 443	Sequence	TCP	1

Server Left – Client Right

Server Client

Wireshark · Expert Information · Server_side_shifted_Conn52942.pcap

Severity	Summary	Group	Protocol	Count	Count
Error	New fragment overlaps old data (retransmission?)	Malformed	TCP	8	2
Warning	Ignored Unknown Record	Protocol	SSL	8	128
Warning	This frame is a (suspected) out-of-order segment	Sequence	TCP	6	99
Warning	Previous segment(s) not captured (common at capture st...	Sequence	TCP	2	89
Note	This frame is a (suspected) fast retransmission	Sequence	TCP	15	4
Note	This frame is a (suspected) retransmission	Sequence	TCP	15	5
Note	ACK to a TCP keep-alive segment	Sequence	TCP	35	35
Note	TCP keep-alive segment	Sequence	TCP	35	35
Note	Duplicate ACK (#1)	Sequence	TCP	178	183
Note	This session reuses previously negotiated keys (Session re...	Sequence	SSL	1	1
Chat	Connection finish (FIN)	Sequence	TCP	2	2
Chat	Connection establish acknowledge (SYN+ACK): server por...	Sequence	TCP	1	1
Chat	Connection establish request (SYN): server port 443	Sequence	TCP	1	1

Discussion - Retransmissions

					Server	Client
Note	This frame is a (suspected) fast retransmission	Sequence	TCP		15	4
Note	This frame is a (suspected) retransmission	Sequence	TCP		15	5

- Why does server report more retransmissions than client?
- How does Wireshark define a “fast retransmission”?

Wireshark Help

TCP Fast Retransmission

Set when all of the following are true:

- This is not a keepalive packet.
- In the forward direction, the segment size is greater than zero or the SYN or FIN is set.
- The next expected sequence number is greater than the current sequence number.
- We have more than two duplicate ACKs in the reverse direction.
- The current sequence number equals the next expected acknowledgement number.
- We saw the last acknowledgement less than 20ms ago.

Supersedes "Out-Of-Order", "Spurious Retransmission", and "Retransmission".

Discussion – OOS & Drops

				Server	Client
Warning	This frame is a (suspected) out-of-order segment	Sequence	TCP	6	99
Warning	Previous segment(s) not captured (common at capture st...	Sequence	TCP	2	89

- Why does client report more OOS and “previous segment(s) not captured” than server?
- Why does server report *any* OOS?

Wireshark Help

TCP Out-Of-Order

Set when all of the following are true:

- This is not a keepalive packet.
- In the forward direction, the segment length is greater than zero or the SYN or FIN is set.
- The next expected sequence number is greater than the current sequence number.
- The next expected sequence number and the next sequence number differ.
- The last segment arrived within the calculated RTT (3ms by default).

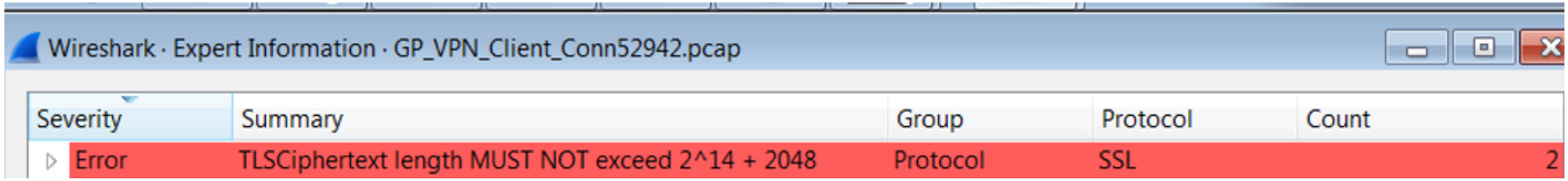
Supersedes “Spurious Retransmission” and “Retransmission”.

Discussion – Unknown Records

				Server	Client
Warning	Ignored Unknown Record	Protocol	SSL	8	128

- Why does client report more SSL unknown records than server?

Discussion – Length Error

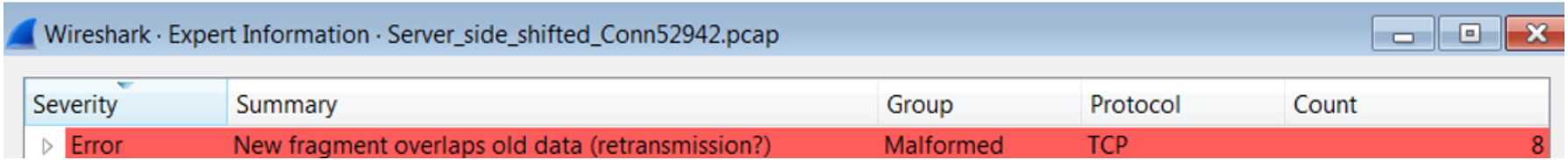


- Why does client report Ciphertext length error and Server does not?

Wireshark Update

- Wireshark 2.6.3 was used for this course
- The SSL/TLS Expert Info was updated in 2.6.4 and may treat these captures differently

Discussion
Fragment Overlap

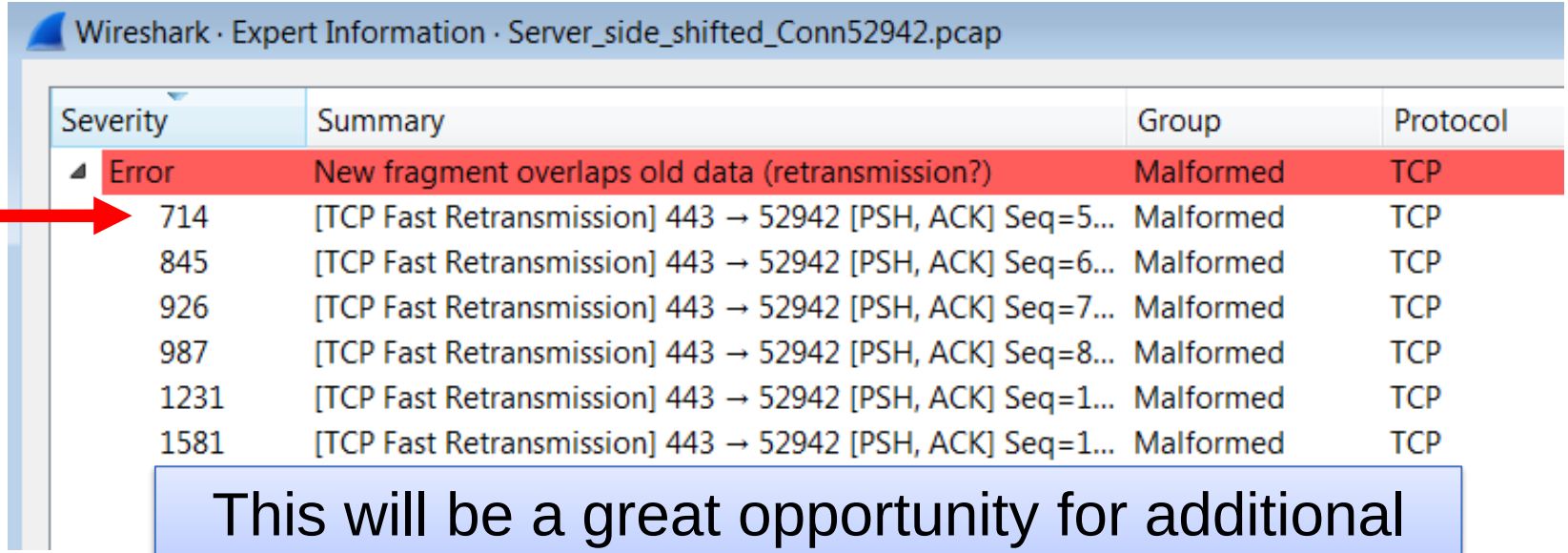


- Why does Server report Fragment Overlap error and Client does not?
- Should this really be Severity == Error?

Discussion

- Do these differences make sense?
- Additional thoughts?
- Comments?

Let's drill into one of these...



This will be a great opportunity for additional Split Brain comparisons

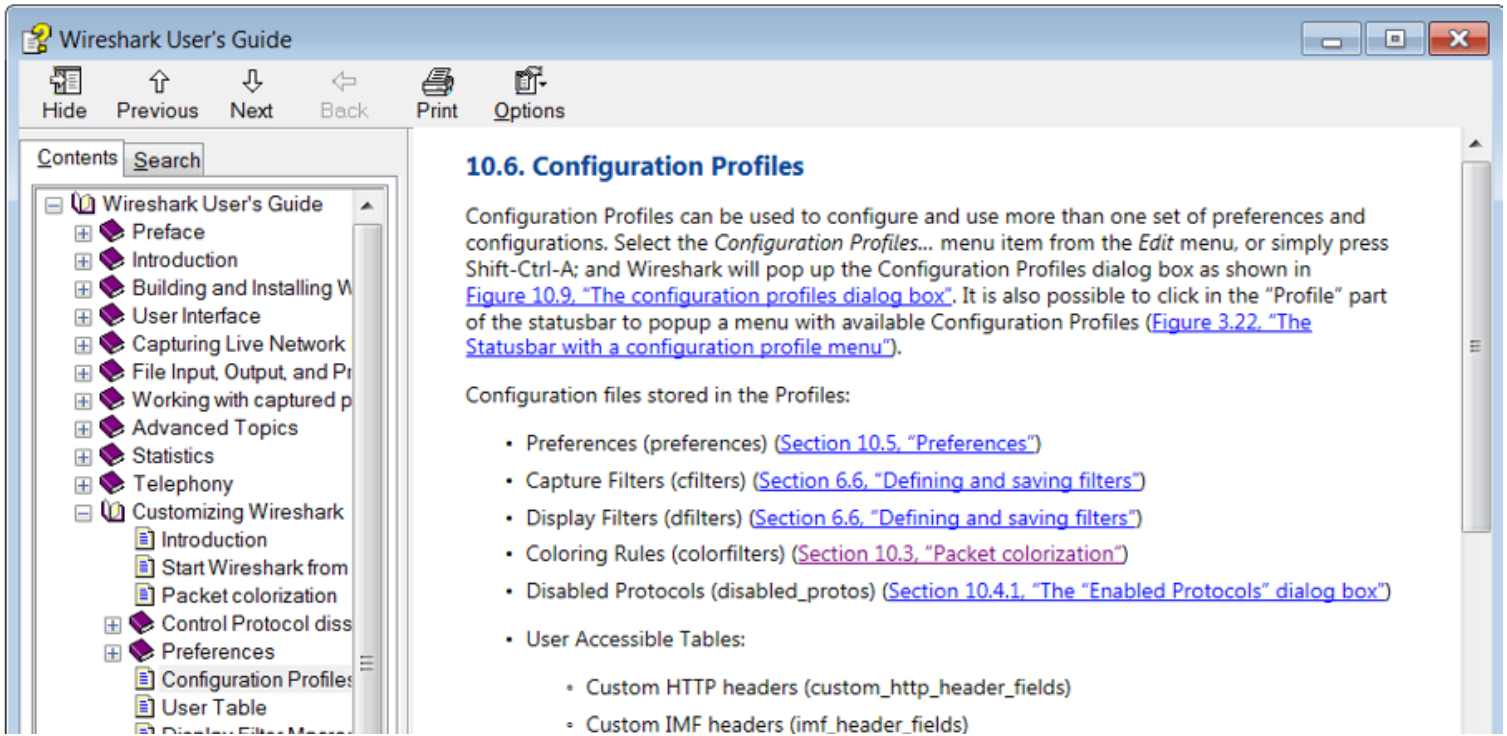
Split Brain Comparisons

- Drill down into Fragment Overlap Details

Pop Quiz

- What's one of the best things about Wireshark?
- OK…, besides the Developers?
- Totally flexible columns, views, and profiles!!

Configuration Profile Help

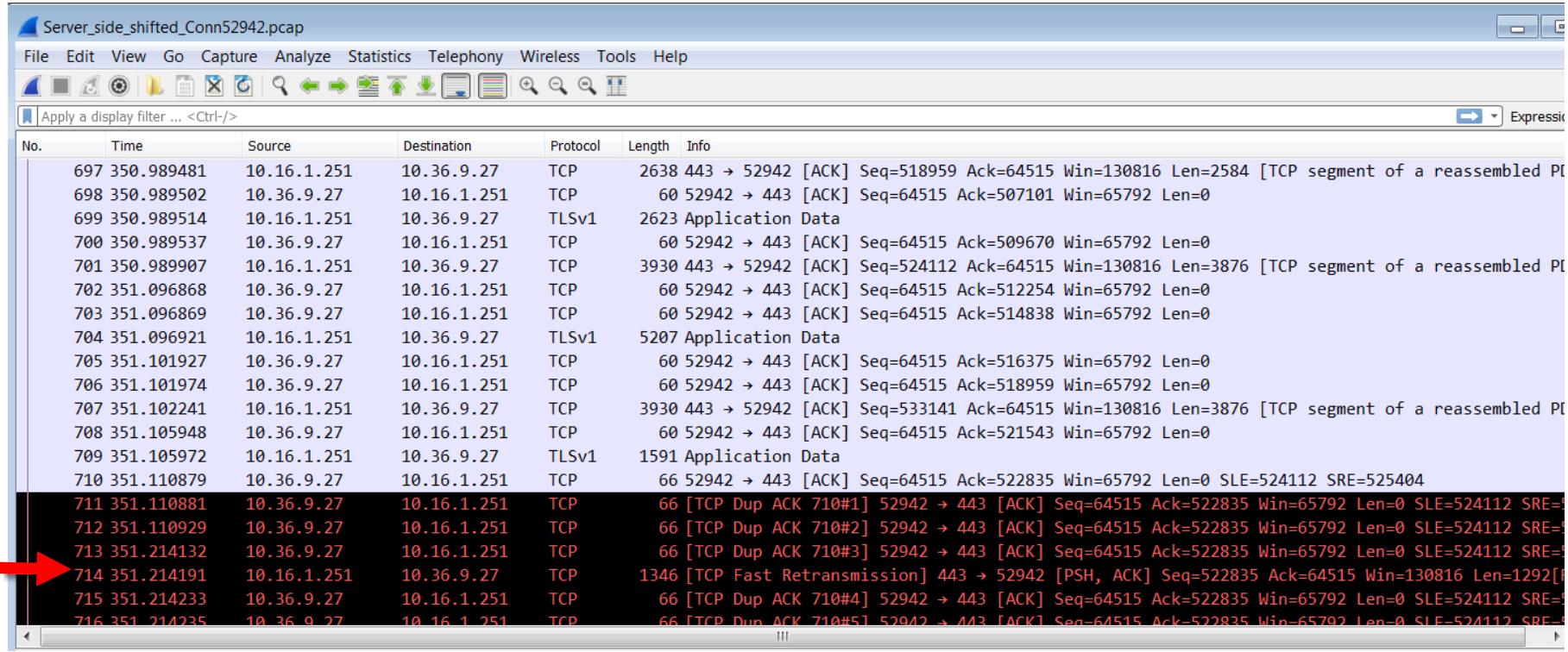


Wireshark Profile Resources

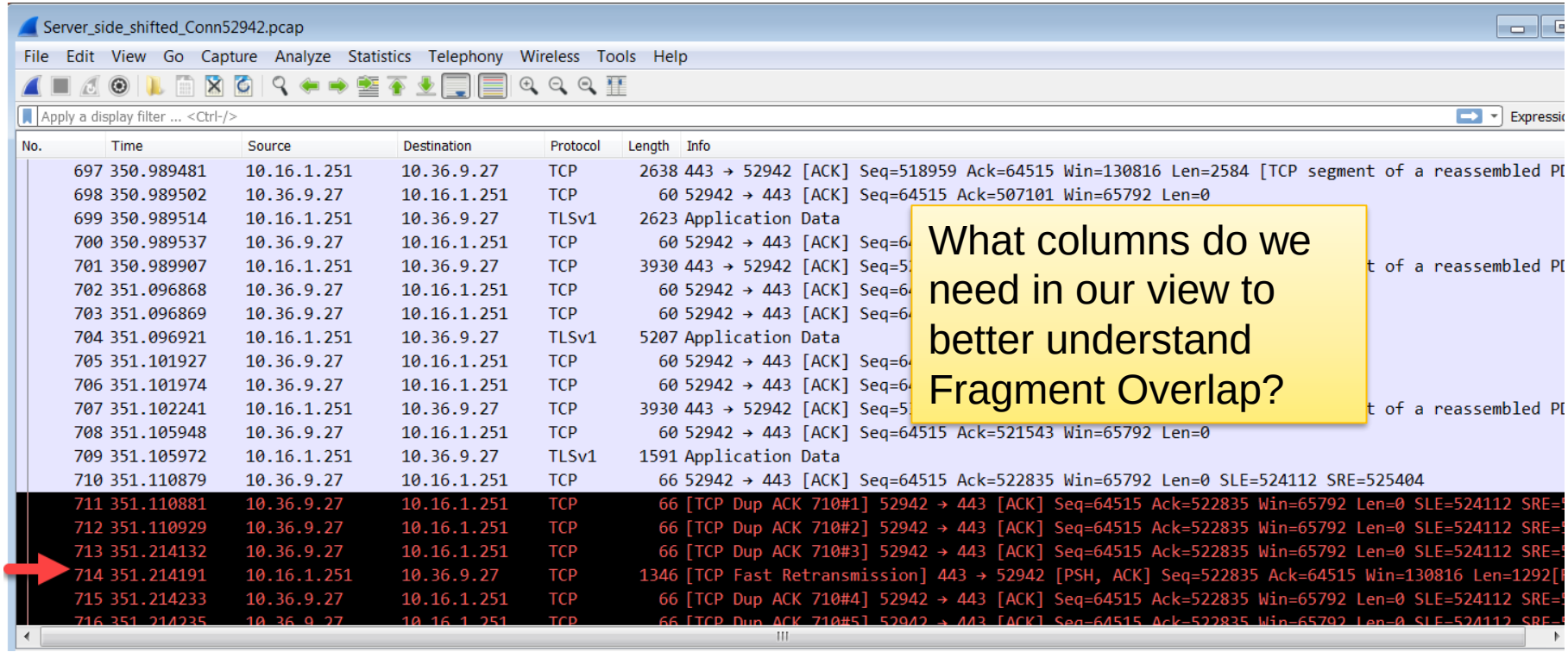
- SB-SACK
- SB-IP ID
- SB-Seq Analysis

- (will be posted on the Sharkfest retrospective page with the capture files)

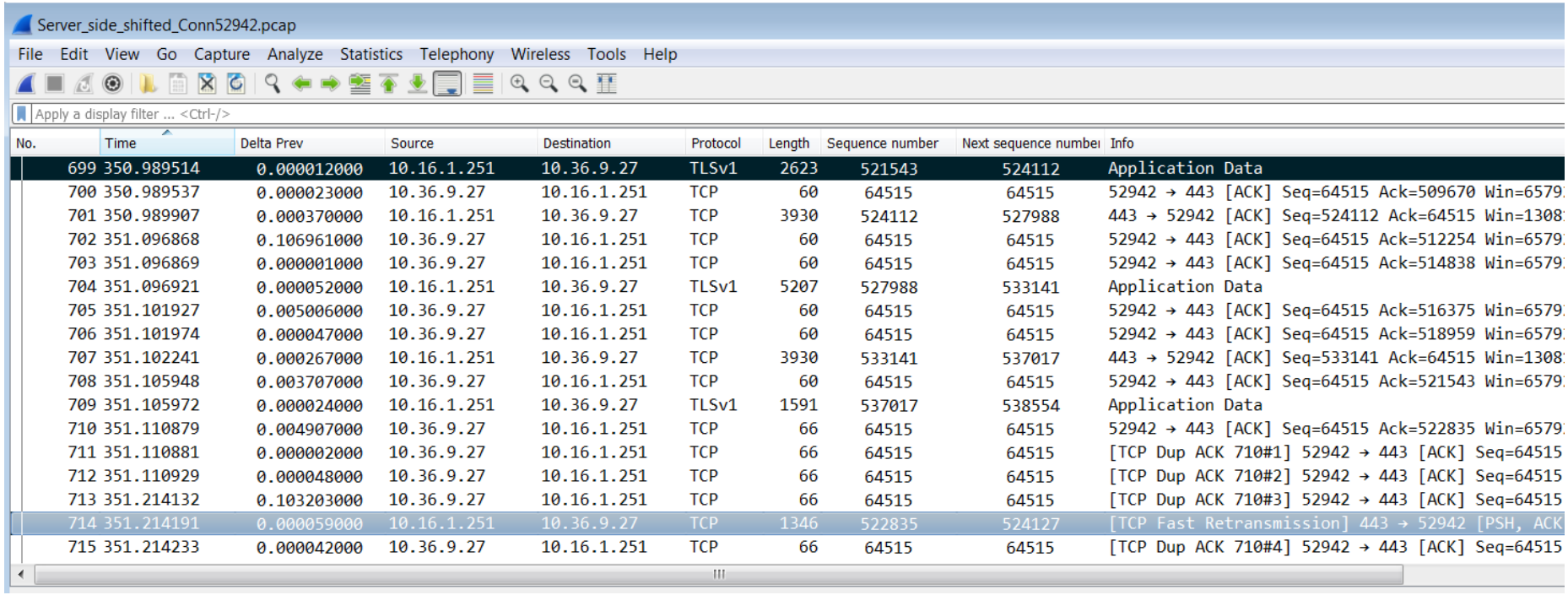
View with Classic Profile



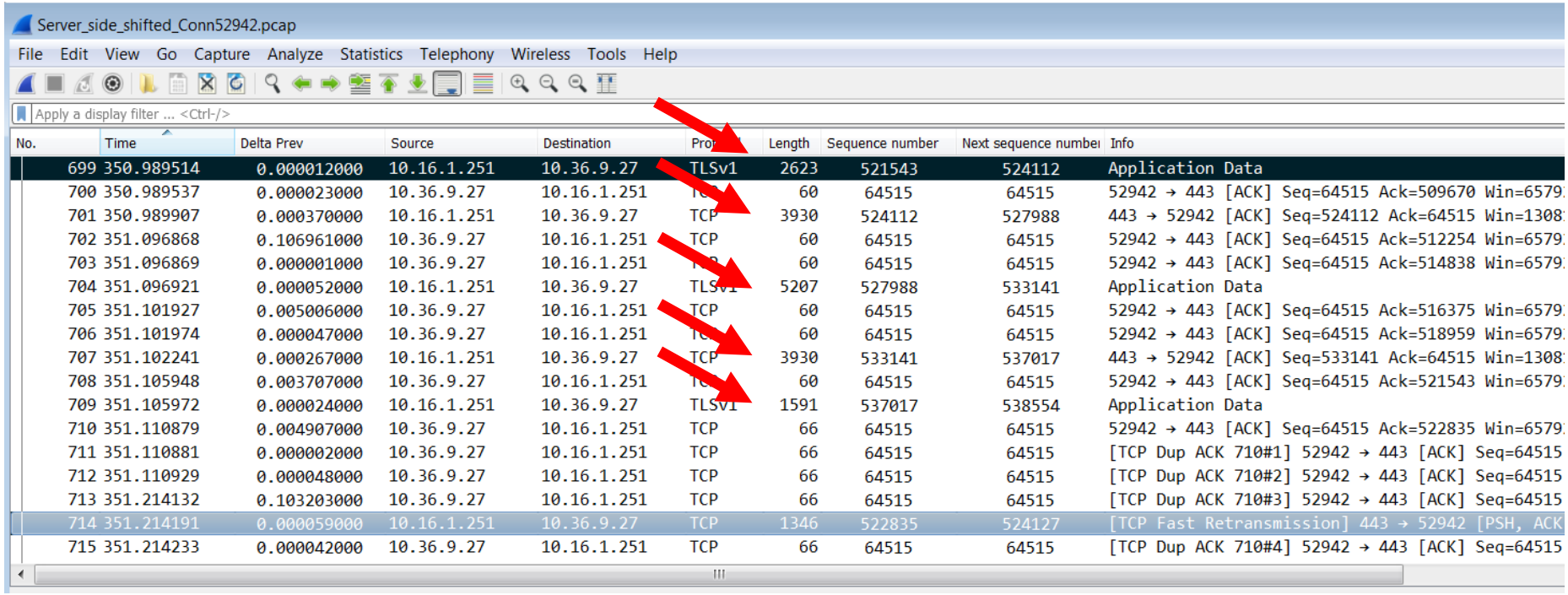
View with Classic Profile



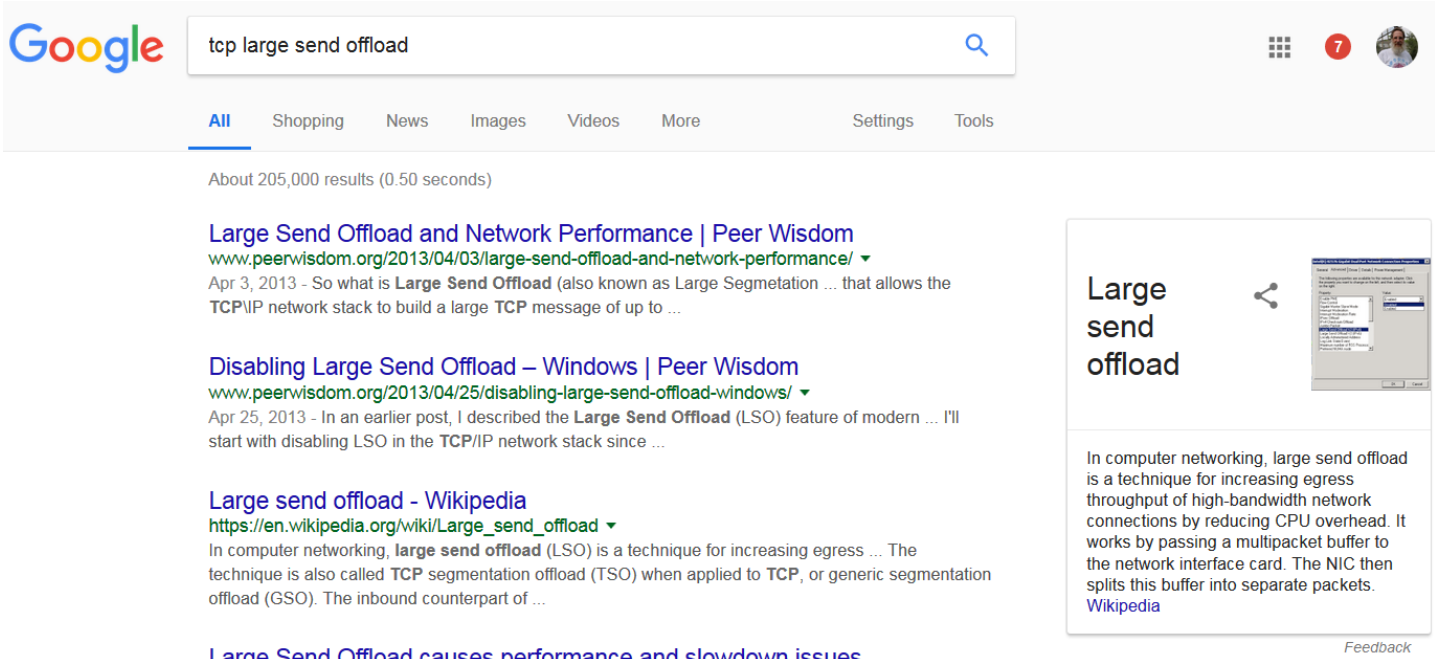
Profile SB-Seq Analysis



What are these “jumbo” packets?



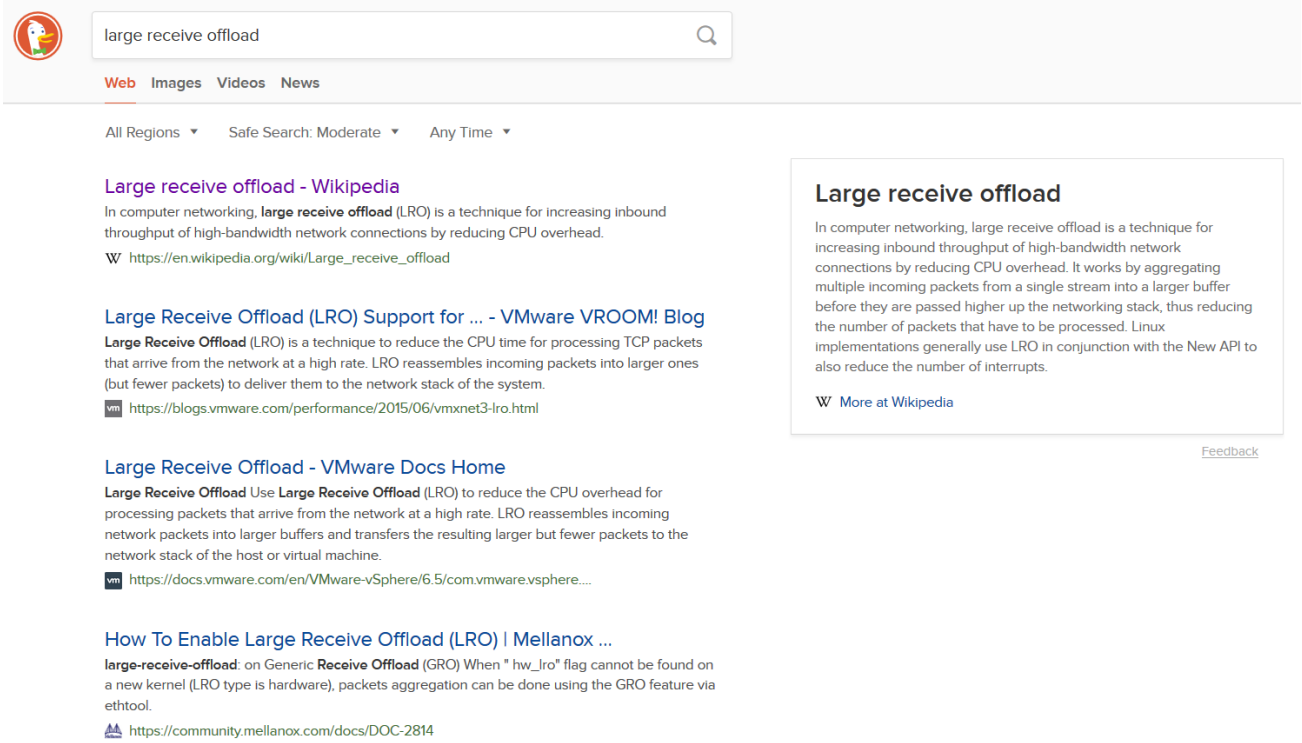
TCP Large Send Offload



Discussion

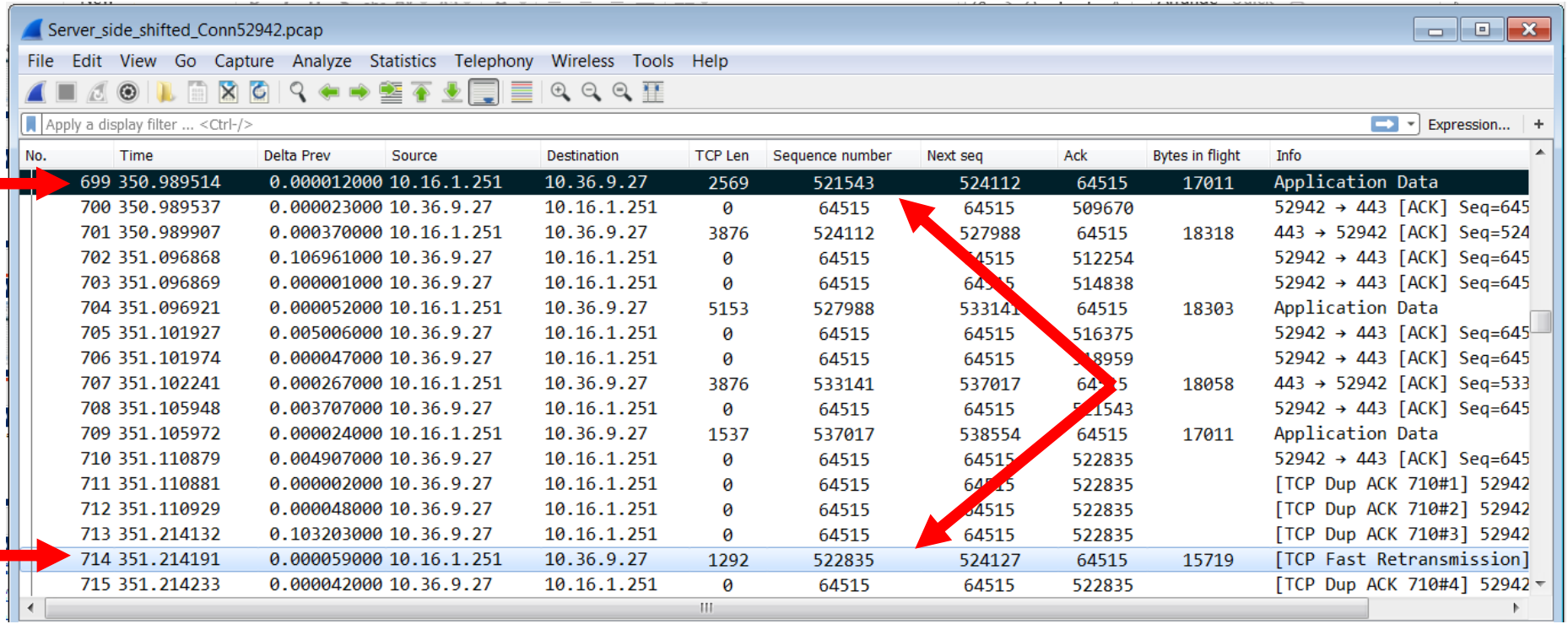
- Where does pcap intercept packets?
- What is the packet's journey after pcap capture?
- How is this different on a VM?
- How about a host based on blade chassis technology (aka Blade Server)?

TCP Large Receive Offload

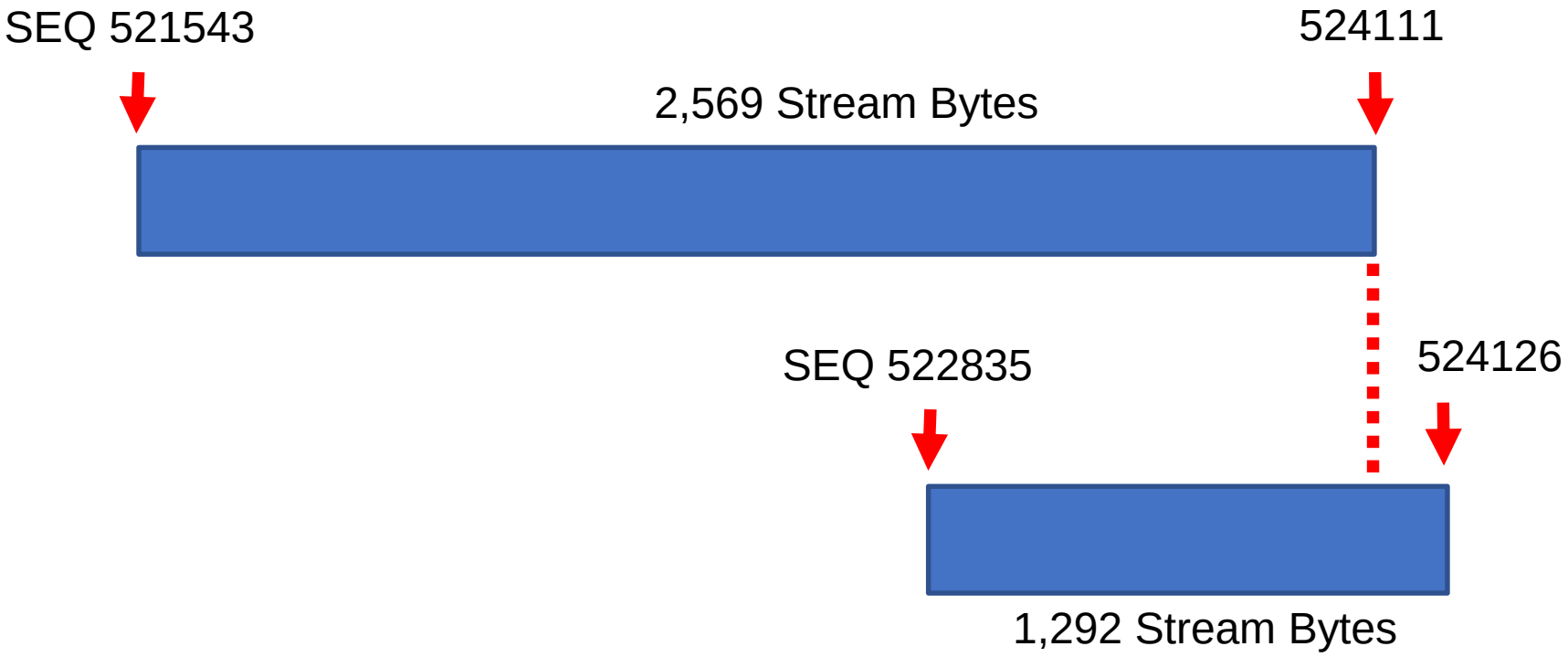


Back to Fragment Overlap

Here is the overlap



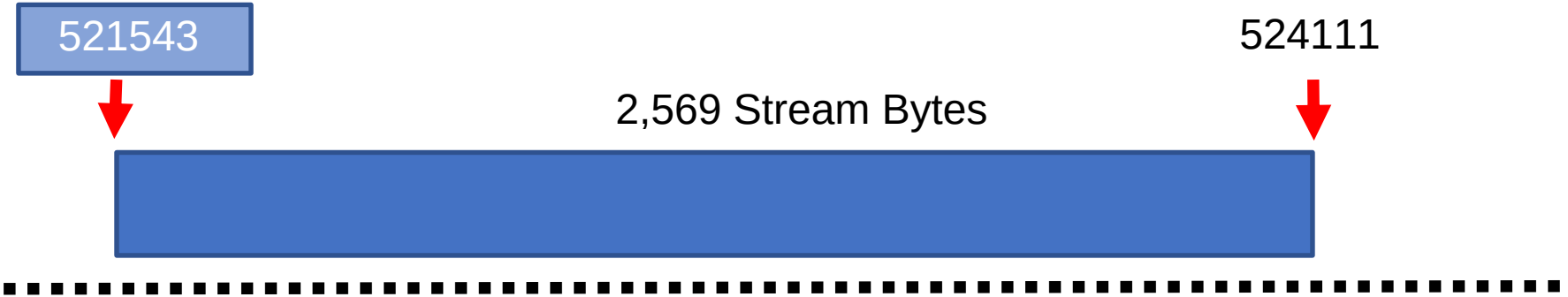
Server Side Stream Visual



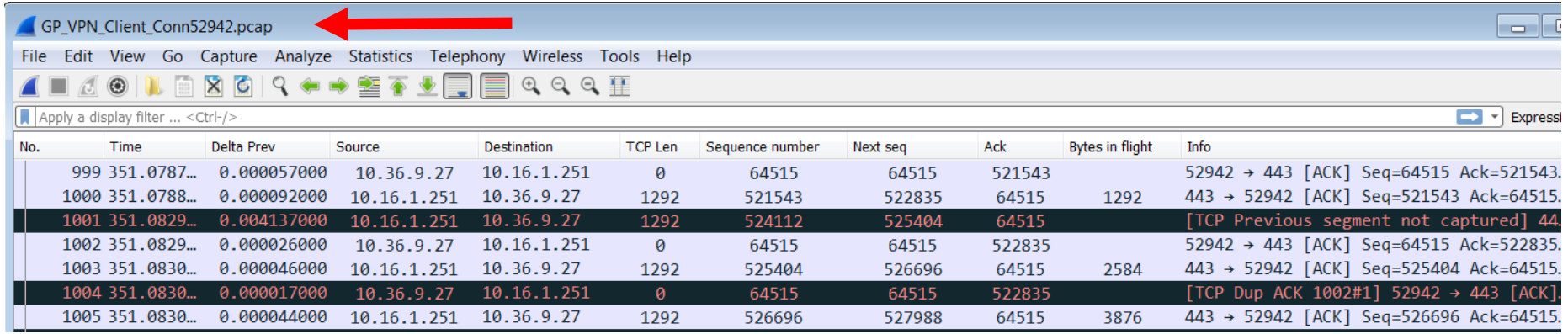
Discussion

- At first, it might seem odd that the retransmitted segment does not “line up” with original segment
- We can be reasonably confident this is result of Large Send Offload
- Let’s jump to client capture to confirm…

Anchor on SEQ==521543



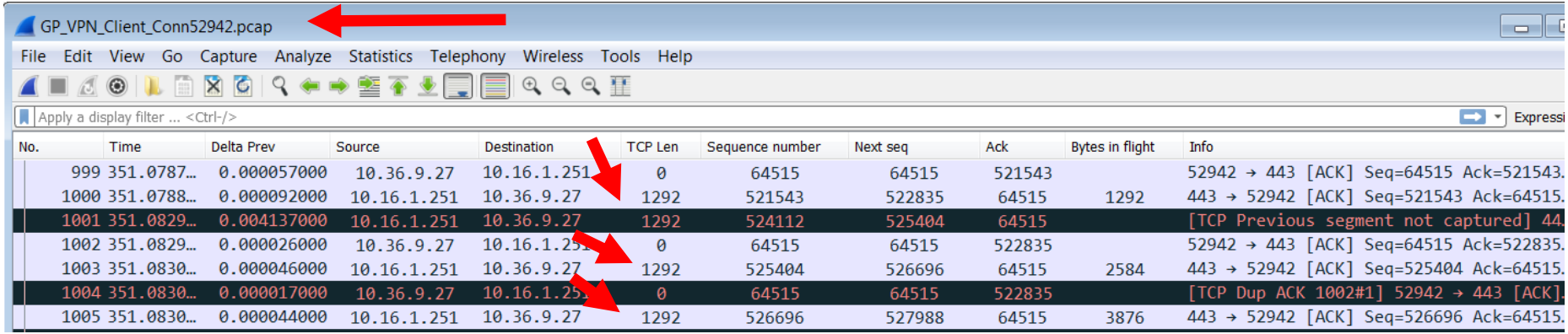
Several Observations



No.	Time	Delta Prev	Source	Destination	TCP Len	Sequence number	Next seq	Ack	Bytes in flight	Info
999	351.0787…	0.000057000	10.36.9.27	10.16.1.251	0	64515	64515	521543		52942 → 443 [ACK] Seq=64515 Ack=521543.
1000	351.0788…	0.000092000	10.16.1.251	10.36.9.27	1292	521543	522835	64515	1292	443 → 52942 [ACK] Seq=521543 Ack=64515.
1001	351.0829…	0.004137000	10.16.1.251	10.36.9.27	1292	524112	525404	64515		[TCP Previous segment not captured] 44.
1002	351.0829…	0.000026000	10.36.9.27	10.16.1.251	0	64515	64515	522835		52942 → 443 [ACK] Seq=64515 Ack=522835.
1003	351.0830…	0.000046000	10.16.1.251	10.36.9.27	1292	525404	526696	64515	2584	443 → 52942 [ACK] Seq=525404 Ack=64515.
1004	351.0830…	0.000017000	10.36.9.27	10.16.1.251	0	64515	64515	522835		[TCP Dup ACK 1002#1] 52942 → 443 [ACK].
1005	351.0830…	0.000044000	10.16.1.251	10.36.9.27	1292	526696	527988	64515	3876	443 → 52942 [ACK] Seq=526696 Ack=64515.

- Note file name in the title bar

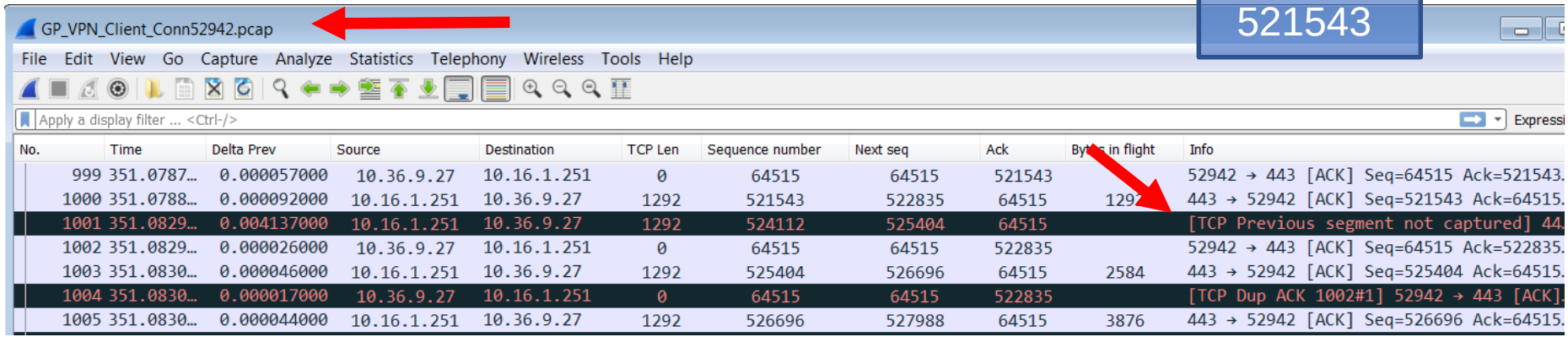
Segment Size Differences



No.	Time	Delta Prev	Source	Destination	TCP Len	Sequence number	Next seq	Ack	Bytes in flight	Info
999	351.0787…	0.000057000	10.36.9.27	10.16.1.251	0	64515	64515	521543		52942 → 443 [ACK] Seq=64515 Ack=521543.
1000	351.0788…	0.000092000	10.16.1.251	10.36.9.27	1292	521543	522835	64515	1292	443 → 52942 [ACK] Seq=521543 Ack=64515.
1001	351.0829…	0.004137000	10.16.1.251	10.36.9.27	1292	524112	525404	64515		[TCP Previous segment not captured] 44.
1002	351.0829…	0.000026000	10.36.9.27	10.16.1.251	0	64515	64515	522835		52942 → 443 [ACK] Seq=64515 Ack=522835.
1003	351.0830…	0.000046000	10.16.1.251	10.36.9.27	1292	525404	526696	64515	2584	443 → 52942 [ACK] Seq=525404 Ack=64515.
1004	351.0830…	0.000017000	10.36.9.27	10.16.1.251	0	64515	64515	522835		[TCP Dup ACK 1002#1] 52942 → 443 [ACK].
1005	351.0830…	0.000044000	10.16.1.251	10.36.9.27	1292	526696	527988	64515	3876	443 → 52942 [ACK] Seq=526696 Ack=64515.

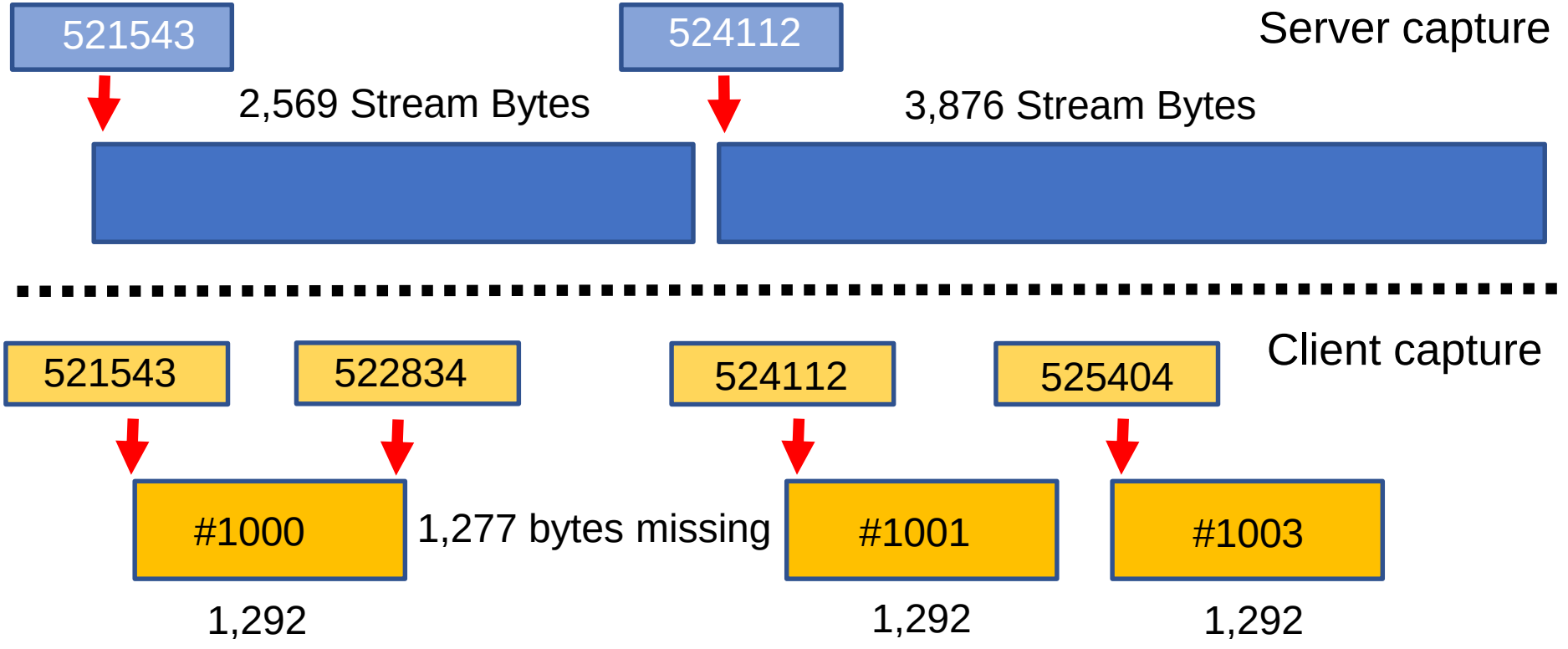
- Confirmed: Large Send Offload configured on Server
- Looks like the server NIC is handling segmentation

Dropped or OOS?

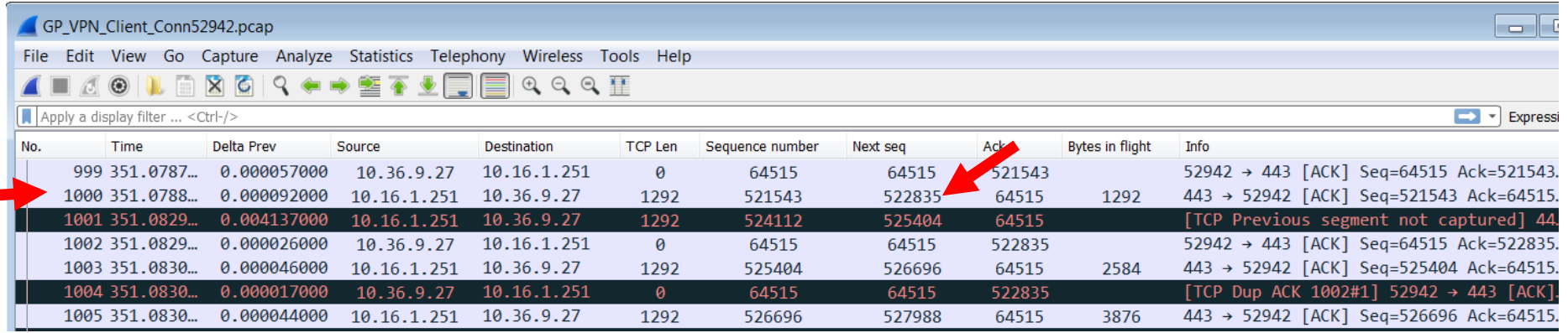


- If we freeze time right here, we can't be sure if it's just OOS or really a dropped packet
- We have to examine what comes next…

Anchor on SEQ==521543



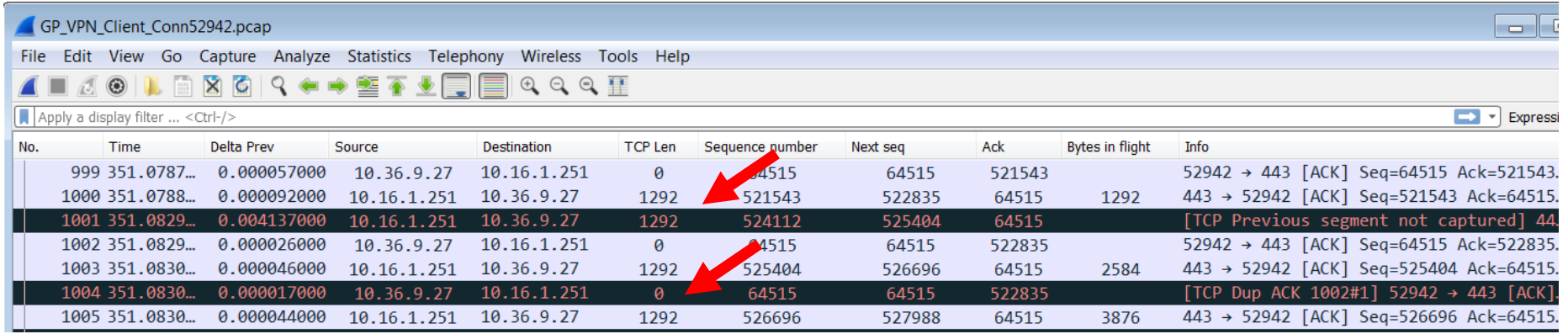
Other Clues



No.	Time	Delta Prev	Source	Destination	TCP Len	Sequence number	Next seq	Ack	Bytes in flight	Info
999	351.0787…	0.000057000	10.36.9.27	10.16.1.251	0	64515	64515	521543		52942 → 443 [ACK] Seq=64515 Ack=521543.
1000	351.0788…	0.000092000	10.16.1.251	10.36.9.27	1292	521543	522835	64515	1292	443 → 52942 [ACK] Seq=521543 Ack=64515.
1001	351.0829…	0.004137000	10.16.1.251	10.36.9.27	1292	524112	525404	64515		[TCP Previous segment not captured] 44.
1002	351.0829…	0.000026000	10.36.9.27	10.16.1.251	0	64515	64515	522835		52942 → 443 [ACK] Seq=64515 Ack=522835.
1003	351.0830…	0.000046000	10.16.1.251	10.36.9.27	1292	525404	526696	64515	2584	443 → 52942 [ACK] Seq=525404 Ack=64515.
1004	351.0830…	0.000017000	10.36.9.27	10.16.1.251	0	64515	64515	522835		[TCP Dup ACK 1002#1] 52942 → 443 [ACK].
1005	351.0830…	0.000044000	10.16.1.251	10.36.9.27	1292	526696	527988	64515	3876	443 → 52942 [ACK] Seq=526696 Ack=64515.

- Next seq after #1000 should have been 522835
- Wait! Isn't this the segment that was retransmitted by server? (yes)

Other Clues

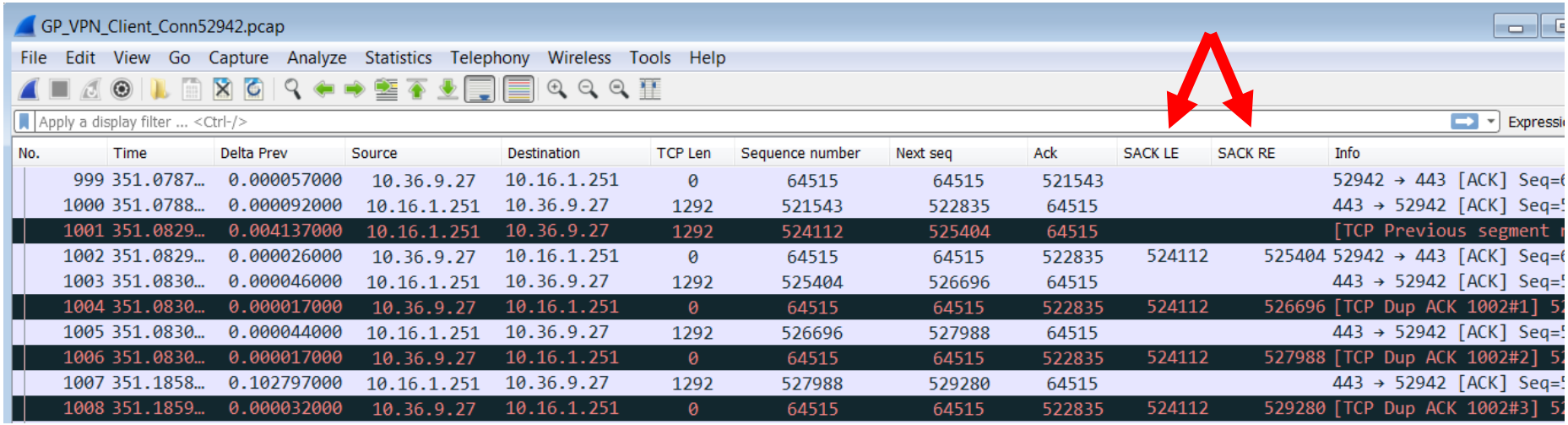


- Notice the change in ACK behavior
- Client ACKs every other packet then starts to ACK every packet
- Why is this?

Profile Power

- Let’s flip the view a little so we can quickly see SACK fields in the decode summary

Profile: SB-SACK



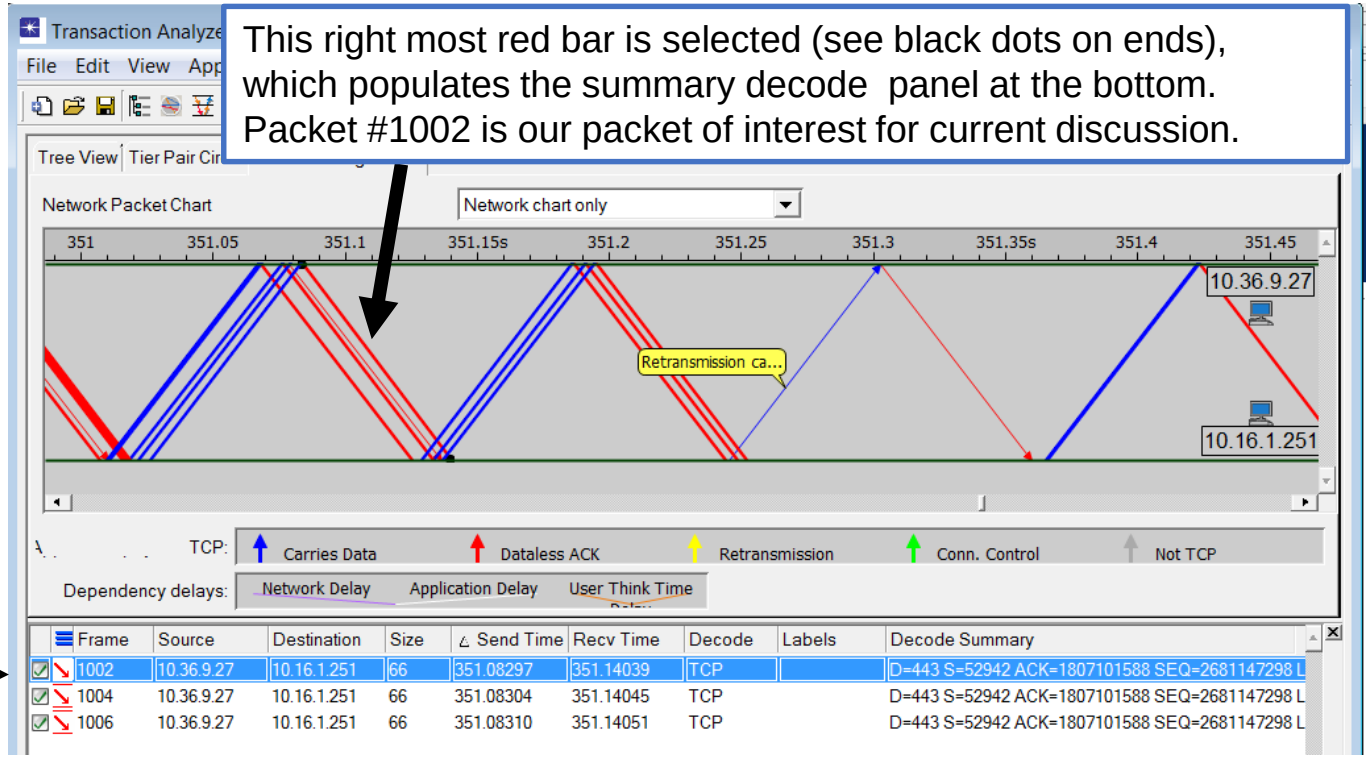
No.	Time	Delta Prev	Source	Destination	TCP Len	Sequence number	Next seq	Ack	SACK LE	SACK RE	Info
999	351.0787…	0.000057000	10.36.9.27	10.16.1.251	0	64515	64515	521543			52942 → 443 [ACK] Seq=6
1000	351.0788…	0.000092000	10.16.1.251	10.36.9.27	1292	521543	522835	64515			443 → 52942 [ACK] Seq=5
1001	351.0829…	0.004137000	10.16.1.251	10.36.9.27	1292	524112	525404	64515			[TCP Previous segment n
1002	351.0829…	0.000026000	10.36.9.27	10.16.1.251	0	64515	64515	522835	524112	525404	52942 → 443 [ACK] Seq=6
1003	351.0830…	0.000046000	10.16.1.251	10.36.9.27	1292	525404	526696	64515			443 → 52942 [ACK] Seq=5
1004	351.0830…	0.000017000	10.36.9.27	10.16.1.251	0	64515	64515	522835	524112	526696	[TCP Dup ACK 1002#1] 52
1005	351.0830…	0.000044000	10.16.1.251	10.36.9.27	1292	526696	527988	64515			443 → 52942 [ACK] Seq=5
1006	351.0830…	0.000017000	10.36.9.27	10.16.1.251	0	64515	64515	522835	524112	527988	[TCP Dup ACK 1002#2] 52
1007	351.1858…	0.102797000	10.16.1.251	10.36.9.27	1292	527988	529280	64515			443 → 52942 [ACK] Seq=5
1008	351.1859…	0.000032000	10.36.9.27	10.16.1.251	0	64515	64515	522835	524112	529280	[TCP Dup ACK 1002#3] 52

- Client “SACKs” the new segments, but continues to report - I’m missing 522835

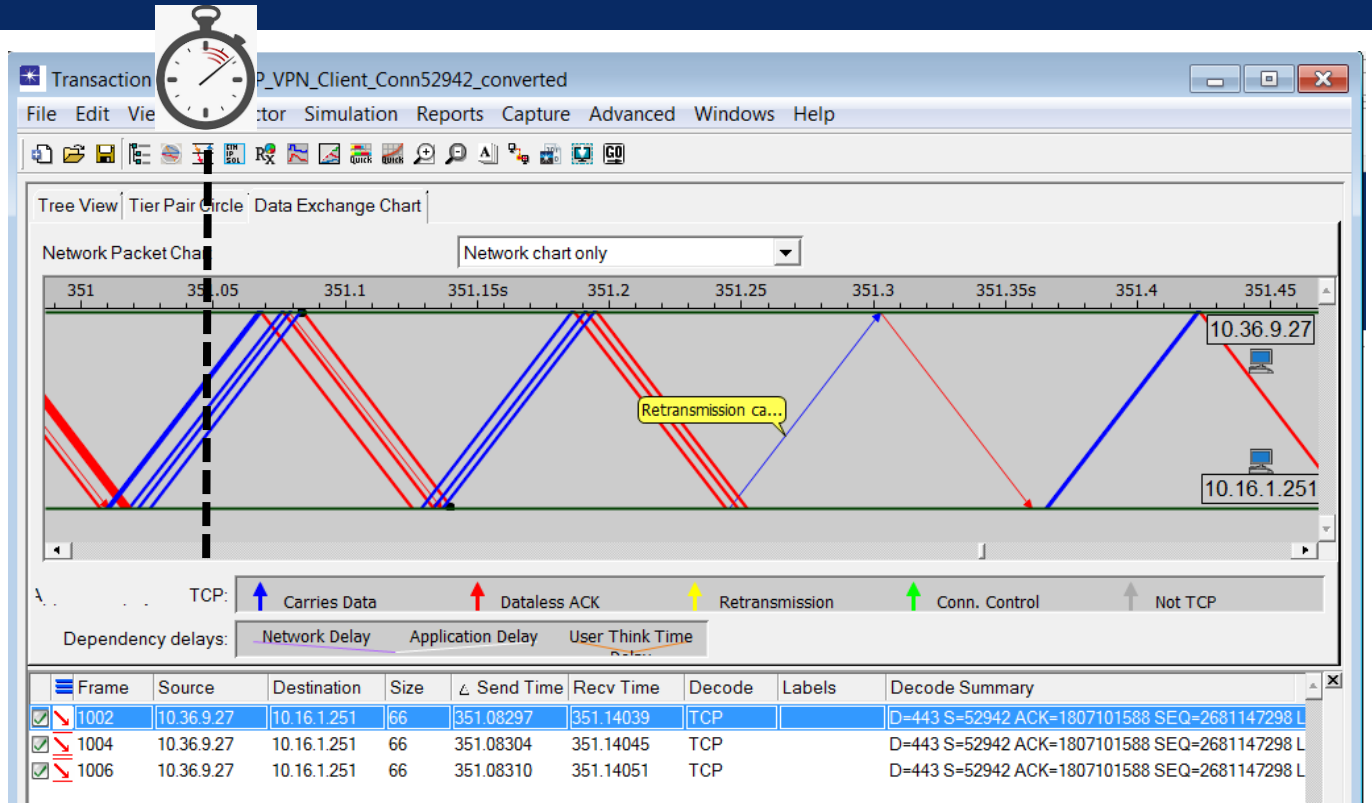
Discussion

- We can see Client is reporting a missing segment
- Yet, why does server continue to send segments other than the one requested?
- Visualization really helps with this…

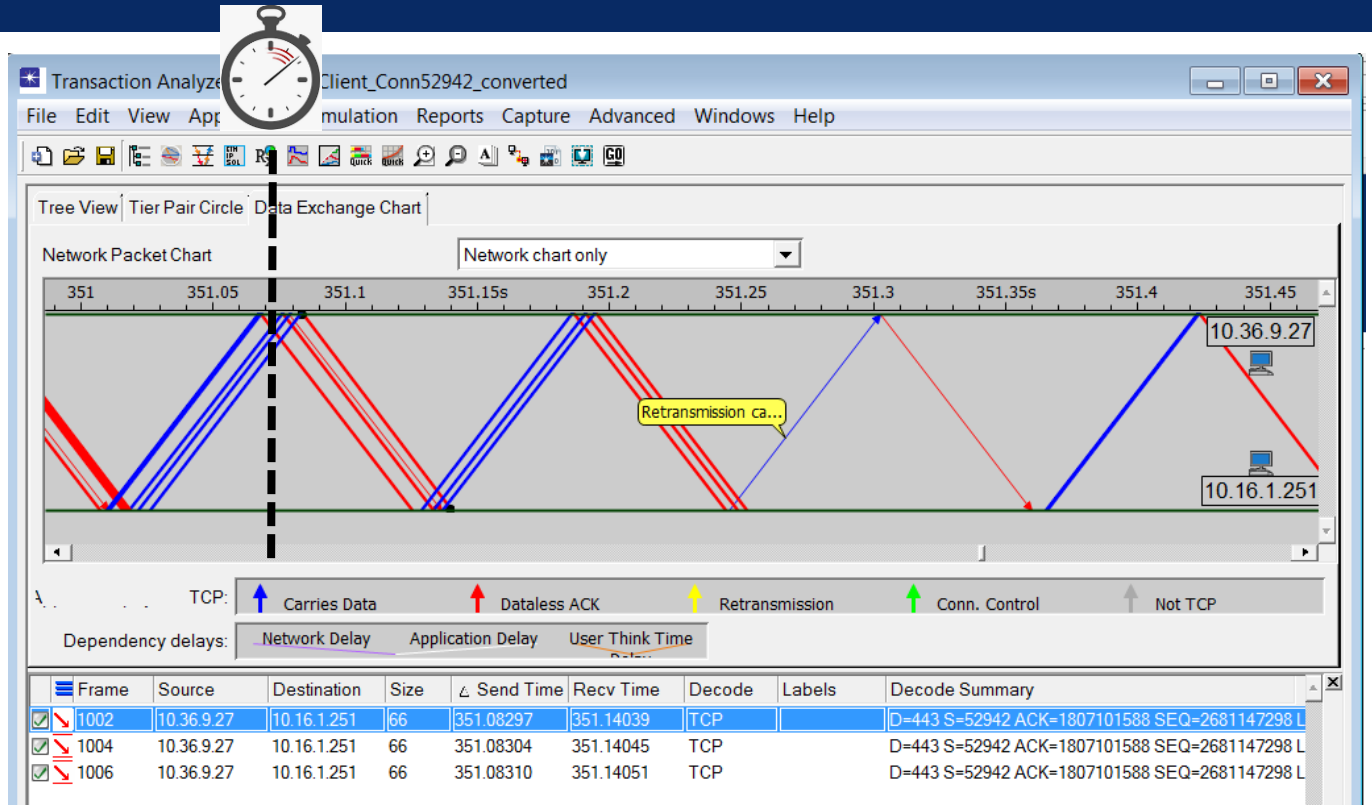
End Point State Review



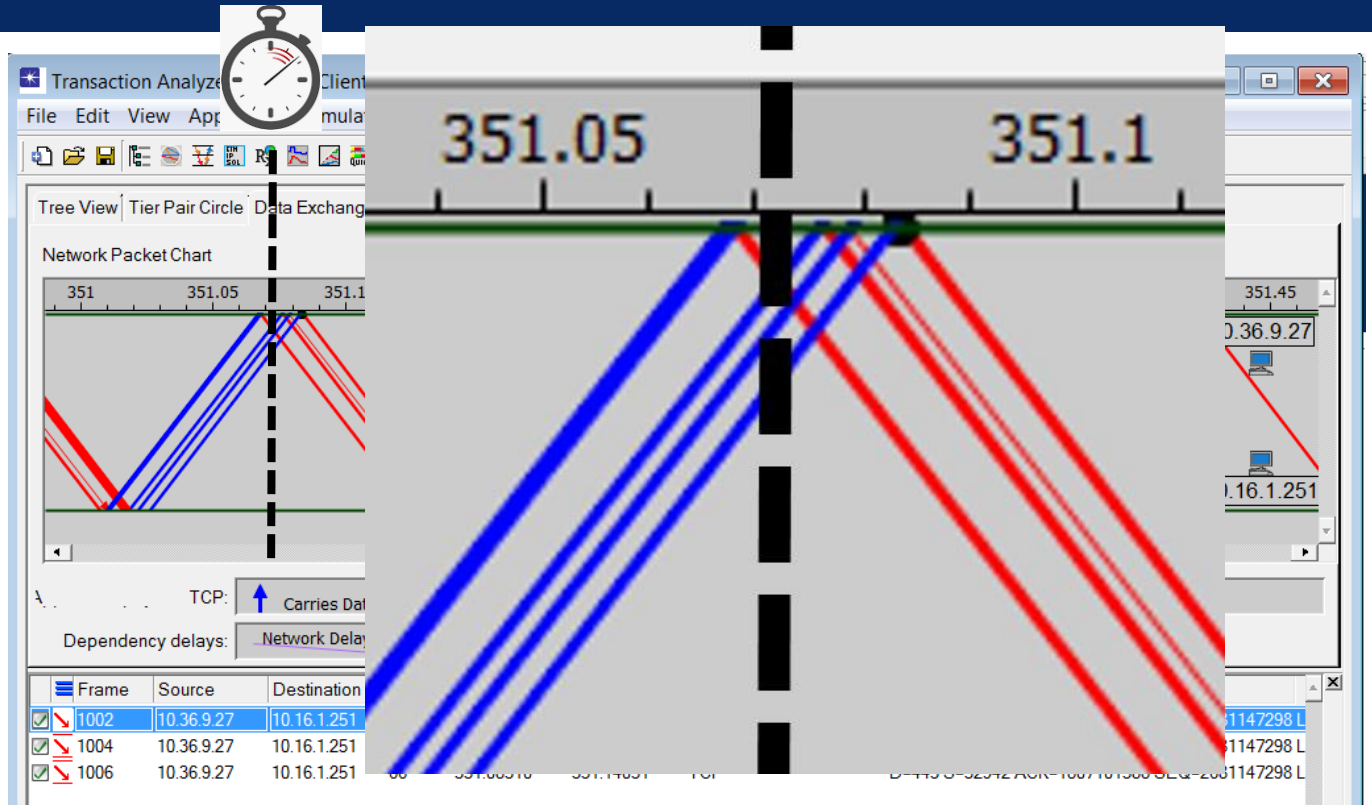
Discuss state of each end point



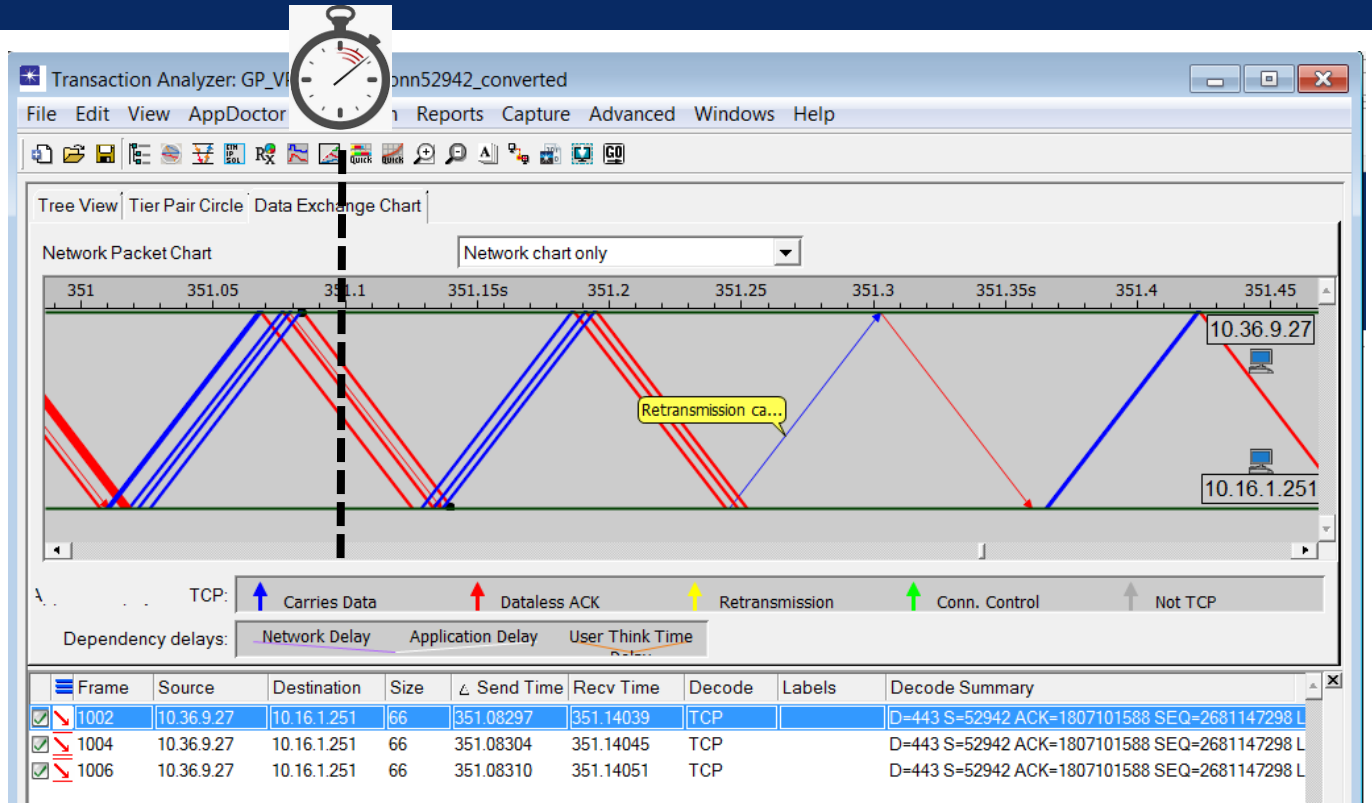
Discuss state of each end point



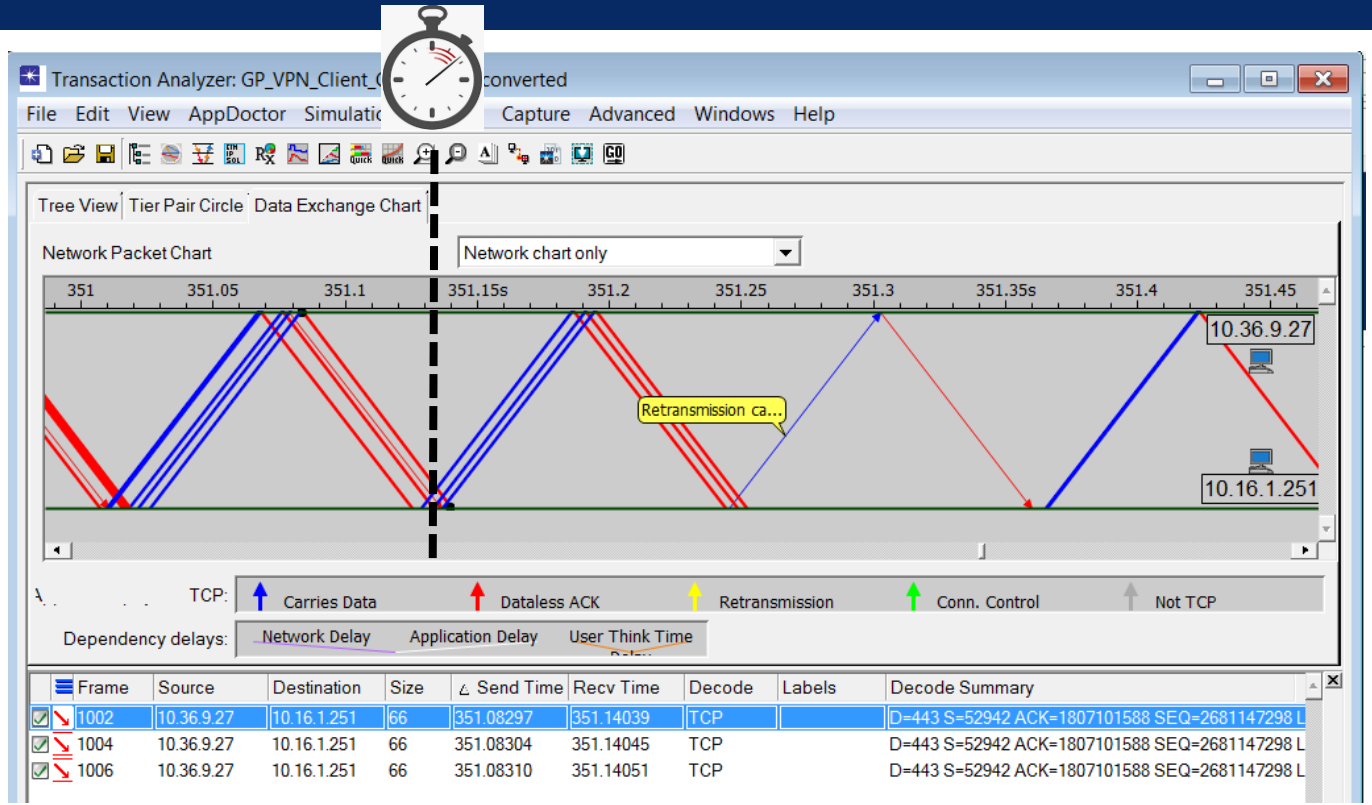
Zoom-in a little…



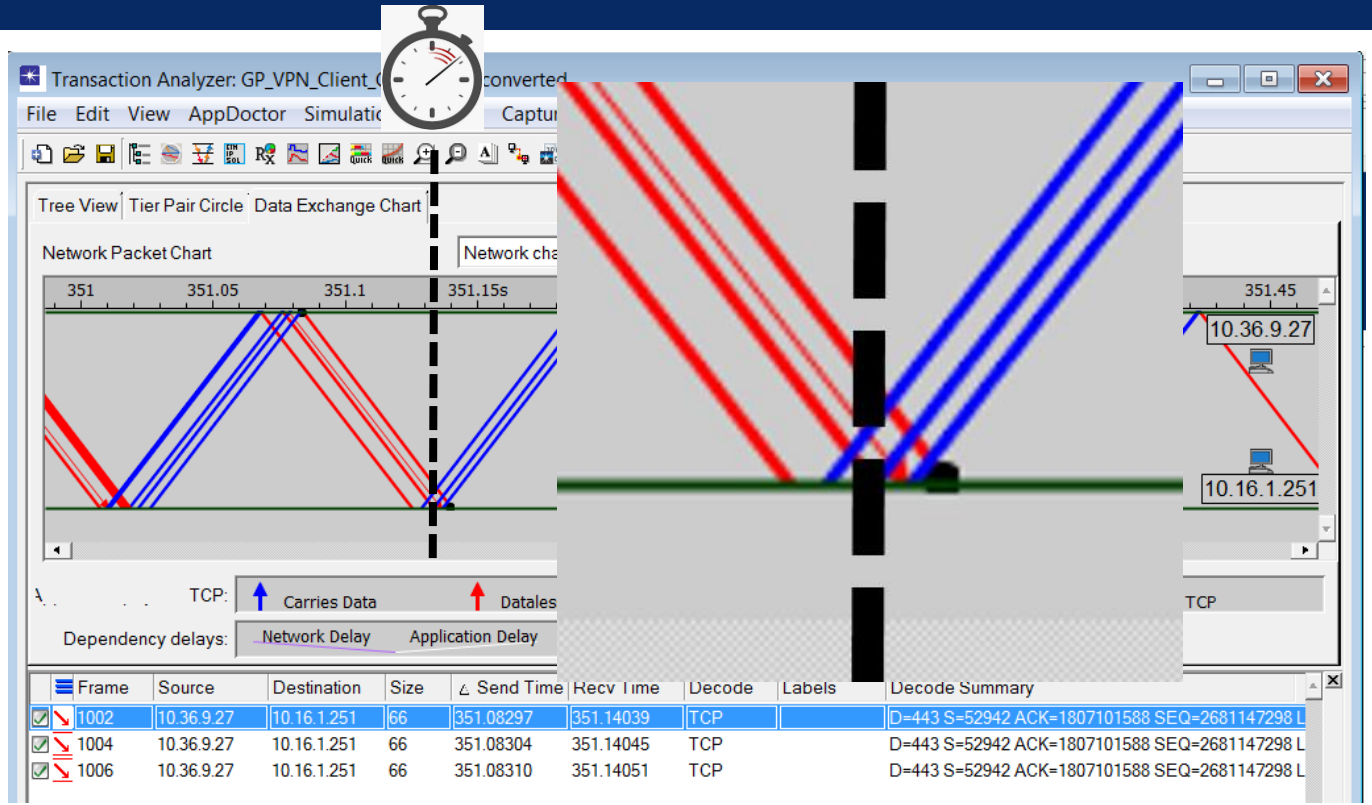
Discuss state of each end point



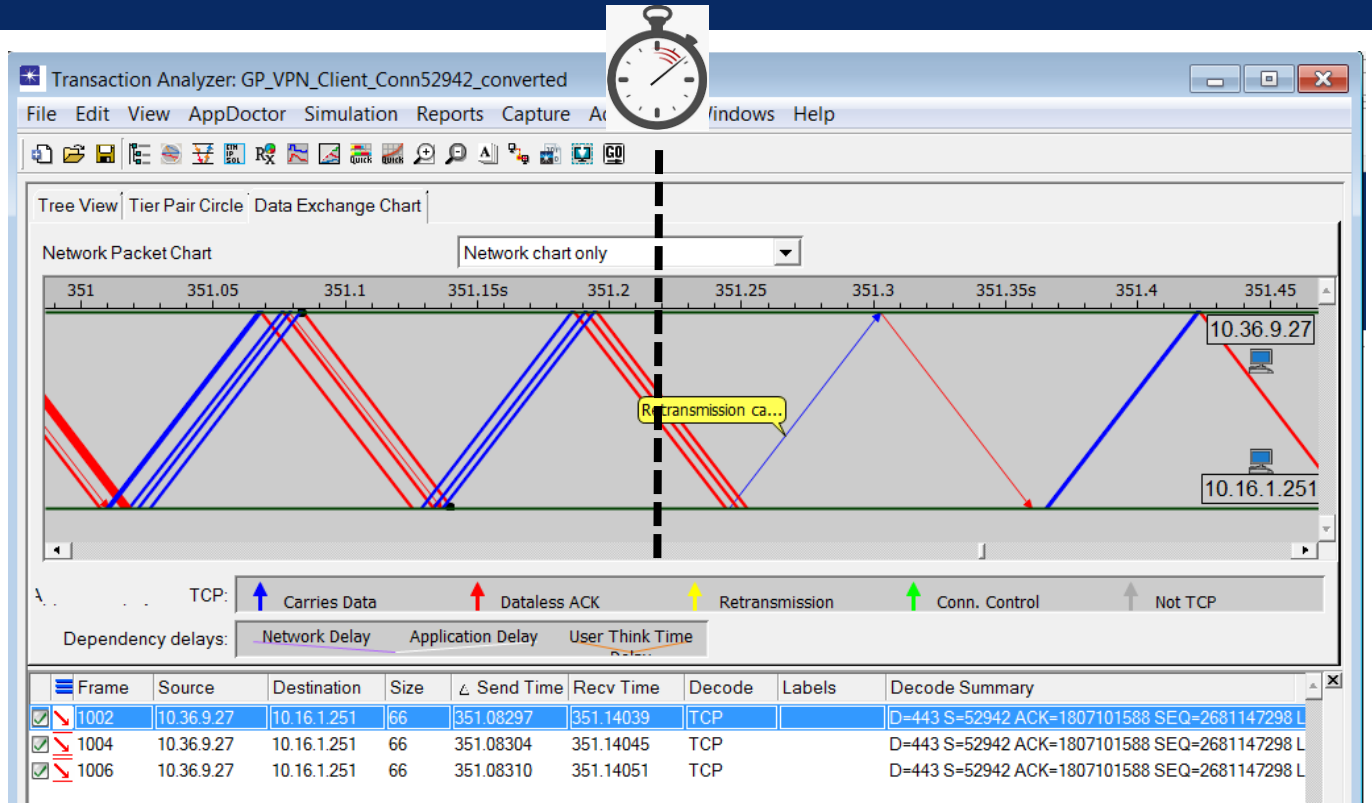
Discuss state of each end point



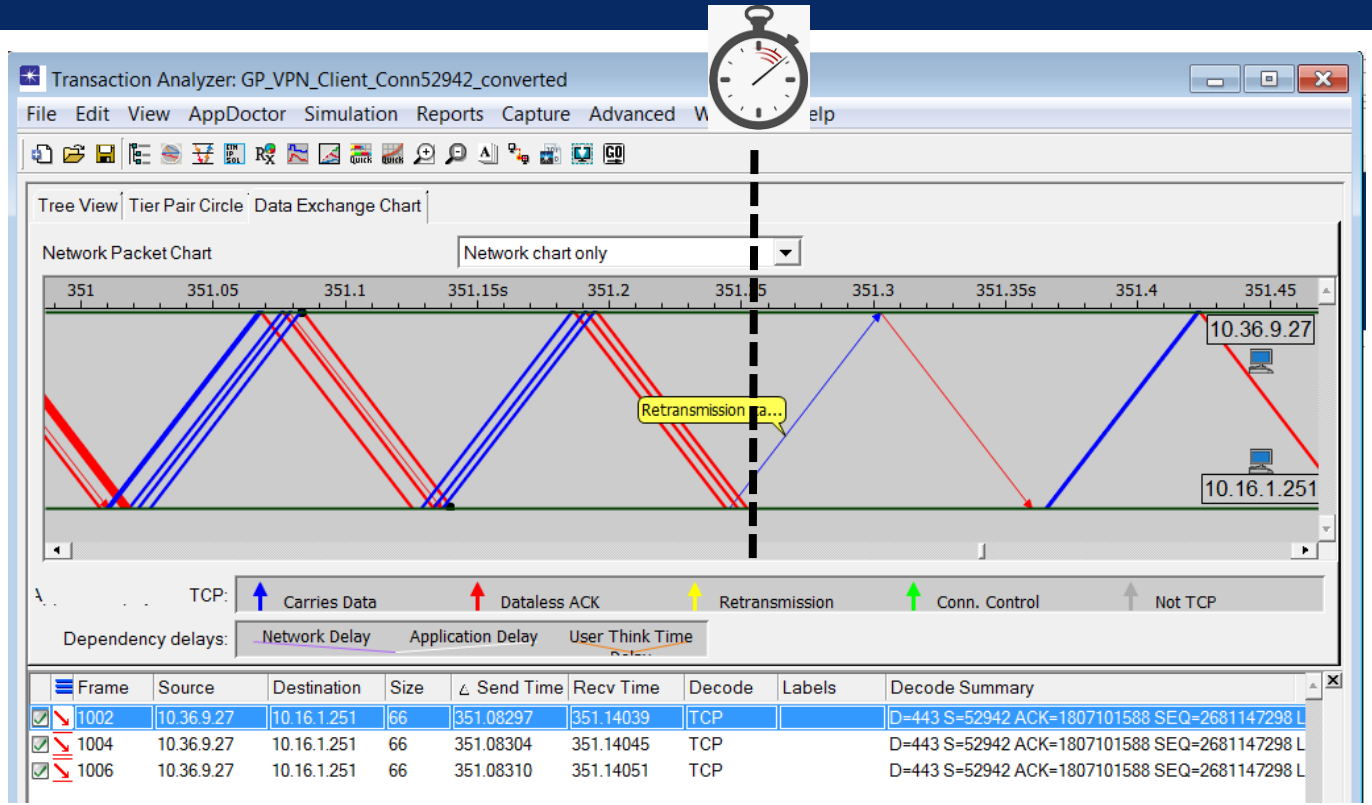
Zoom-in


Transaction Analyzer: GP_VPN_Client_
converted
File Edit View AppDoctor Simulatic
Captur
Tree View
Tier Pair Circle
Data Exchange Chart
Network Packet Chart
Network cha
351
351.05
351.1
351.15s
351.45
10.36.9.27
10.16.1.251
TCP:
Carries Data
Datales
TCP
Dependency delays:
Network Delay
Application Delay
Frame
Source
Destination
Size
Send Time
Recv Time
Decode
Labels
Decode Summary
1002 10.36.9.27 10.16.1.251 66 351.08297 351.14039 TCP D=443 S=52942 ACK=1807101588 SEQ=2681147298 L
1004 10.36.9.27 10.16.1.251 66 351.08304 351.14045 TCP D=443 S=52942 ACK=1807101588 SEQ=2681147298 L
1006 10.36.9.27 10.16.1.251 66 351.08310 351.14051 TCP D=443 S=52942 ACK=1807101588 SEQ=2681147298 L

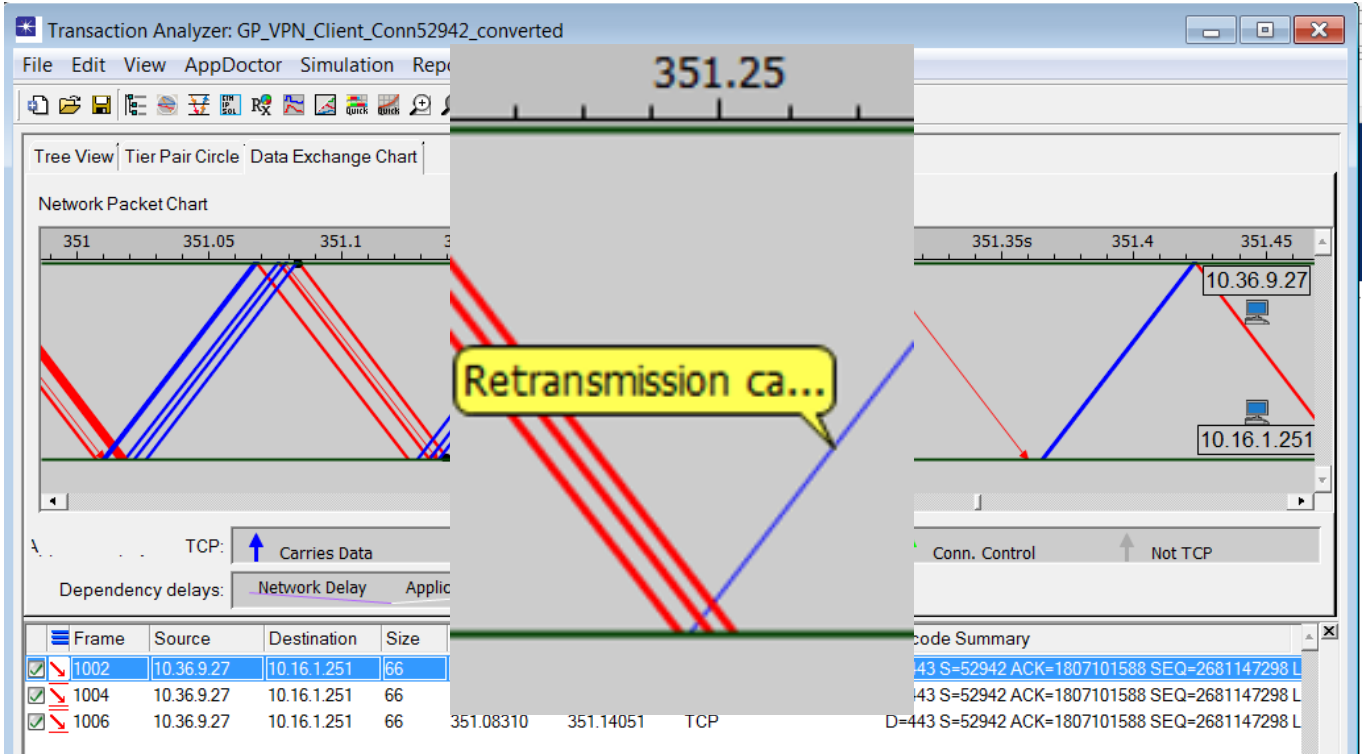
Discuss state of each end point



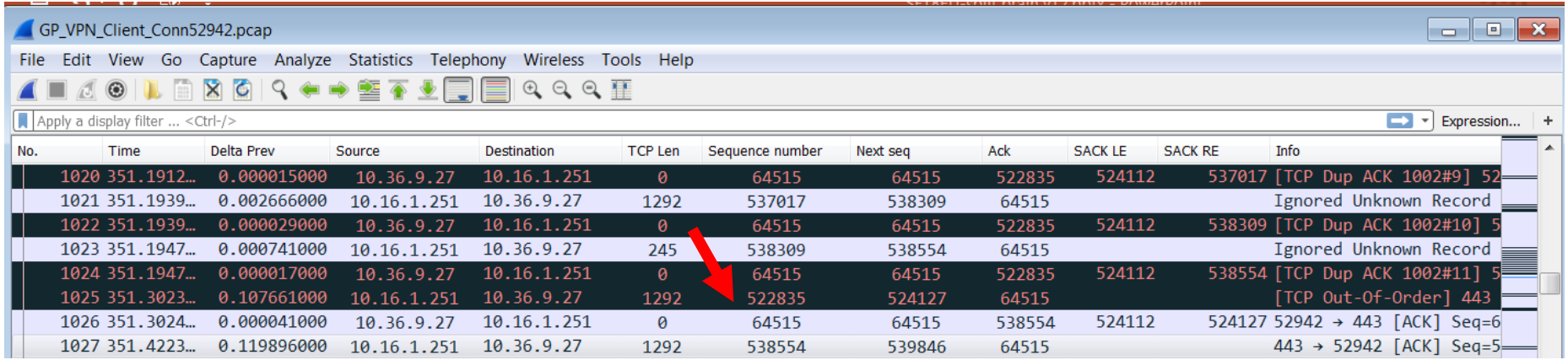
Discuss state of each end point



Discuss state of each end point

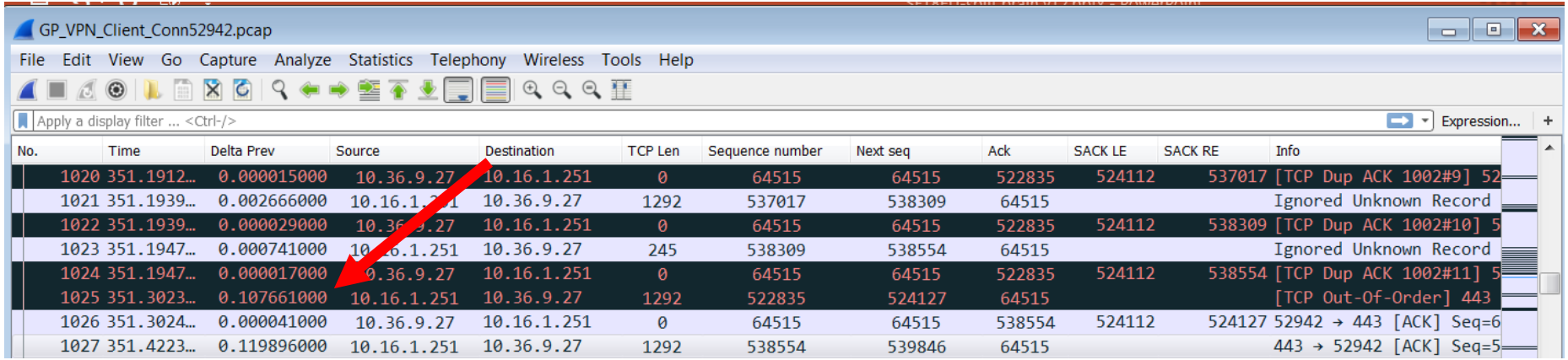


Missing segment finally arrives



- Let's examine a few details...

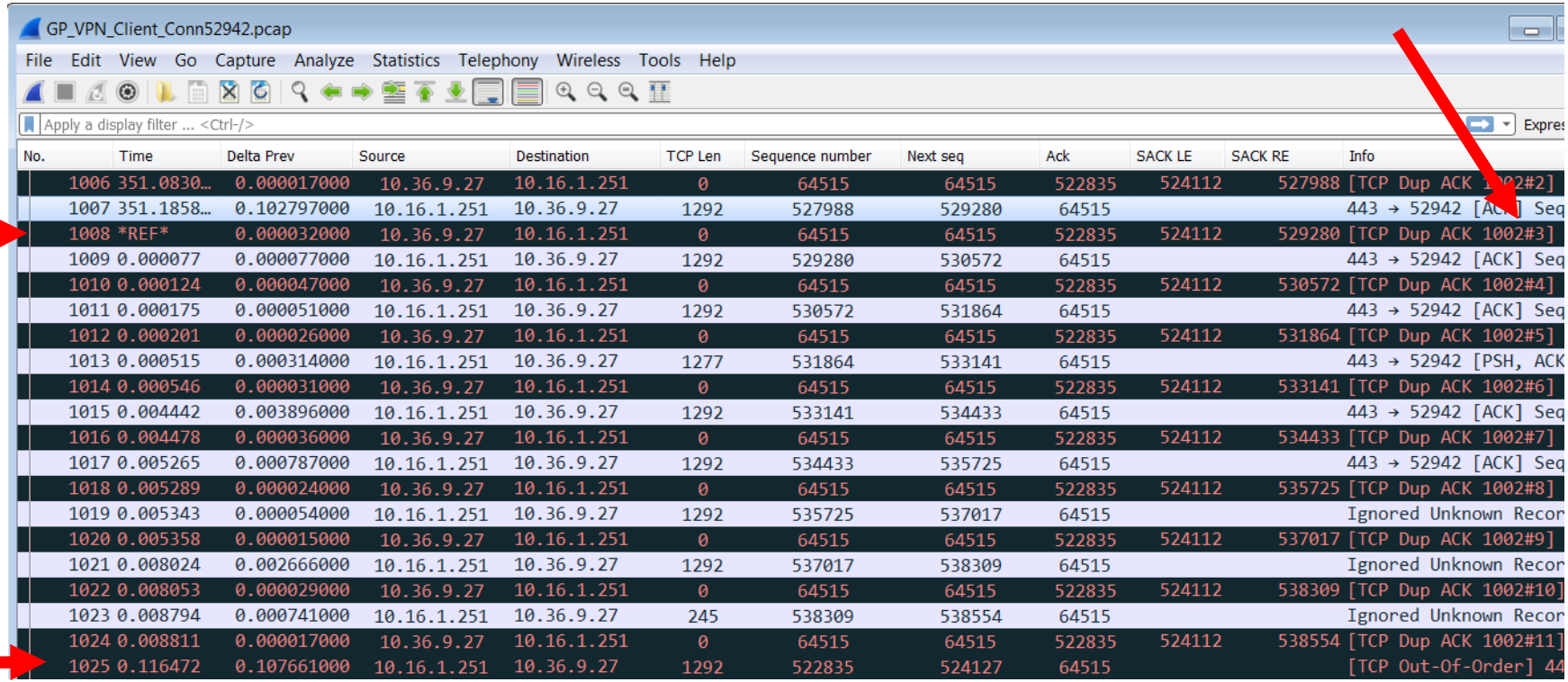
107ms Time Delta



- Why 107ms from previous packet?

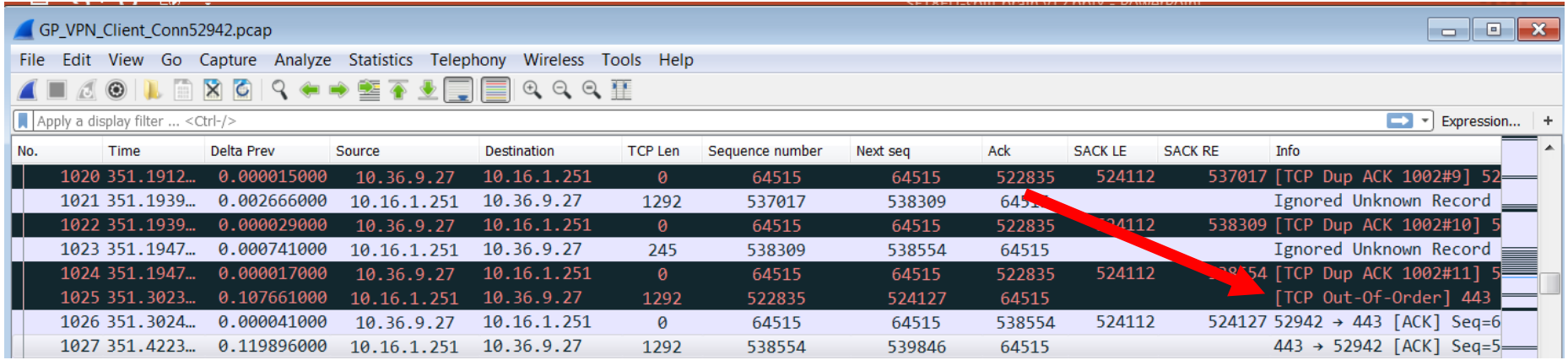
Wireshark Feature Parade

With time ref set to DupACK #3



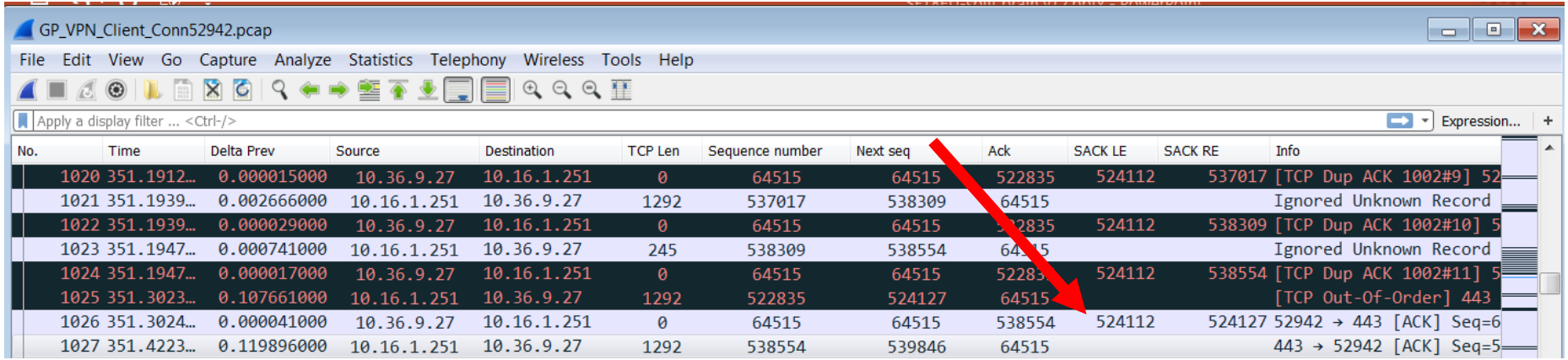
No.	Time	Delta Prev	Source	Destination	TCP Len	Sequence number	Next seq	Ack	SACK LE	SACK RE	Info
1006	351.0830…	0.000017000	10.36.9.27	10.16.1.251	0	64515	64515	522835	524112	527988	[TCP Dup ACK 1002#2]
1007	351.1858…	0.102797000	10.16.1.251	10.36.9.27	1292	527988	529280	64515			443 → 52942 [ACK] Seq
1008	*REF*	0.000032000	10.36.9.27	10.16.1.251	0	64515	64515	522835	524112	529280	[TCP Dup ACK 1002#3]
1009	0.000077	0.000077000	10.16.1.251	10.36.9.27	1292	529280	530572	64515			443 → 52942 [ACK] Seq
1010	0.000124	0.000047000	10.36.9.27	10.16.1.251	0	64515	64515	522835	524112	530572	[TCP Dup ACK 1002#4]
1011	0.000175	0.000051000	10.16.1.251	10.36.9.27	1292	530572	531864	64515			443 → 52942 [ACK] Seq
1012	0.000201	0.000026000	10.36.9.27	10.16.1.251	0	64515	64515	522835	524112	531864	[TCP Dup ACK 1002#5]
1013	0.000515	0.000314000	10.16.1.251	10.36.9.27	1277	531864	533141	64515			443 → 52942 [PSH, ACK
1014	0.000546	0.000031000	10.36.9.27	10.16.1.251	0	64515	64515	522835	524112	533141	[TCP Dup ACK 1002#6]
1015	0.004442	0.003896000	10.16.1.251	10.36.9.27	1292	533141	534433	64515			443 → 52942 [ACK] Seq
1016	0.004478	0.000036000	10.36.9.27	10.16.1.251	0	64515	64515	522835	524112	534433	[TCP Dup ACK 1002#7]
1017	0.005265	0.000787000	10.16.1.251	10.36.9.27	1292	534433	535725	64515			443 → 52942 [ACK] Seq
1018	0.005289	0.000024000	10.36.9.27	10.16.1.251	0	64515	64515	522835	524112	535725	[TCP Dup ACK 1002#8]
1019	0.005343	0.000054000	10.16.1.251	10.36.9.27	1292	535725	537017	64515			Ignored Unknown Recor
1020	0.005358	0.000015000	10.36.9.27	10.16.1.251	0	64515	64515	522835	524112	537017	[TCP Dup ACK 1002#9]
1021	0.008024	0.002666000	10.16.1.251	10.36.9.27	1292	537017	538309	64515			Ignored Unknown Recor
1022	0.008053	0.000029000	10.36.9.27	10.16.1.251	0	64515	64515	522835	524112	538309	[TCP Dup ACK 1002#10]
1023	0.008794	0.000741000	10.16.1.251	10.36.9.27	245	538309	538554	64515			Ignored Unknown Recor
1024	0.008811	0.000017000	10.36.9.27	10.16.1.251	0	64515	64515	522835	524112	538554	[TCP Dup ACK 1002#11]
1025	0.116472	0.107661000	10.16.1.251	10.36.9.27	1292	522835	524127	64515			[TCP Out-Of-Order] 44

Why “Out of Order”?



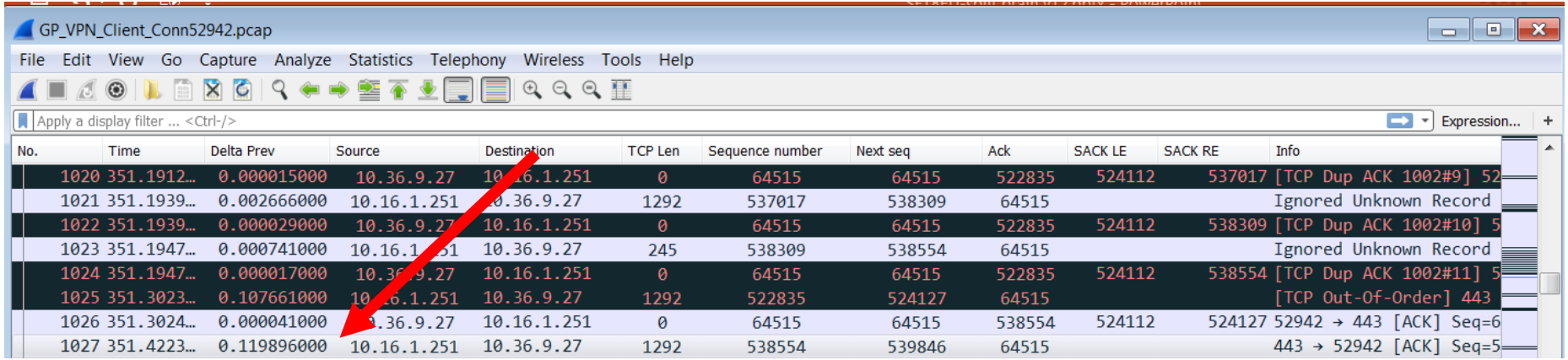
- Wouldn’t “Retrans” or “Fast Retrans” be more accurate?

Why ACK with SACK Fields?



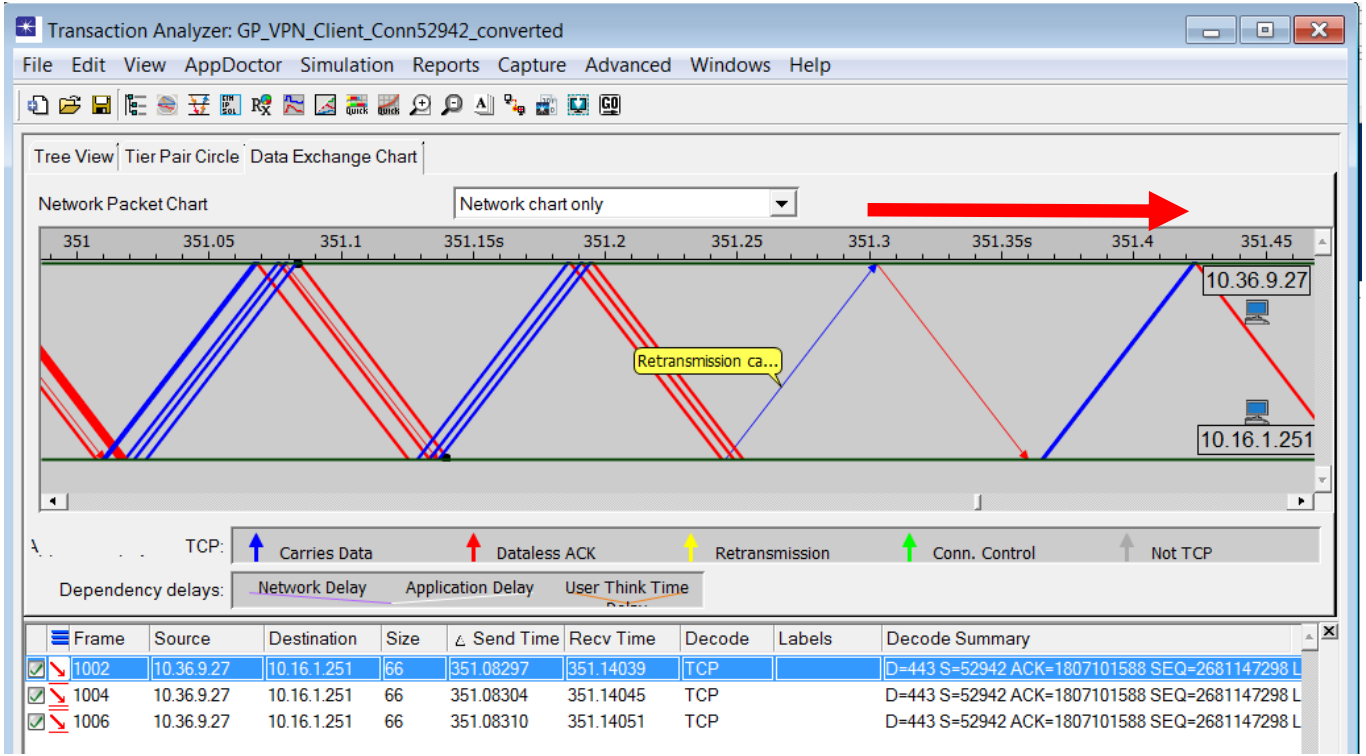
- We're all caught up, nothing is missing…so why send SACK?
- "Oh, btw Sender – I received these 15 bytes of TCP stream unexpectedly. Just letting you know"

Why 119ms delay here?



- What does this tell us about bytes in flight and the congestion window?

Here's the delay – 1 x RTT



Discussion

- Did you find that interesting?
- Anything new you've not seen before?
- Do you find visualization helpful?
- Other Comments?

Summary & Wrap-Up

- Interpreting TCP behavior can be confusing and complicated, especially when there is “high” latency in the path
- Captures from both end points can be beneficial
- You need to split your brain into “server perspective” and “client perspective”

Summary & Wrap-Up

- Wireshark features are extremely helpful, but can only take you so far
- Visualization can help you understand behavior and quickly interpret root cause
- Advanced analytics are icing on the cake…

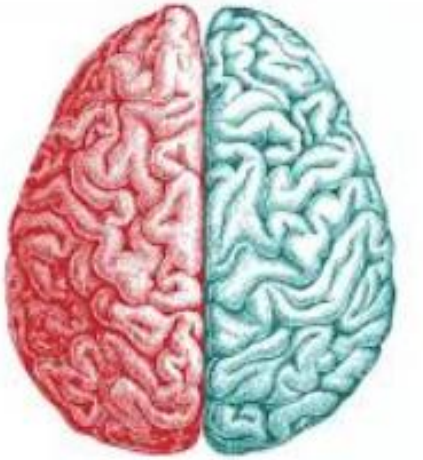
Thank You!

- For your participation
- Hope you found this session informative and insightful
- There’s a bonus section to compare Bytes in flight, followed by Latency & Congestion that follow for your viewing enjoyment after the session

End of Session

Split Brain Comparisons

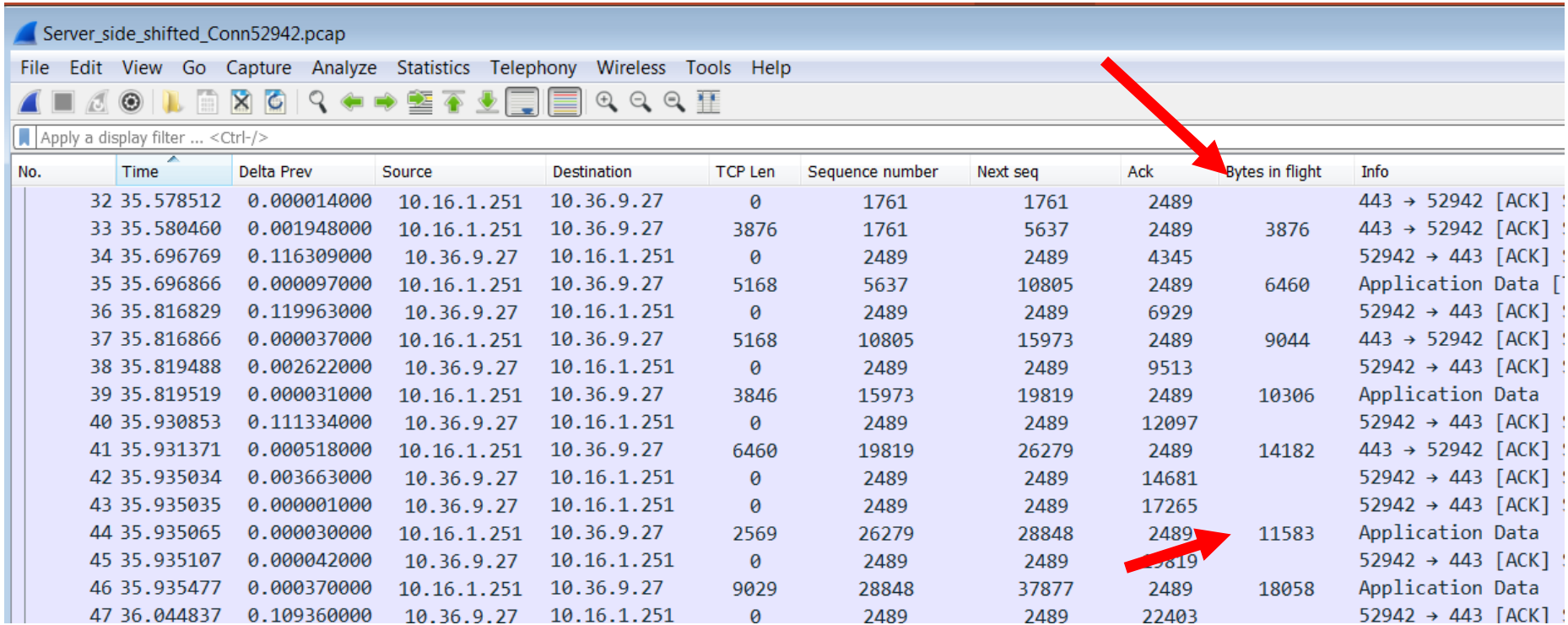
- Bytes in Flight



Review

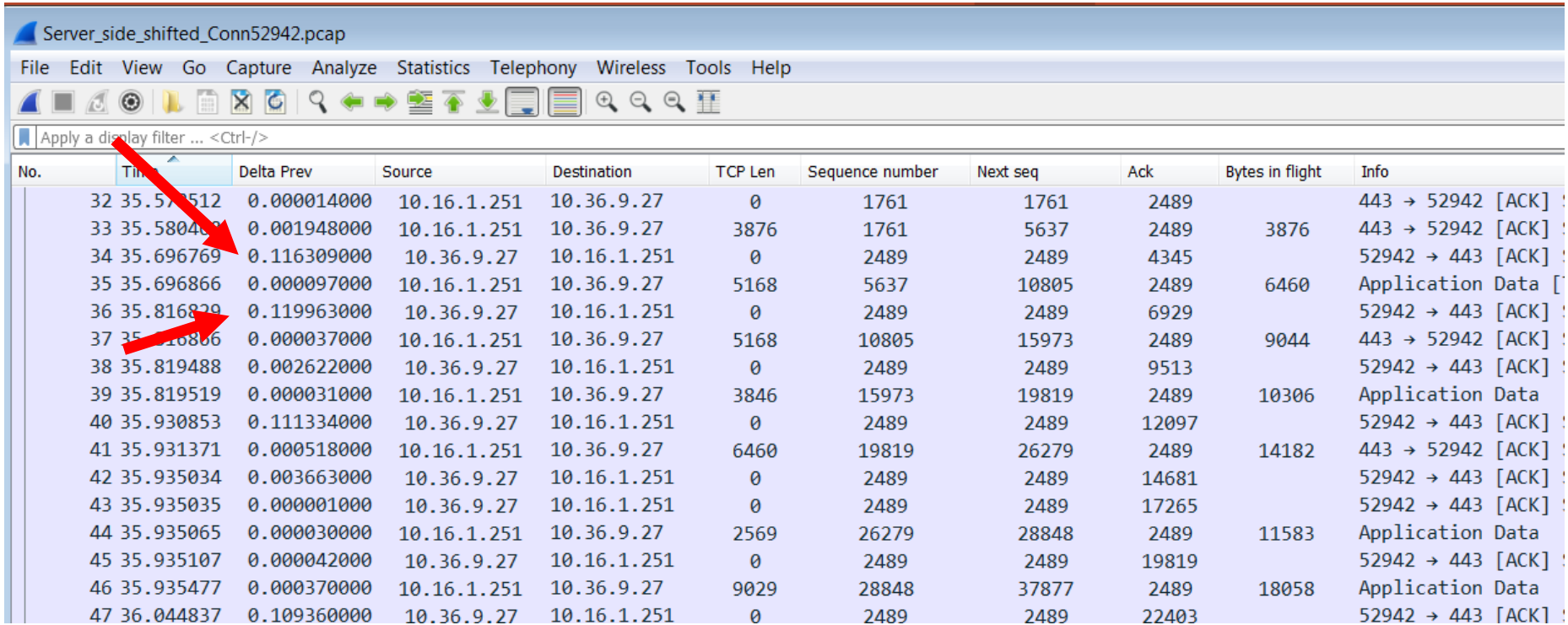
- What is the meaning of Bytes in Flight?
- How is this metric related to performance?
- BIF for Server and client captures look very different, let’s compare a few packet exchanges

Server Side



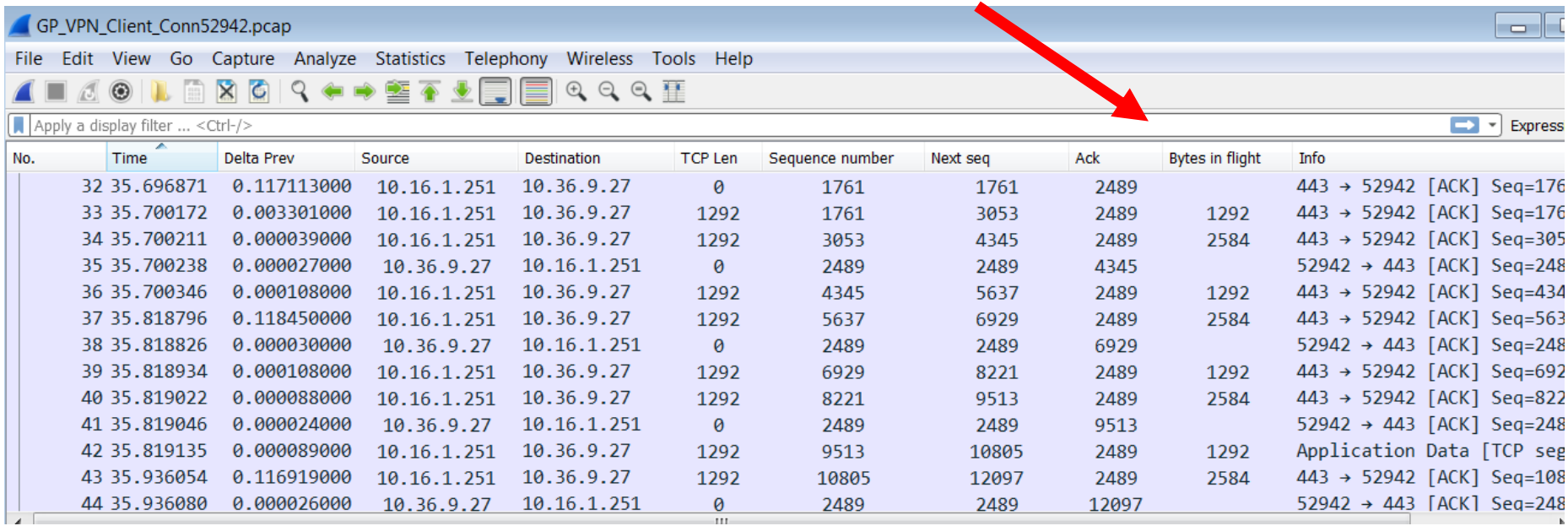
No.	Time	Delta Prev	Source	Destination	TCP Len	Sequence number	Next seq	Ack	Bytes in flight	Info
32	35.578512	0.000014000	10.16.1.251	10.36.9.27	0	1761	1761	2489		443 → 52942 [ACK]
33	35.580460	0.001948000	10.16.1.251	10.36.9.27	3876	1761	5637	2489	3876	443 → 52942 [ACK]
34	35.696769	0.116309000	10.36.9.27	10.16.1.251	0	2489	2489	4345		52942 → 443 [ACK]
35	35.696866	0.000097000	10.16.1.251	10.36.9.27	5168	5637	10805	2489	6460	Application Data [
36	35.816829	0.119963000	10.36.9.27	10.16.1.251	0	2489	2489	6929		52942 → 443 [ACK]
37	35.816866	0.000037000	10.16.1.251	10.36.9.27	5168	10805	15973	2489	9044	443 → 52942 [ACK]
38	35.819488	0.002622000	10.36.9.27	10.16.1.251	0	2489	2489	9513		52942 → 443 [ACK]
39	35.819519	0.000031000	10.16.1.251	10.36.9.27	3846	15973	19819	2489	10306	Application Data
40	35.930853	0.111334000	10.36.9.27	10.16.1.251	0	2489	2489	12097		52942 → 443 [ACK]
41	35.931371	0.000518000	10.16.1.251	10.36.9.27	6460	19819	26279	2489	14182	443 → 52942 [ACK]
42	35.935034	0.003663000	10.36.9.27	10.16.1.251	0	2489	2489	14681		52942 → 443 [ACK]
43	35.935035	0.000001000	10.36.9.27	10.16.1.251	0	2489	2489	17265		52942 → 443 [ACK]
44	35.935065	0.000030000	10.16.1.251	10.36.9.27	2569	26279	28848	2489	11583	Application Data
45	35.935107	0.000042000	10.36.9.27	10.16.1.251	0	2489	2489	19819		52942 → 443 [ACK]
46	35.935477	0.000370000	10.16.1.251	10.36.9.27	9029	28848	37877	2489	18058	Application Data
47	36.044837	0.109360000	10.36.9.27	10.16.1.251	0	2489	2489	22403		52942 → 443 [ACK]

Server Side



No.	Time	Delta Prev	Source	Destination	TCP Len	Sequence number	Next seq	Ack	Bytes in flight	Info
32	35.57[illegible]512	0.000014000	10.16.1.251	10.36.9.27	0	1761	1761	2489		443 → 52942 [ACK]
33	35.580[illegible]	0.001948000	10.16.1.251	10.36.9.27	3876	1761	5637	2489	3876	443 → 52942 [ACK]
34	35.696769	0.116309000	10.36.9.27	10.16.1.251	0	2489	2489	4345		52942 → 443 [ACK]
35	35.696866	0.000097000	10.16.1.251	10.36.9.27	5168	5637	10805	2489	6460	Application Data [
36	35.816829	0.119963000	10.36.9.27	10.16.1.251	0	2489	2489	6929		52942 → 443 [ACK]
37	35.[illegible]816866	0.000037000	10.16.1.251	10.36.9.27	5168	10805	15973	2489	9044	443 → 52942 [ACK]
38	35.819488	0.002622000	10.36.9.27	10.16.1.251	0	2489	2489	9513		52942 → 443 [ACK]
39	35.819519	0.000031000	10.16.1.251	10.36.9.27	3846	15973	19819	2489	10306	Application Data
40	35.930853	0.111334000	10.36.9.27	10.16.1.251	0	2489	2489	12097		52942 → 443 [ACK]
41	35.931371	0.000518000	10.16.1.251	10.36.9.27	6460	19819	26279	2489	14182	443 → 52942 [ACK]
42	35.935034	0.003663000	10.36.9.27	10.16.1.251	0	2489	2489	14681		52942 → 443 [ACK]
43	35.935035	0.000001000	10.36.9.27	10.16.1.251	0	2489	2489	17265		52942 → 443 [ACK]
44	35.935065	0.000030000	10.16.1.251	10.36.9.27	2569	26279	28848	2489	11583	Application Data
45	35.935107	0.000042000	10.36.9.27	10.16.1.251	0	2489	2489	19819		52942 → 443 [ACK]
46	35.935477	0.000370000	10.16.1.251	10.36.9.27	9029	28848	37877	2489	18058	Application Data
47	36.044837	0.109360000	10.36.9.27	10.16.1.251	0	2489	2489	22403		52942 → 443 [ACK]

Client Side



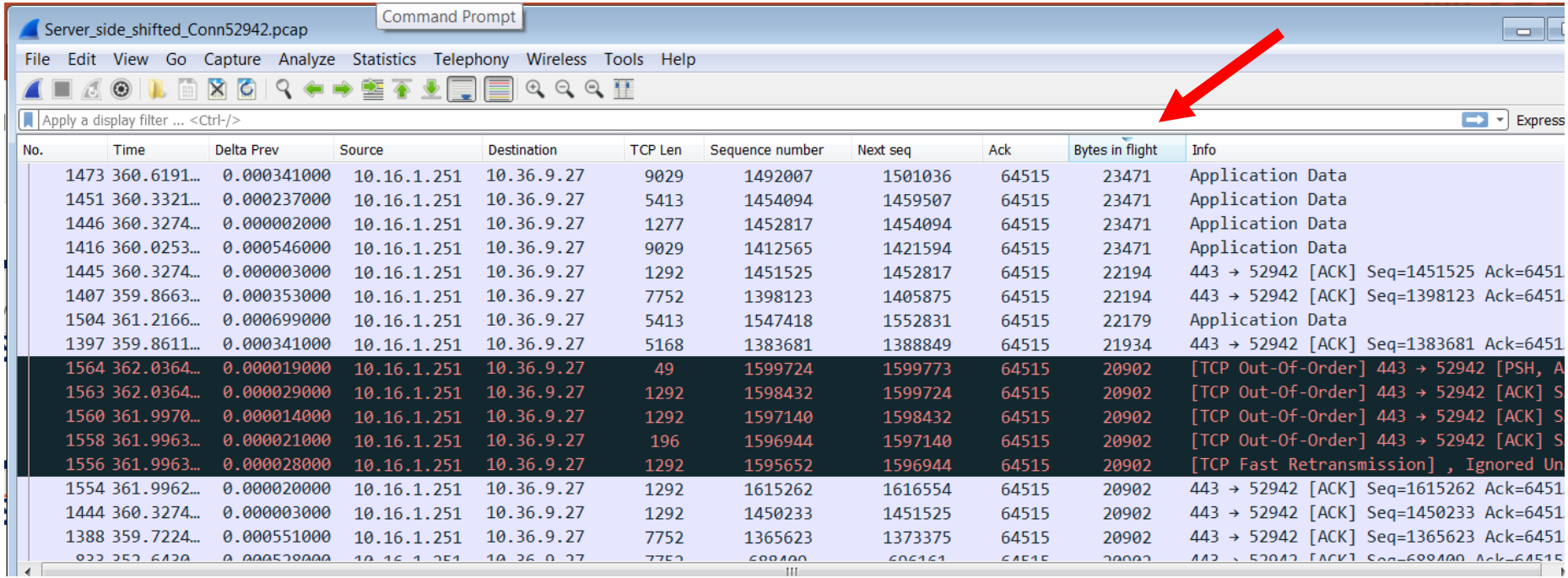
No.	Time	Delta Prev	Source	Destination	TCP Len	Sequence number	Next seq	Ack	Bytes in flight	Info
32	35.696871	0.117113000	10.16.1.251	10.36.9.27	0	1761	1761	2489		443 → 52942 [ACK] Seq=176
33	35.700172	0.003301000	10.16.1.251	10.36.9.27	1292	1761	3053	2489	1292	443 → 52942 [ACK] Seq=176
34	35.700211	0.000039000	10.16.1.251	10.36.9.27	1292	3053	4345	2489	2584	443 → 52942 [ACK] Seq=305
35	35.700238	0.000027000	10.36.9.27	10.16.1.251	0	2489	2489	4345		52942 → 443 [ACK] Seq=248
36	35.700346	0.000108000	10.16.1.251	10.36.9.27	1292	4345	5637	2489	1292	443 → 52942 [ACK] Seq=434
37	35.818796	0.118450000	10.16.1.251	10.36.9.27	1292	5637	6929	2489	2584	443 → 52942 [ACK] Seq=563
38	35.818826	0.000030000	10.36.9.27	10.16.1.251	0	2489	2489	6929		52942 → 443 [ACK] Seq=248
39	35.818934	0.000108000	10.16.1.251	10.36.9.27	1292	6929	8221	2489	1292	443 → 52942 [ACK] Seq=692
40	35.819022	0.000088000	10.16.1.251	10.36.9.27	1292	8221	9513	2489	2584	443 → 52942 [ACK] Seq=822
41	35.819046	0.000024000	10.36.9.27	10.16.1.251	0	2489	2489	9513		52942 → 443 [ACK] Seq=248
42	35.819135	0.000089000	10.16.1.251	10.36.9.27	1292	9513	10805	2489	1292	Application Data [TCP seg
43	35.936054	0.116919000	10.16.1.251	10.36.9.27	1292	10805	12097	2489	2584	443 → 52942 [ACK] Seq=108
44	35.936080	0.000026000	10.36.9.27	10.16.1.251	0	2489	2489	12097		52942 → 443 [ACK] Seq=248

Inferring Send Buffer Size

- Bytes In-flight can give you some insight into how the Send Buffer size might be limiting the Congestion Window

- How can we find the maximum observed Bytes In-Flight?

Server Capture

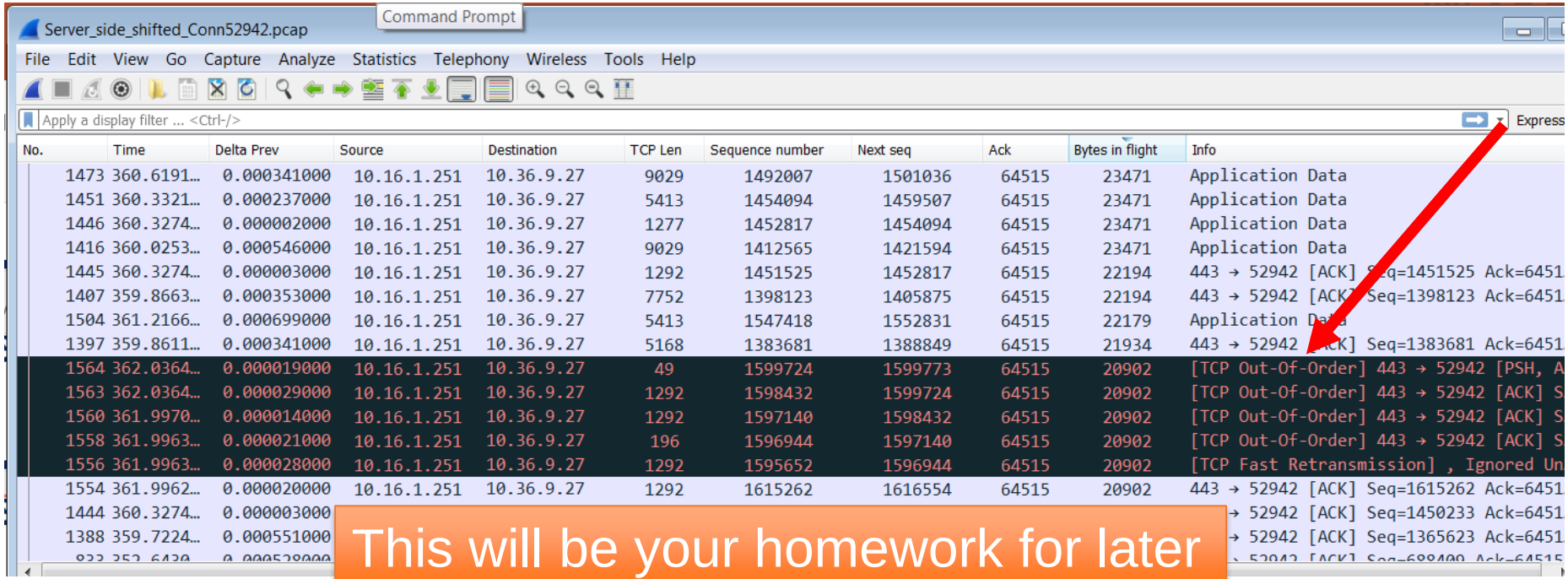


No.	Time	Delta Prev	Source	Destination	TCP Len	Sequence number	Next seq	Ack	Bytes in flight	Info
1473	360.6191…	0.000341000	10.16.1.251	10.36.9.27	9029	1492007	1501036	64515	23471	Application Data
1451	360.3321…	0.000237000	10.16.1.251	10.36.9.27	5413	1454094	1459507	64515	23471	Application Data
1446	360.3274…	0.000002000	10.16.1.251	10.36.9.27	1277	1452817	1454094	64515	23471	Application Data
1416	360.0253…	0.000546000	10.16.1.251	10.36.9.27	9029	1412565	1421594	64515	23471	Application Data
1445	360.3274…	0.000003000	10.16.1.251	10.36.9.27	1292	1451525	1452817	64515	22194	443 → 52942 [ACK] Seq=1451525 Ack=6451
1407	359.8663…	0.000353000	10.16.1.251	10.36.9.27	7752	1398123	1405875	64515	22194	443 → 52942 [ACK] Seq=1398123 Ack=6451
1504	361.2166…	0.000699000	10.16.1.251	10.36.9.27	5413	1547418	1552831	64515	22179	Application Data
1397	359.8611…	0.000341000	10.16.1.251	10.36.9.27	5168	1383681	1388849	64515	21934	443 → 52942 [ACK] Seq=1383681 Ack=6451
1564	362.0364…	0.000019000	10.16.1.251	10.36.9.27	49	1599724	1599773	64515	20902	[TCP Out-Of-Order] 443 → 52942 [PSH, A
1563	362.0364…	0.000029000	10.16.1.251	10.36.9.27	1292	1598432	1599724	64515	20902	[TCP Out-Of-Order] 443 → 52942 [ACK] S
1560	361.9970…	0.000014000	10.16.1.251	10.36.9.27	1292	1597140	1598432	64515	20902	[TCP Out-Of-Order] 443 → 52942 [ACK] S
1558	361.9963…	0.000021000	10.16.1.251	10.36.9.27	196	1596944	1597140	64515	20902	[TCP Out-Of-Order] 443 → 52942 [ACK] S
1556	361.9963…	0.000028000	10.16.1.251	10.36.9.27	1292	1595652	1596944	64515	20902	[TCP Fast Retransmission] , Ignored Un
1554	361.9962…	0.000020000	10.16.1.251	10.36.9.27	1292	1615262	1616554	64515	20902	443 → 52942 [ACK] Seq=1615262 Ack=6451
1444	360.3274…	0.000003000	10.16.1.251	10.36.9.27	1292	1450233	1451525	64515	20902	443 → 52942 [ACK] Seq=1450233 Ack=6451
1388	359.7224…	0.000551000	10.16.1.251	10.36.9.27	7752	1365623	1373375	64515	20902	443 → 52942 [ACK] Seq=1365623 Ack=6451

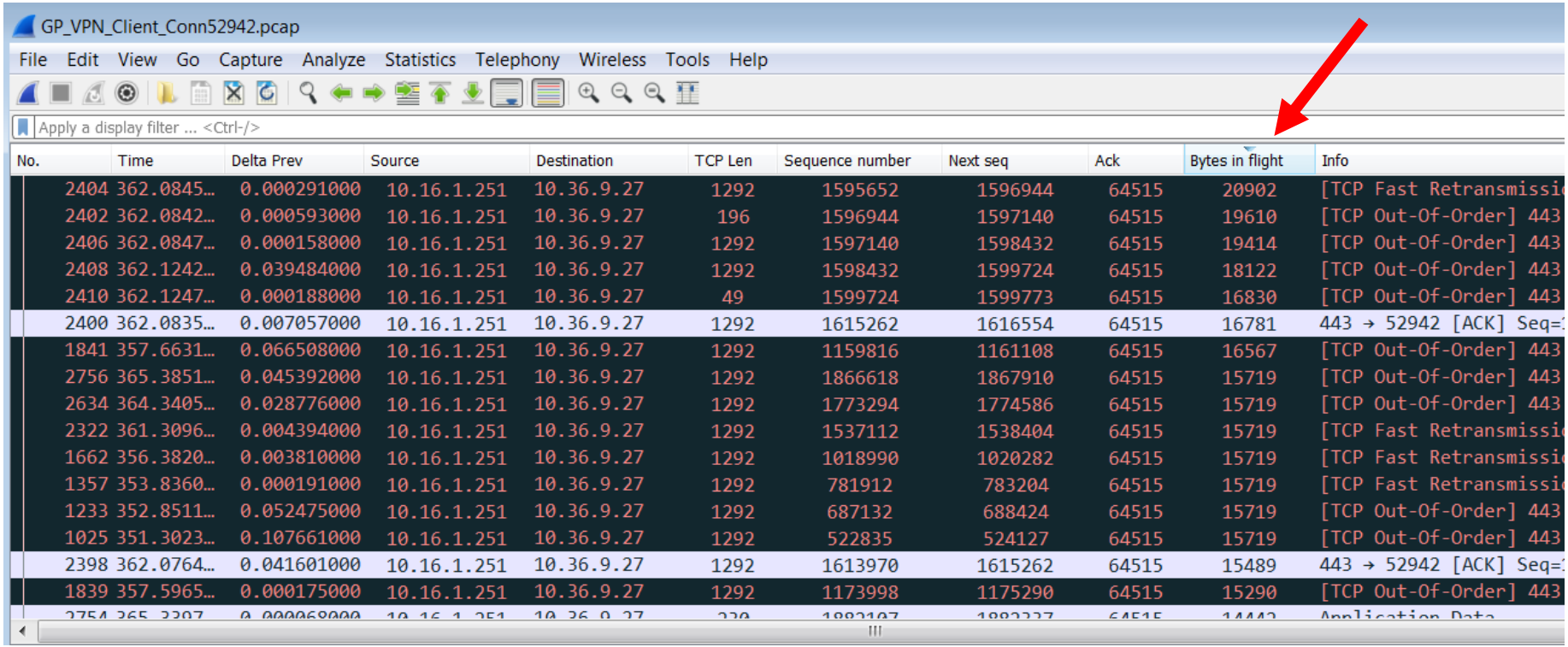
Wait, something was odd…

- Did you notice anything odd in the last screen?
- Let’s look again…

Server Capture – Out of order?



BIF - Client Side

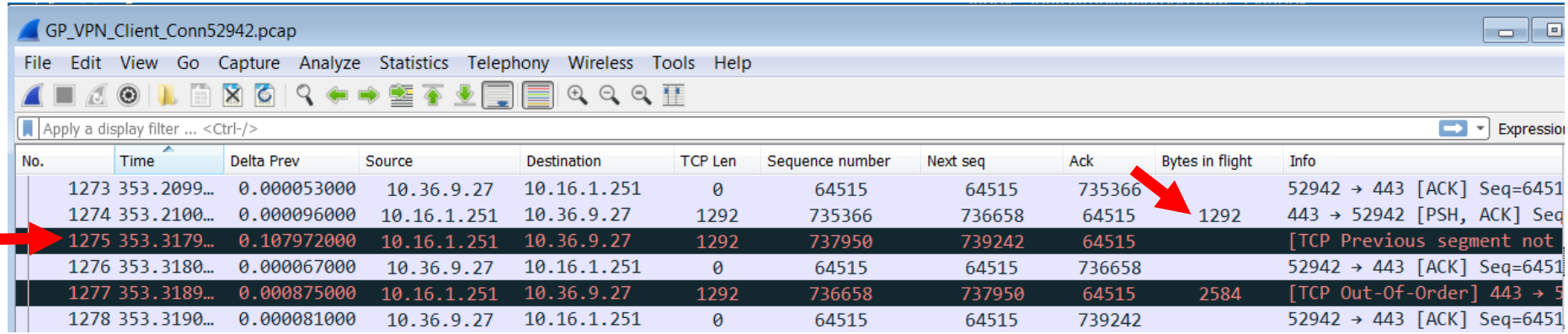


No.	Time	Delta Prev	Source	Destination	TCP Len	Sequence number	Next seq	Ack	Bytes in flight	Info
2404	362.0845…	0.000291000	10.16.1.251	10.36.9.27	1292	1595652	1596944	64515	20902	[TCP Fast Retransmissi
2402	362.0842…	0.000593000	10.16.1.251	10.36.9.27	196	1596944	1597140	64515	19610	[TCP Out-Of-Order] 443
2406	362.0847…	0.000158000	10.16.1.251	10.36.9.27	1292	1597140	1598432	64515	19414	[TCP Out-Of-Order] 443
2408	362.1242…	0.039484000	10.16.1.251	10.36.9.27	1292	1598432	1599724	64515	18122	[TCP Out-Of-Order] 443
2410	362.1247…	0.000188000	10.16.1.251	10.36.9.27	49	1599724	1599773	64515	16830	[TCP Out-Of-Order] 443
2400	362.0835…	0.007057000	10.16.1.251	10.36.9.27	1292	1615262	1616554	64515	16781	443 → 52942 [ACK] Seq=
1841	357.6631…	0.066508000	10.16.1.251	10.36.9.27	1292	1159816	1161108	64515	16567	[TCP Out-Of-Order] 443
2756	365.3851…	0.045392000	10.16.1.251	10.36.9.27	1292	1866618	1867910	64515	15719	[TCP Out-Of-Order] 443
2634	364.3405…	0.028776000	10.16.1.251	10.36.9.27	1292	1773294	1774586	64515	15719	[TCP Out-Of-Order] 443
2322	361.3096…	0.004394000	10.16.1.251	10.36.9.27	1292	1537112	1538404	64515	15719	[TCP Fast Retransmissi
1662	356.3820…	0.003810000	10.16.1.251	10.36.9.27	1292	1018990	1020282	64515	15719	[TCP Fast Retransmissi
1357	353.8360…	0.000191000	10.16.1.251	10.36.9.27	1292	781912	783204	64515	15719	[TCP Fast Retransmissi
1233	352.8511…	0.052475000	10.16.1.251	10.36.9.27	1292	687132	688424	64515	15719	[TCP Out-Of-Order] 443
1025	351.3023…	0.107661000	10.16.1.251	10.36.9.27	1292	522835	524127	64515	15719	[TCP Out-Of-Order] 443
2398	362.0764…	0.041601000	10.16.1.251	10.36.9.27	1292	1613970	1615262	64515	15489	443 → 52942 [ACK] Seq=
1839	357.5965…	0.000175000	10.16.1.251	10.36.9.27	1292	1173998	1175290	64515	15290	[TCP Out-Of-Order] 443

In-Flight & OOS / Loss

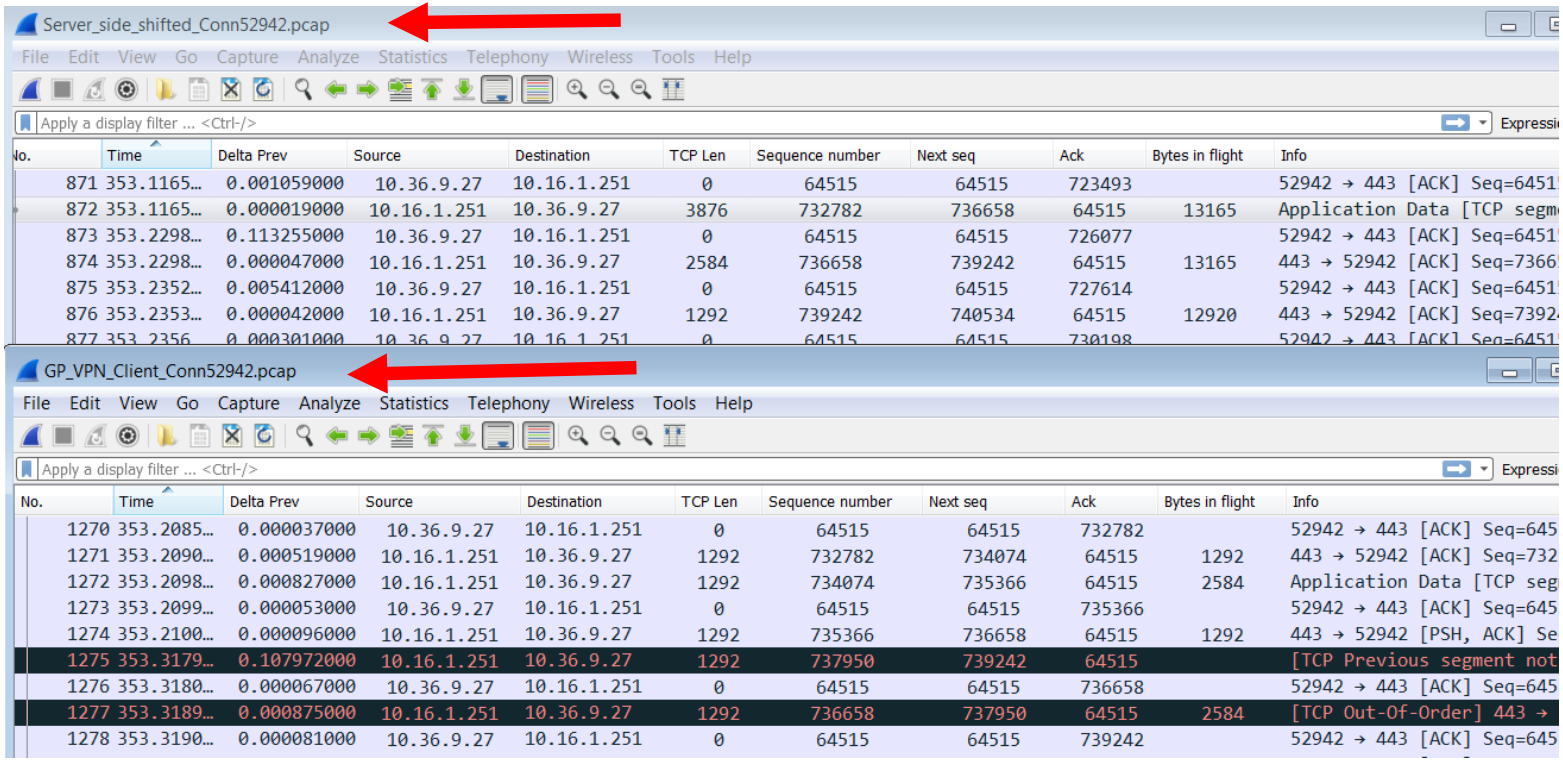
- Let’s look at how Wireshark calculates Bytes in-flight when there is packet loss or OOS

Client – Receive #1275 OOS

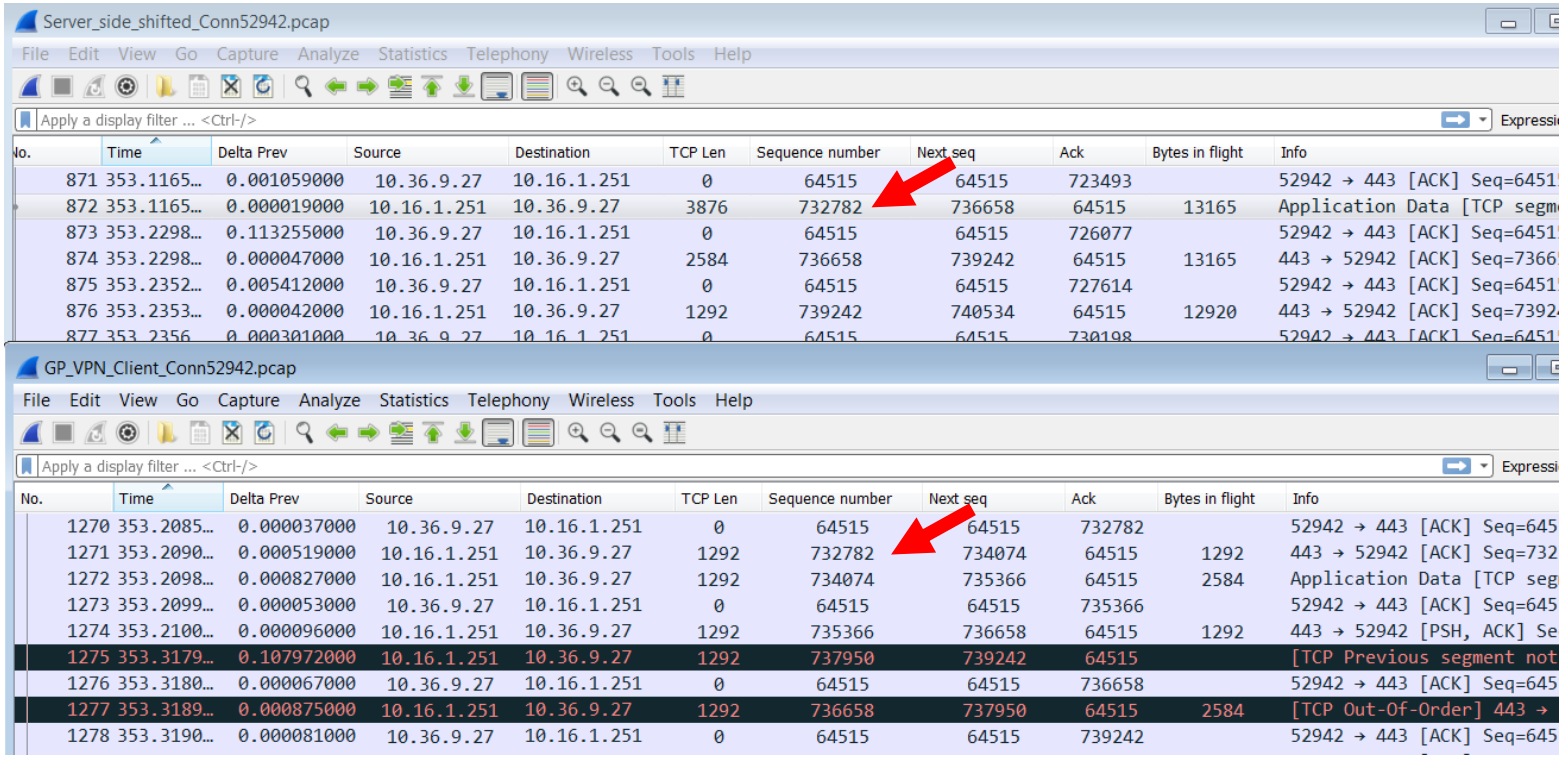


- BIF did not increment for #1275

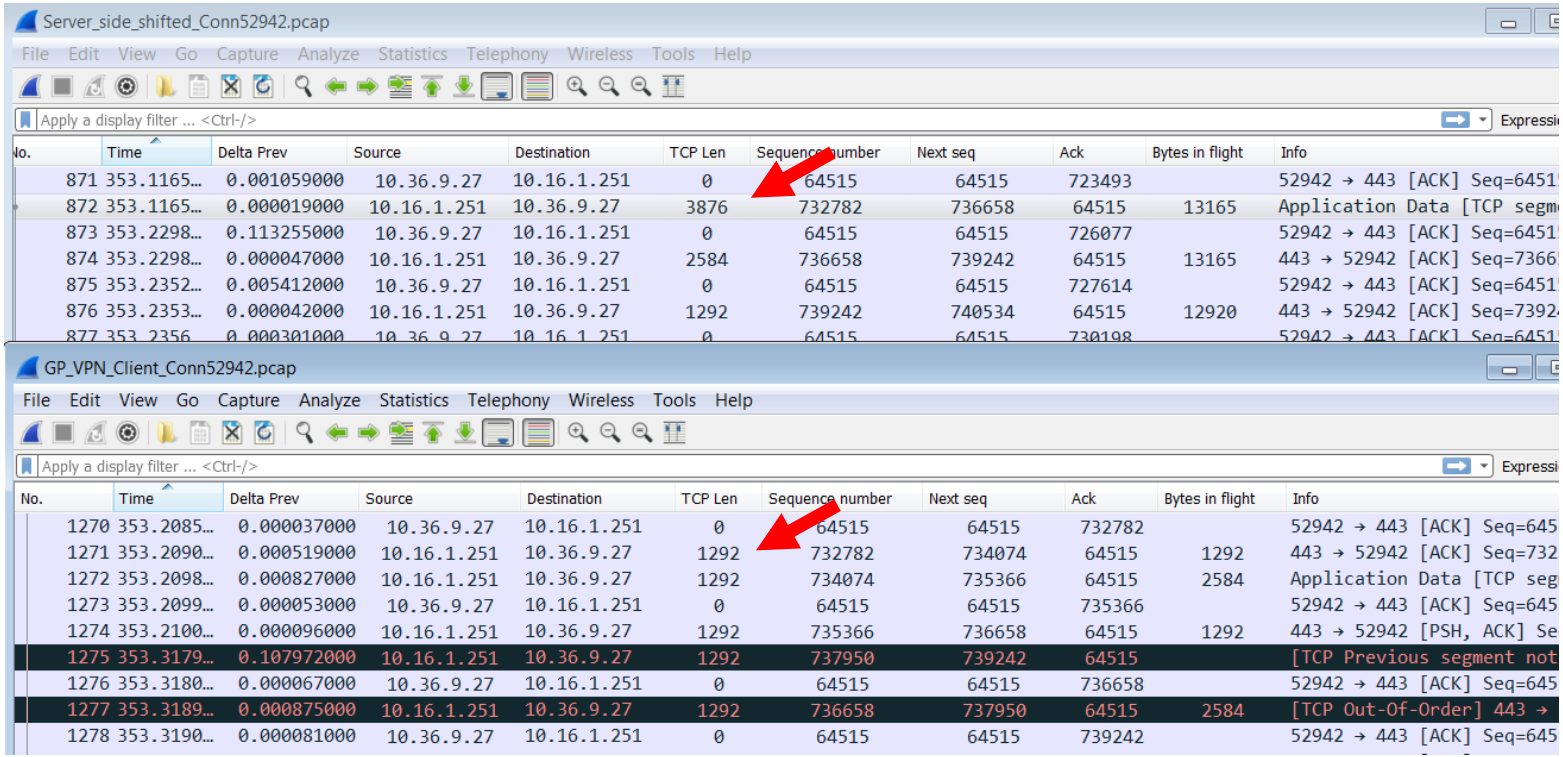
Comparison Storyboard



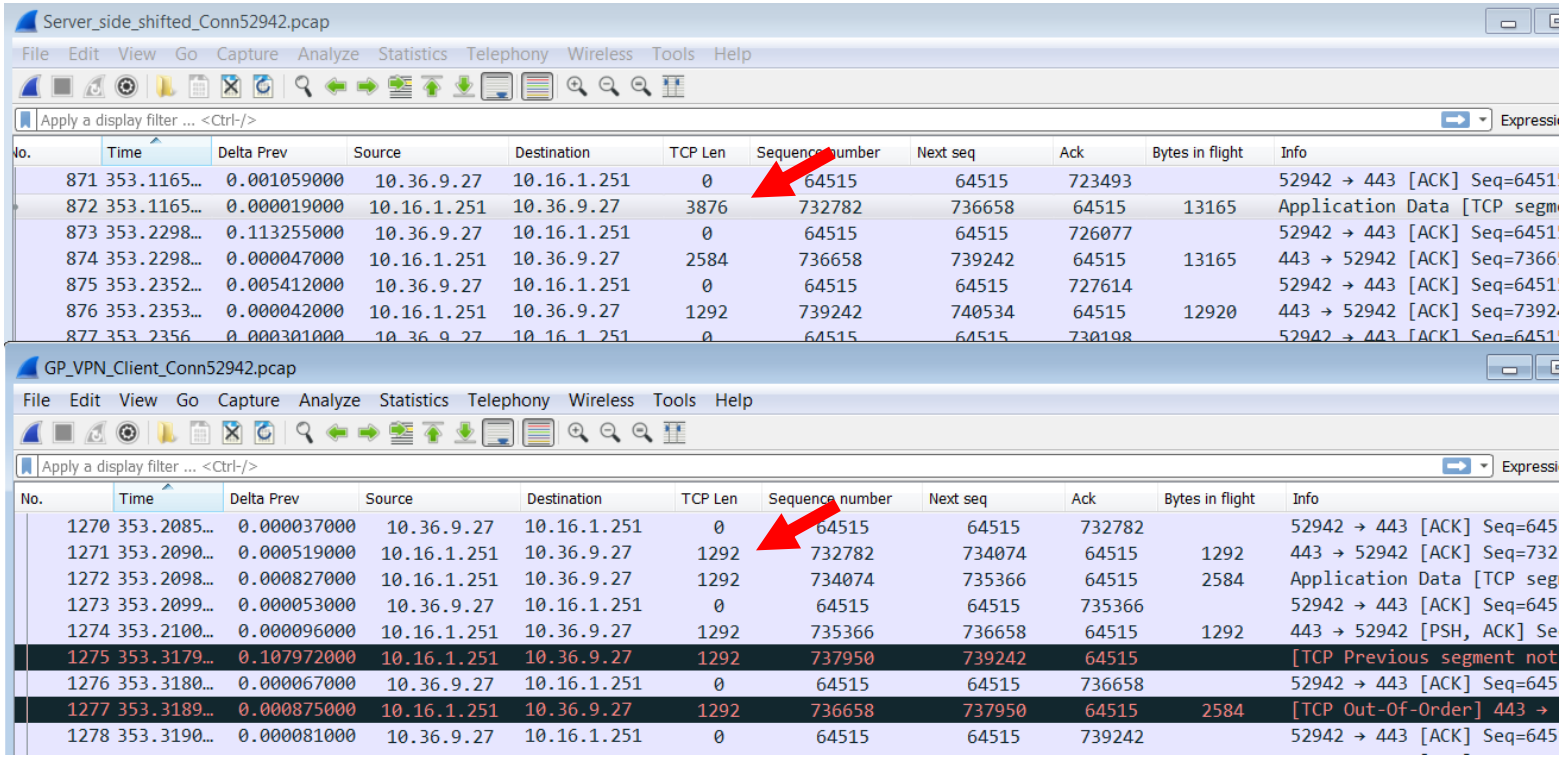
Comparison Storyboard



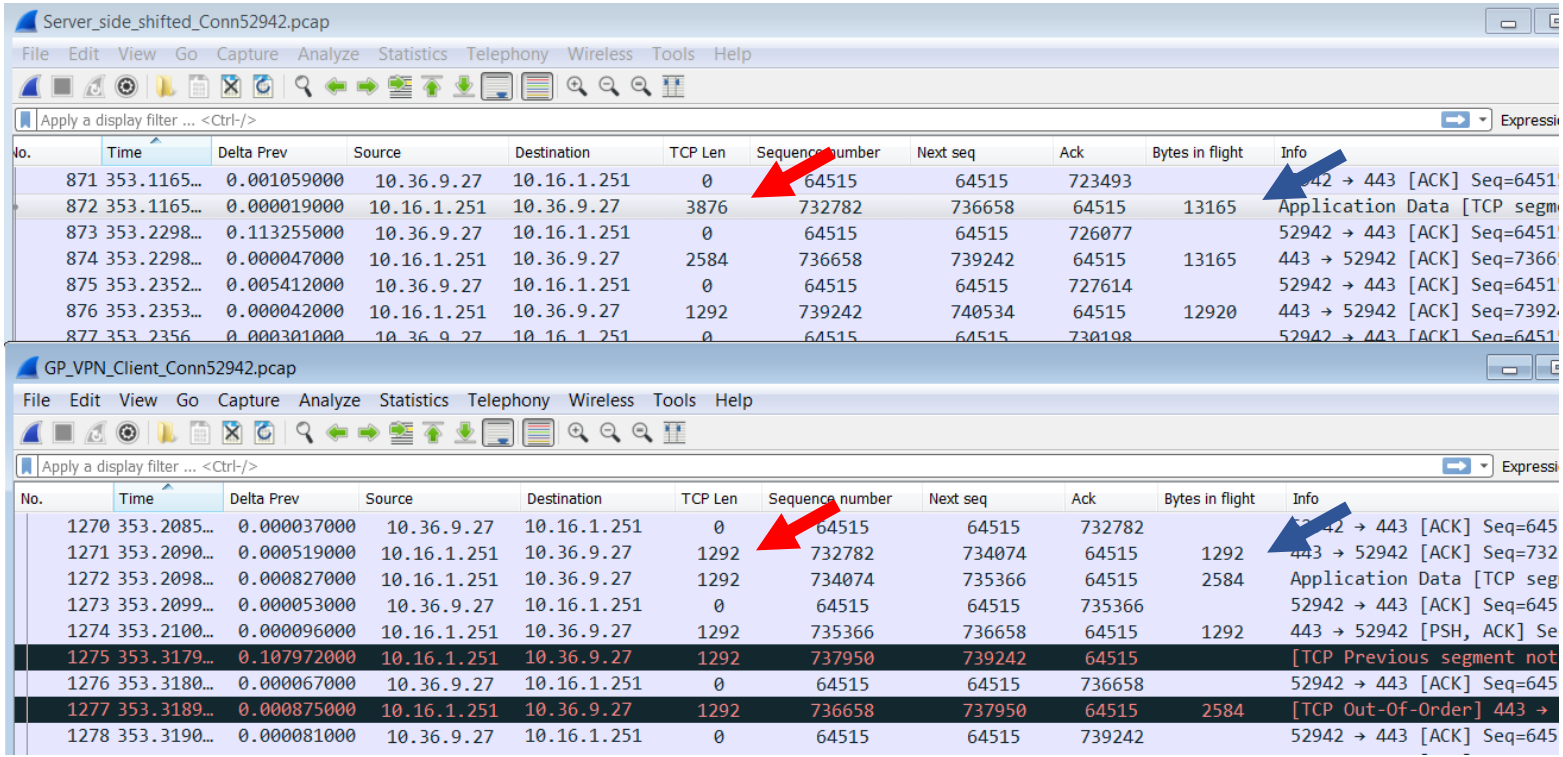
Comparison Storyboard



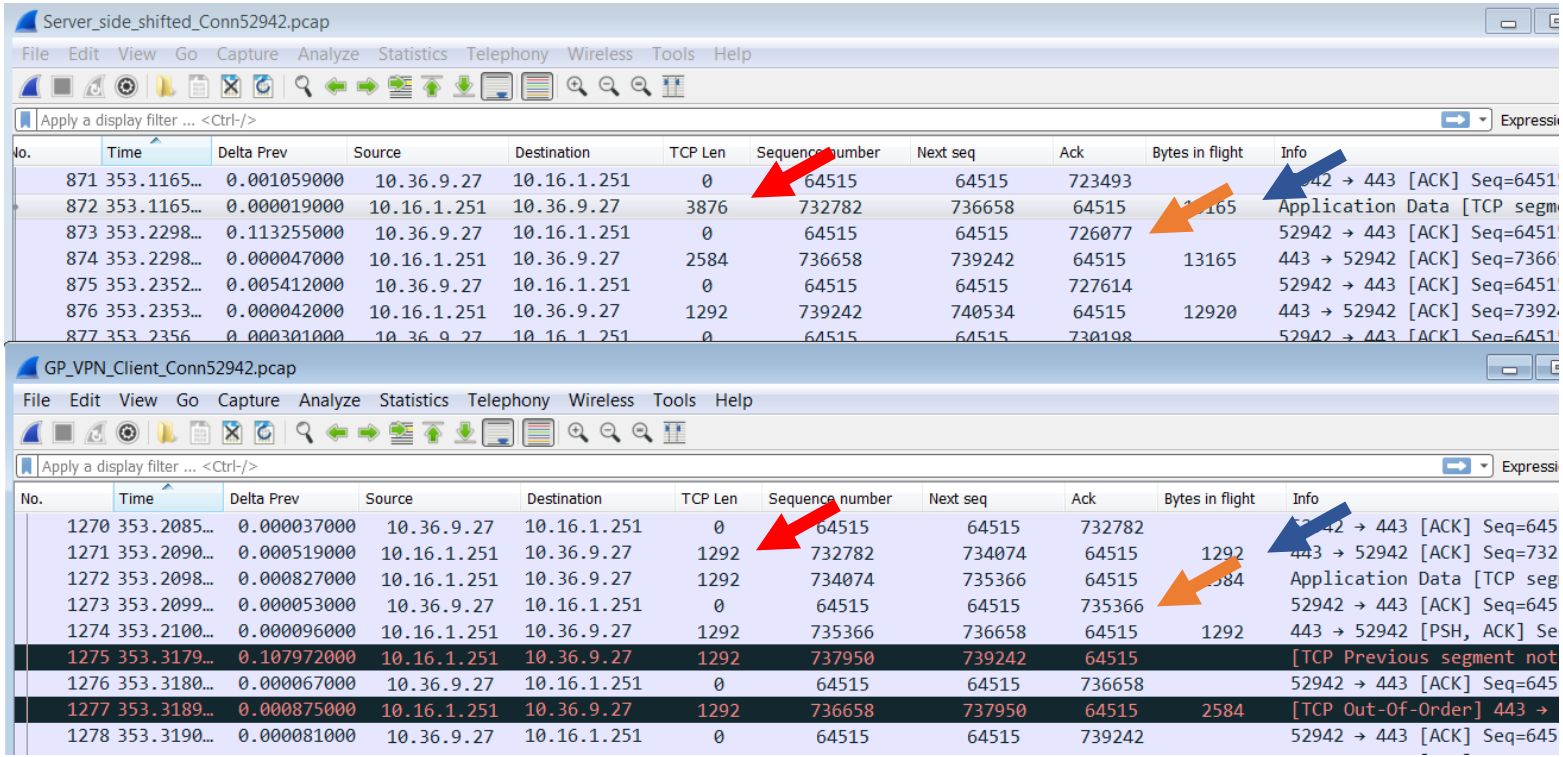
Comparison Storyboard



Comparison Storyboard



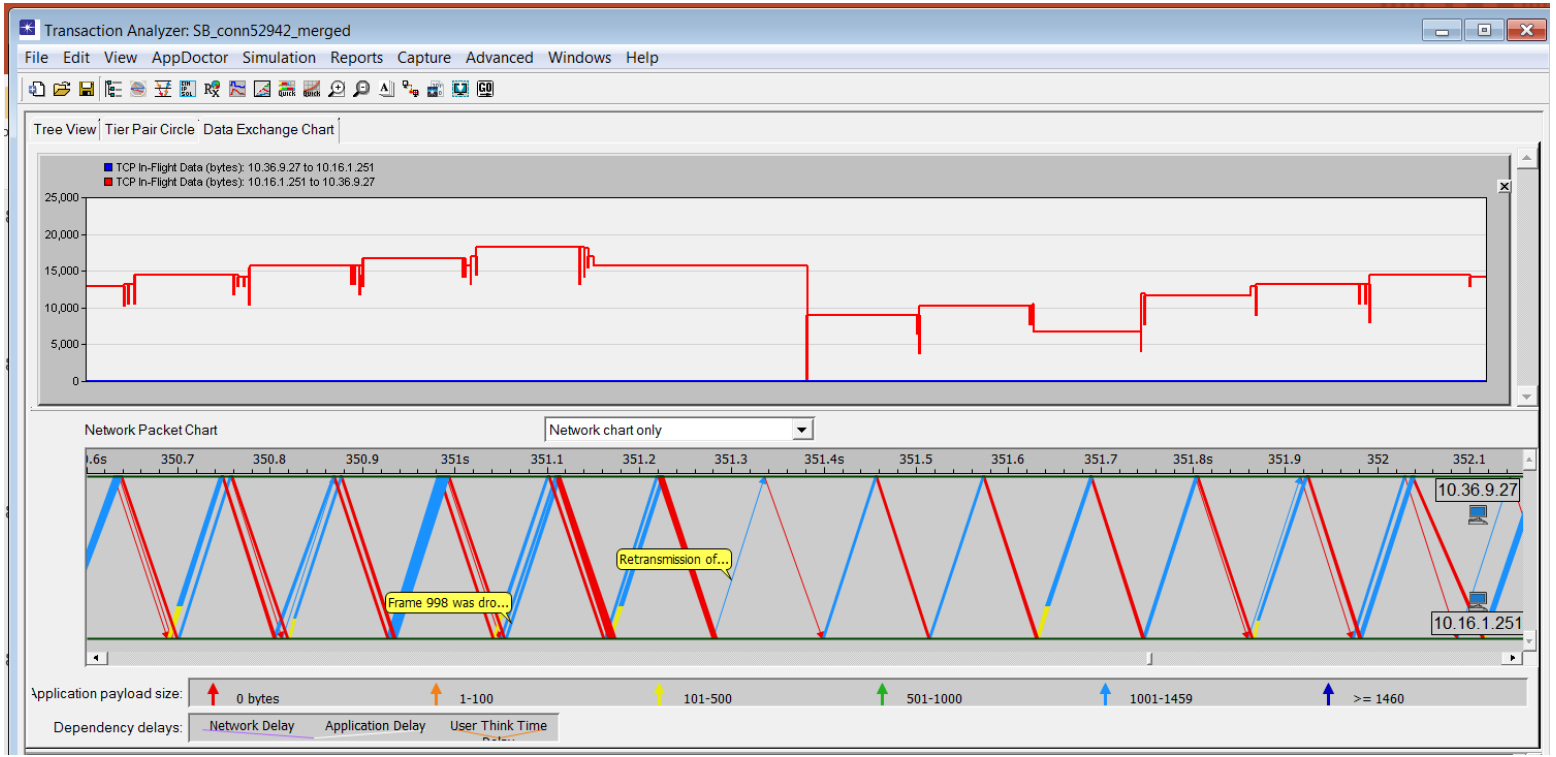
Comparison Storyboard



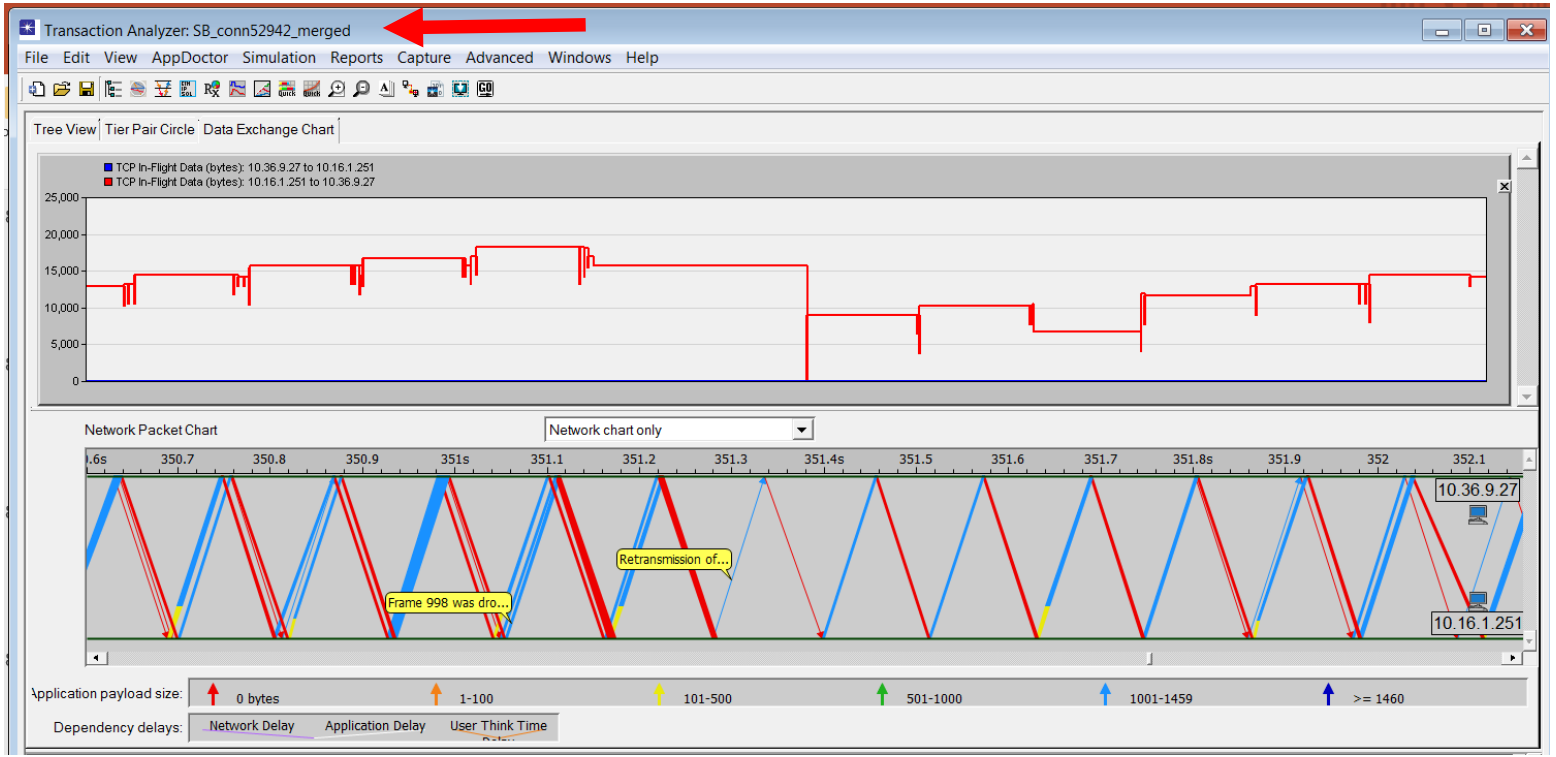
We need to visualize this…

- Now that we've seen how Wireshark counts bytes in flight…
- …and some of the challenges of side by side comparisons…
- …let's look at how we can gain better insight from visualization

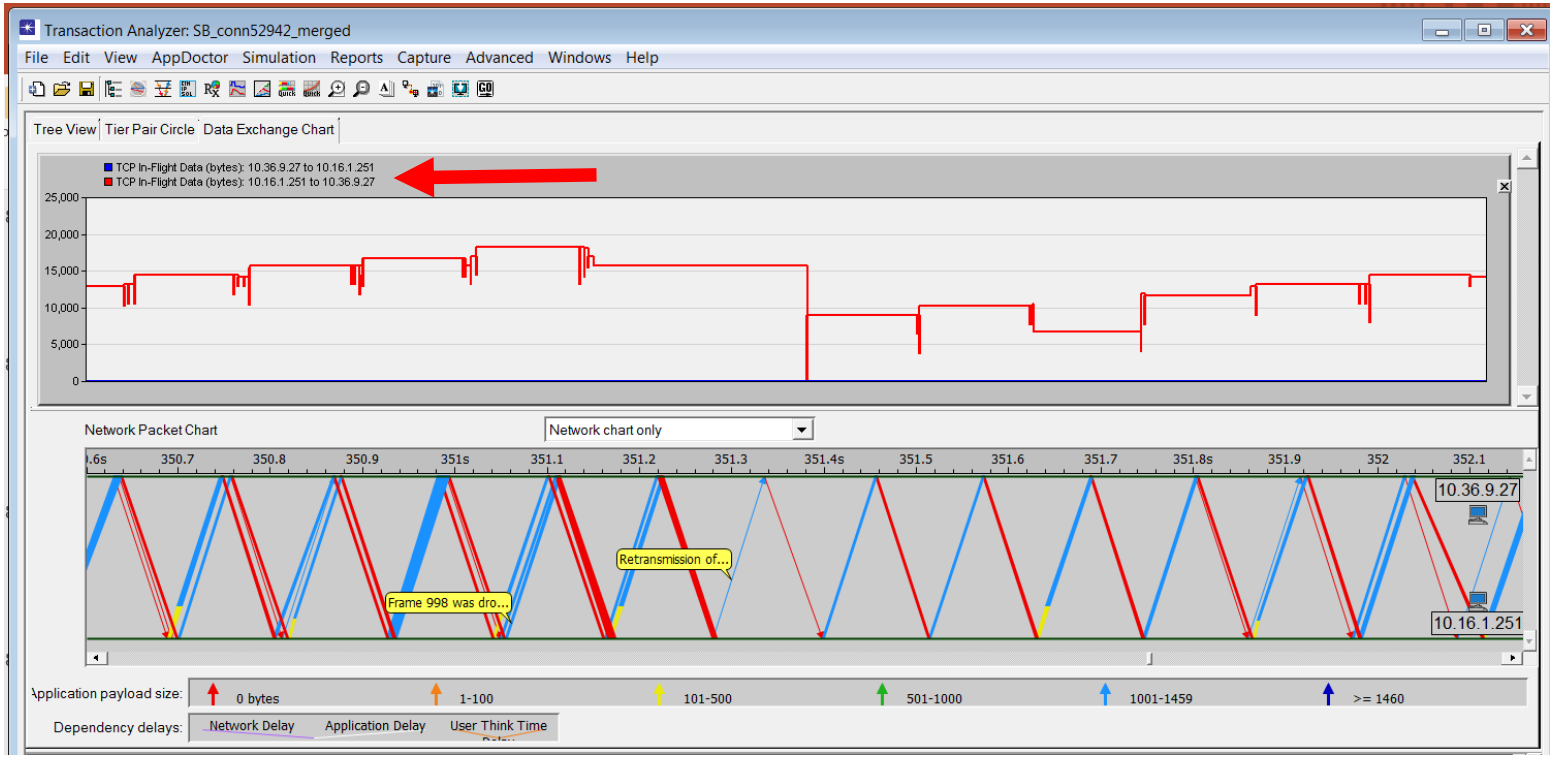
Packet Exchange with BIF Overlay



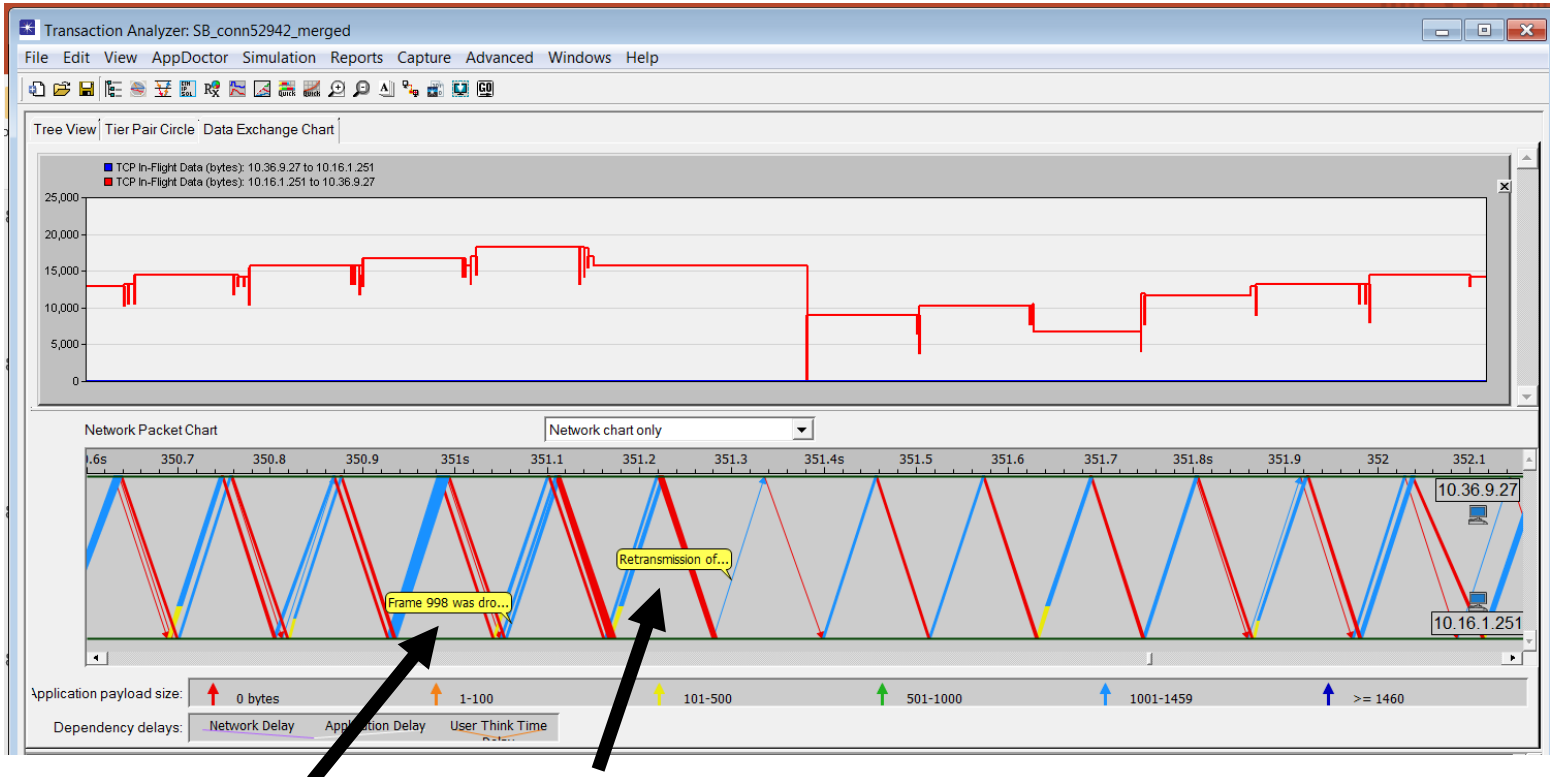
Quick Orientation



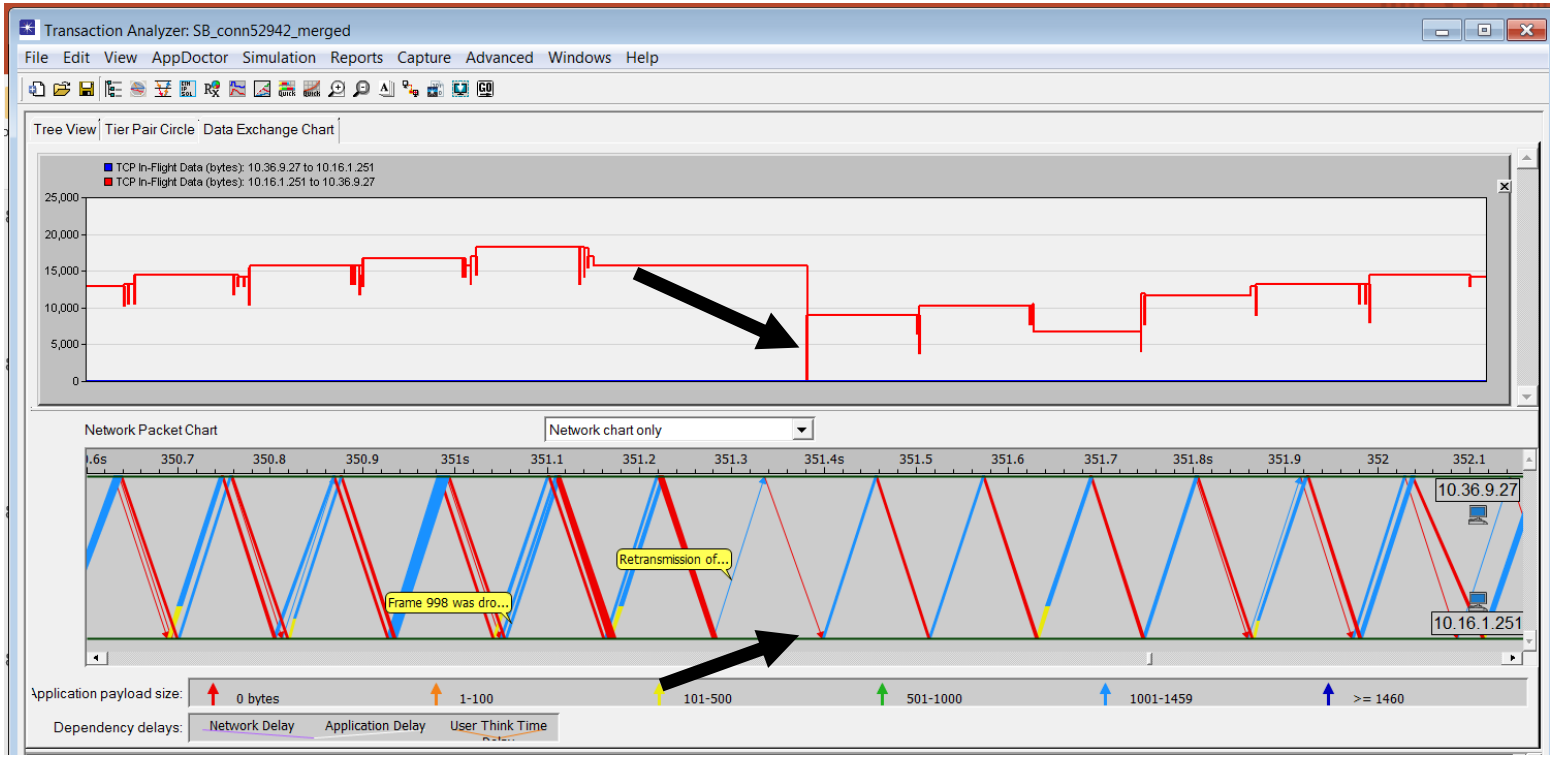
In-flight Bytes



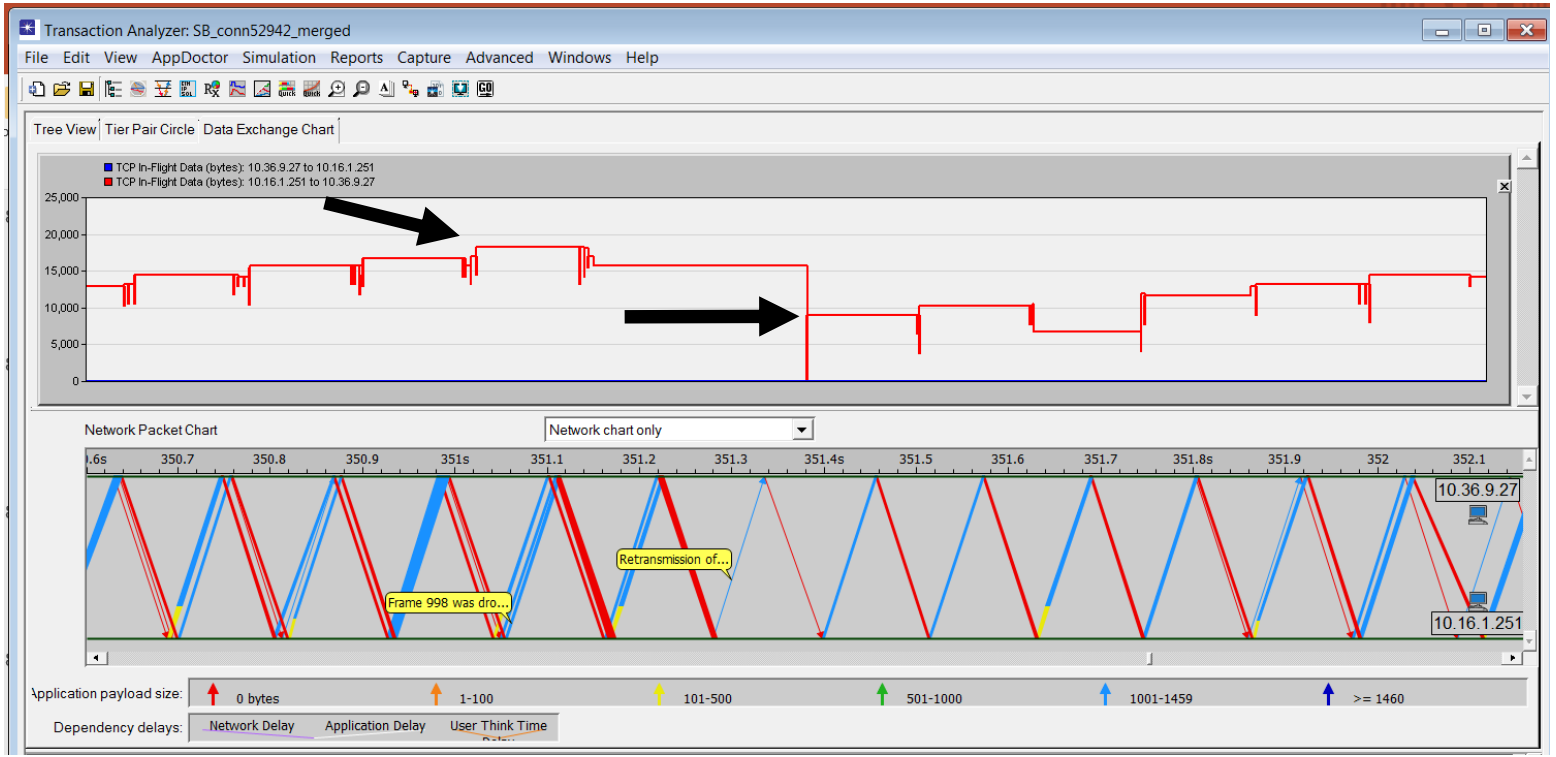
Symptom Pop-up Labels



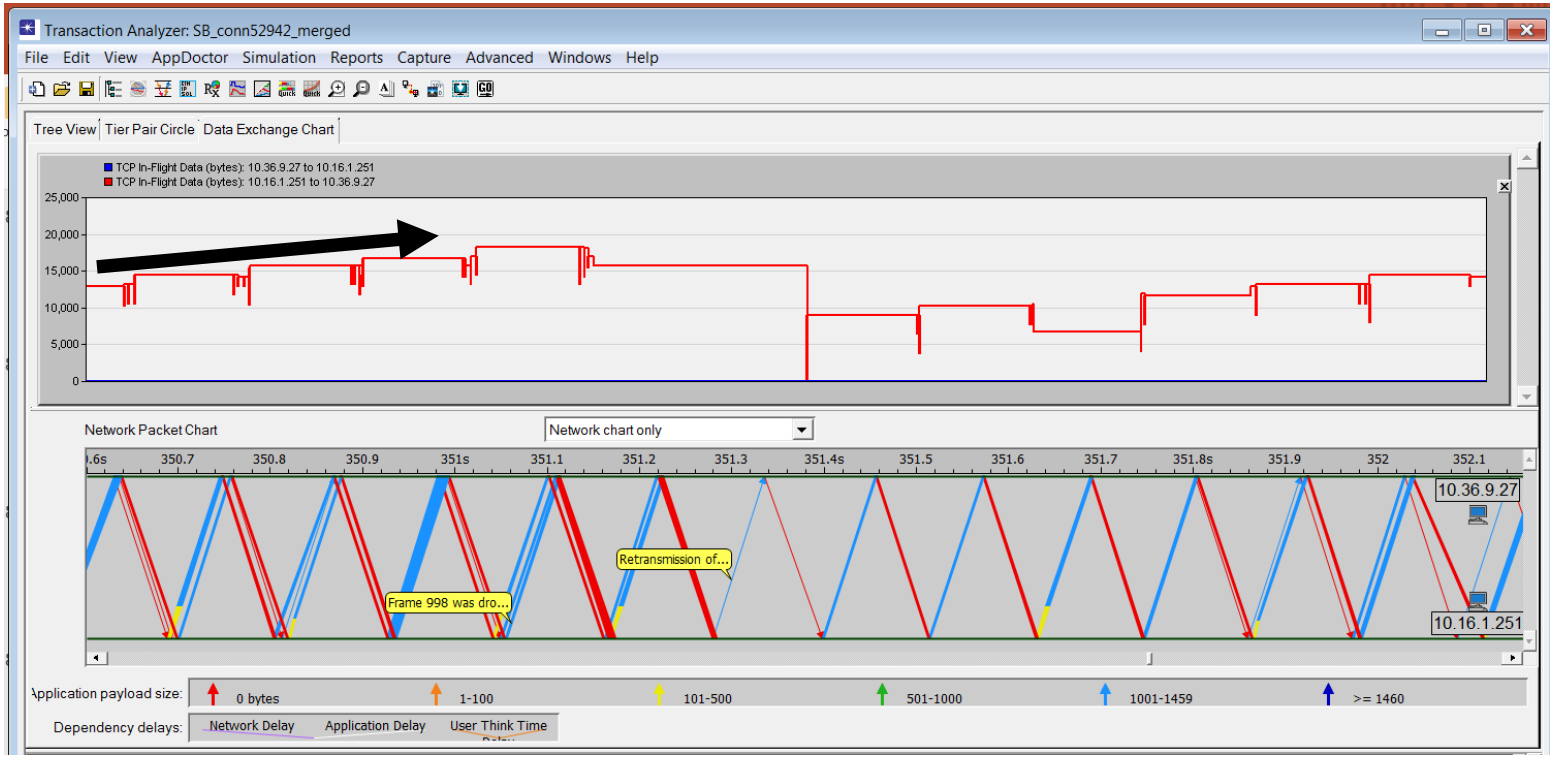
Why In-flight drops to 0?



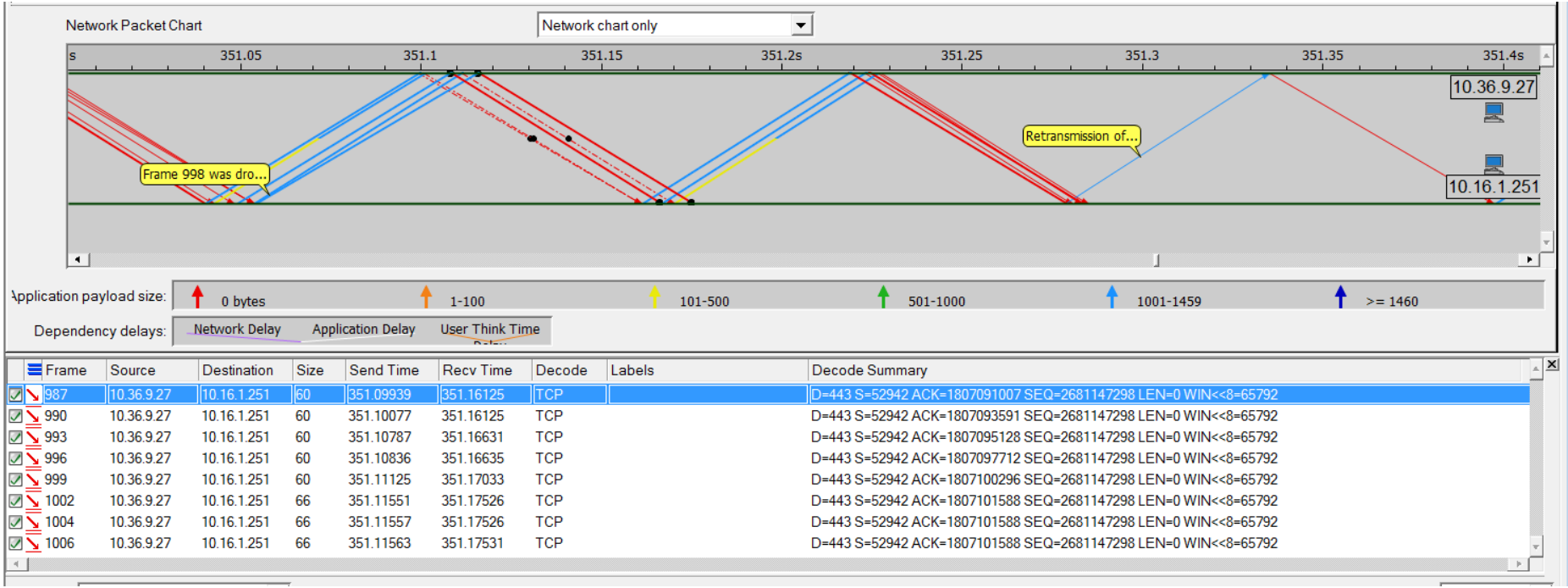
What does this delta tell us?



What does this upward progression tell us?

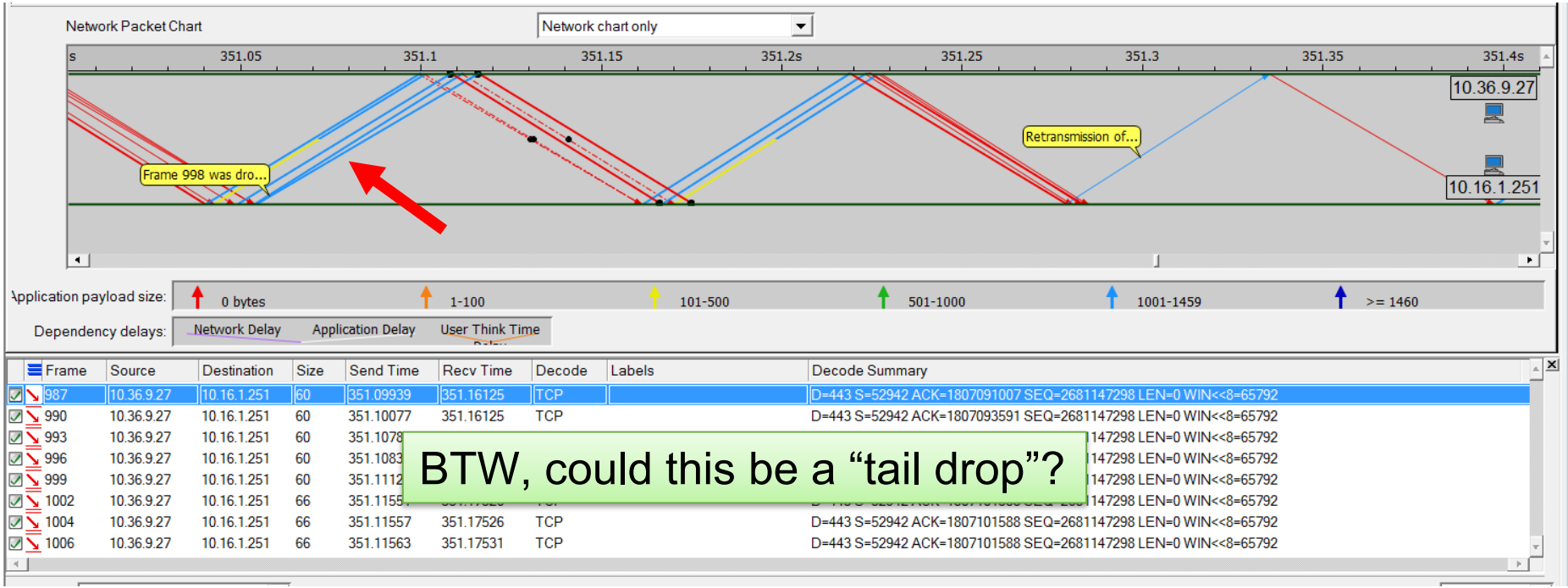


When does Server hear the news?



Frame	Source	Destination	Size	Send Time	Recv Time	Decode	Labels	Decode Summary
987	10.36.9.27	10.16.1.251	60	351.09939	351.16125	TCP		D=443 S=52942 ACK=1807091007 SEQ=2681147298 LEN=0 WIN<<8=65792
990	10.36.9.27	10.16.1.251	60	351.10077	351.16125	TCP		D=443 S=52942 ACK=1807093591 SEQ=2681147298 LEN=0 WIN<<8=65792
993	10.36.9.27	10.16.1.251	60	351.10787	351.16631	TCP		D=443 S=52942 ACK=1807095128 SEQ=2681147298 LEN=0 WIN<<8=65792
996	10.36.9.27	10.16.1.251	60	351.10836	351.16635	TCP		D=443 S=52942 ACK=1807097712 SEQ=2681147298 LEN=0 WIN<<8=65792
999	10.36.9.27	10.16.1.251	60	351.11125	351.17033	TCP		D=443 S=52942 ACK=1807100296 SEQ=2681147298 LEN=0 WIN<<8=65792
1002	10.36.9.27	10.16.1.251	66	351.11551	351.17526	TCP		D=443 S=52942 ACK=1807101588 SEQ=2681147298 LEN=0 WIN<<8=65792
1004	10.36.9.27	10.16.1.251	66	351.11557	351.17526	TCP		D=443 S=52942 ACK=1807101588 SEQ=2681147298 LEN=0 WIN<<8=65792
1006	10.36.9.27	10.16.1.251	66	351.11563	351.17531	TCP		D=443 S=52942 ACK=1807101588 SEQ=2681147298 LEN=0 WIN<<8=65792

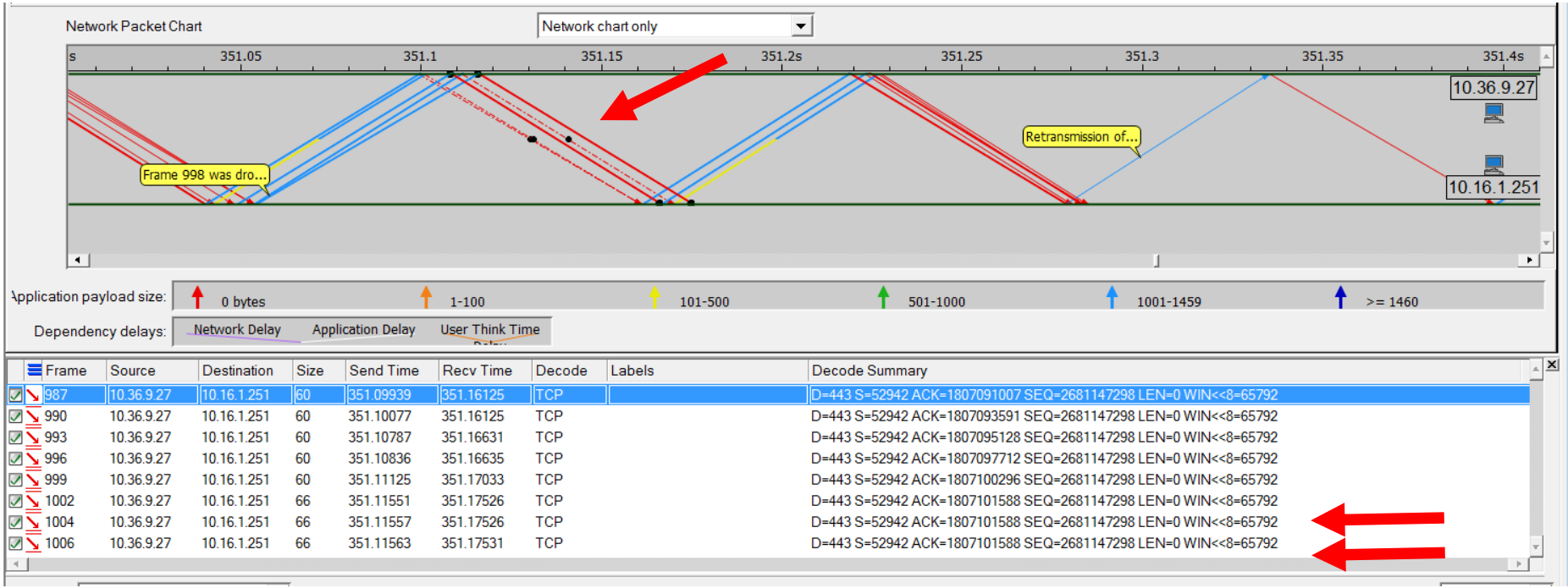
#998 is dropped



Frame	Source	Destination	Size	Send Time	Recv Time	Decode	Labels	Decode Summary
987	10.36.9.27	10.16.1.251	60	351.09939	351.16125	TCP		D=443 S=52942 ACK=1807091007 SEQ=2681147298 LEN=0 WIN<<8=65792
990	10.36.9.27	10.16.1.251	60	351.10077	351.16125	TCP		D=443 S=52942 ACK=1807093591 SEQ=2681147298 LEN=0 WIN<<8=65792
993	10.36.9.27	10.16.1.251	60	351.1078				147298 LEN=0 WIN<<8=65792
996	10.36.9.27	10.16.1.251	60	351.1083				147298 LEN=0 WIN<<8=65792
999	10.36.9.27	10.16.1.251	60	351.1112				147298 LEN=0 WIN<<8=65792
1002	10.36.9.27	10.16.1.251	66	351.1155				147298 LEN=0 WIN<<8=65792
1004	10.36.9.27	10.16.1.251	66	351.11557	351.17526	TCP		D=443 S=52942 ACK=1807101588 SEQ=2681147298 LEN=0 WIN<<8=65792
1006	10.36.9.27	10.16.1.251	66	351.11563	351.17531	TCP		D=443 S=52942 ACK=1807101588 SEQ=2681147298 LEN=0 WIN<<8=65792

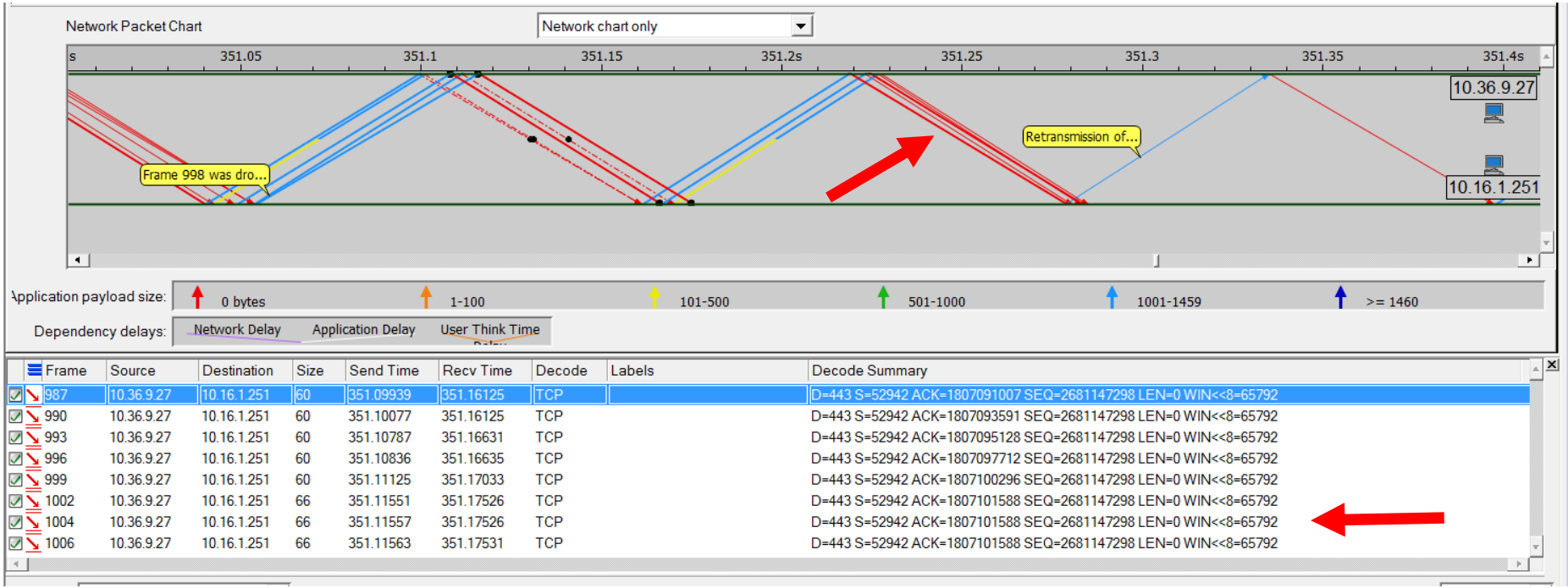
BTW, could this be a “tail drop”?

2 of the DupACKs



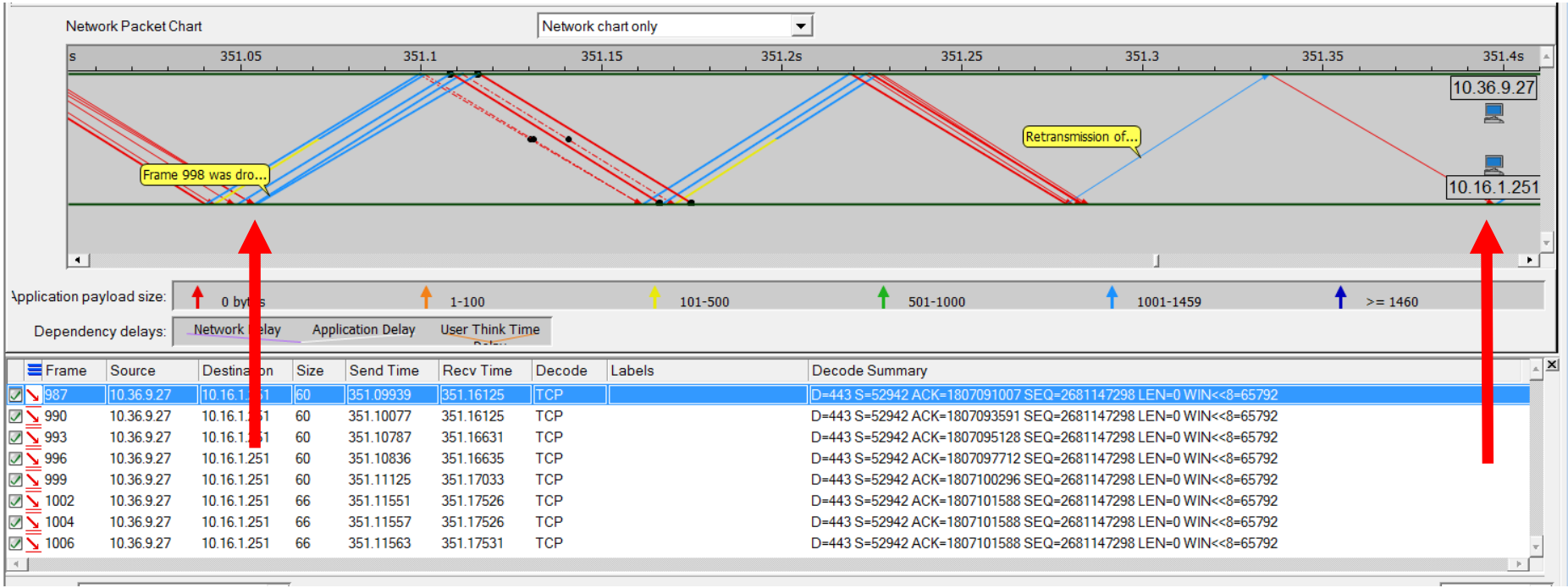
Frame	Source	Destination	Size	Send Time	Recv Time	Decode	Labels	Decode Summary
987	10.36.9.27	10.16.1.251	60	351.09939	351.16125	TCP		D=443 S=52942 ACK=1807091007 SEQ=2681147298 LEN=0 WIN<<8=65792
990	10.36.9.27	10.16.1.251	60	351.10077	351.16125	TCP		D=443 S=52942 ACK=1807093591 SEQ=2681147298 LEN=0 WIN<<8=65792
993	10.36.9.27	10.16.1.251	60	351.10787	351.16631	TCP		D=443 S=52942 ACK=1807095128 SEQ=2681147298 LEN=0 WIN<<8=65792
996	10.36.9.27	10.16.1.251	60	351.10836	351.16635	TCP		D=443 S=52942 ACK=1807097712 SEQ=2681147298 LEN=0 WIN<<8=65792
999	10.36.9.27	10.16.1.251	60	351.11125	351.17033	TCP		D=443 S=52942 ACK=1807100296 SEQ=2681147298 LEN=0 WIN<<8=65792
1002	10.36.9.27	10.16.1.251	66	351.11551	351.17526	TCP		D=443 S=52942 ACK=1807101588 SEQ=2681147298 LEN=0 WIN<<8=65792
1004	10.36.9.27	10.16.1.251	66	351.11557	351.17526	TCP		D=443 S=52942 ACK=1807101588 SEQ=2681147298 LEN=0 WIN<<8=65792
1006	10.36.9.27	10.16.1.251	66	351.11563	351.17531	TCP		D=443 S=52942 ACK=1807101588 SEQ=2681147298 LEN=0 WIN<<8=65792

3 of 3 DupACKs



Frame	Source	Destination	Size	Send Time	Recv Time	Decode	Labels	Decode Summary
987	10.36.9.27	10.16.1.251	60	351.09939	351.16125	TCP		D=443 S=52942 ACK=1807091007 SEQ=2681147298 LEN=0 WIN<<8=65792
990	10.36.9.27	10.16.1.251	60	351.10077	351.16125	TCP		D=443 S=52942 ACK=1807093591 SEQ=2681147298 LEN=0 WIN<<8=65792
993	10.36.9.27	10.16.1.251	60	351.10787	351.16631	TCP		D=443 S=52942 ACK=1807095128 SEQ=2681147298 LEN=0 WIN<<8=65792
996	10.36.9.27	10.16.1.251	60	351.10836	351.16635	TCP		D=443 S=52942 ACK=1807097712 SEQ=2681147298 LEN=0 WIN<<8=65792
999	10.36.9.27	10.16.1.251	60	351.11125	351.17033	TCP		D=443 S=52942 ACK=1807100296 SEQ=2681147298 LEN=0 WIN<<8=65792
1002	10.36.9.27	10.16.1.251	66	351.11551	351.17526	TCP		D=443 S=52942 ACK=1807101588 SEQ=2681147298 LEN=0 WIN<<8=65792
1004	10.36.9.27	10.16.1.251	66	351.11557	351.17526	TCP		D=443 S=52942 ACK=1807101588 SEQ=2681147298 LEN=0 WIN<<8=65792
1006	10.36.9.27	10.16.1.251	66	351.11563	351.17531	TCP		D=443 S=52942 ACK=1807101588 SEQ=2681147298 LEN=0 WIN<<8=65792

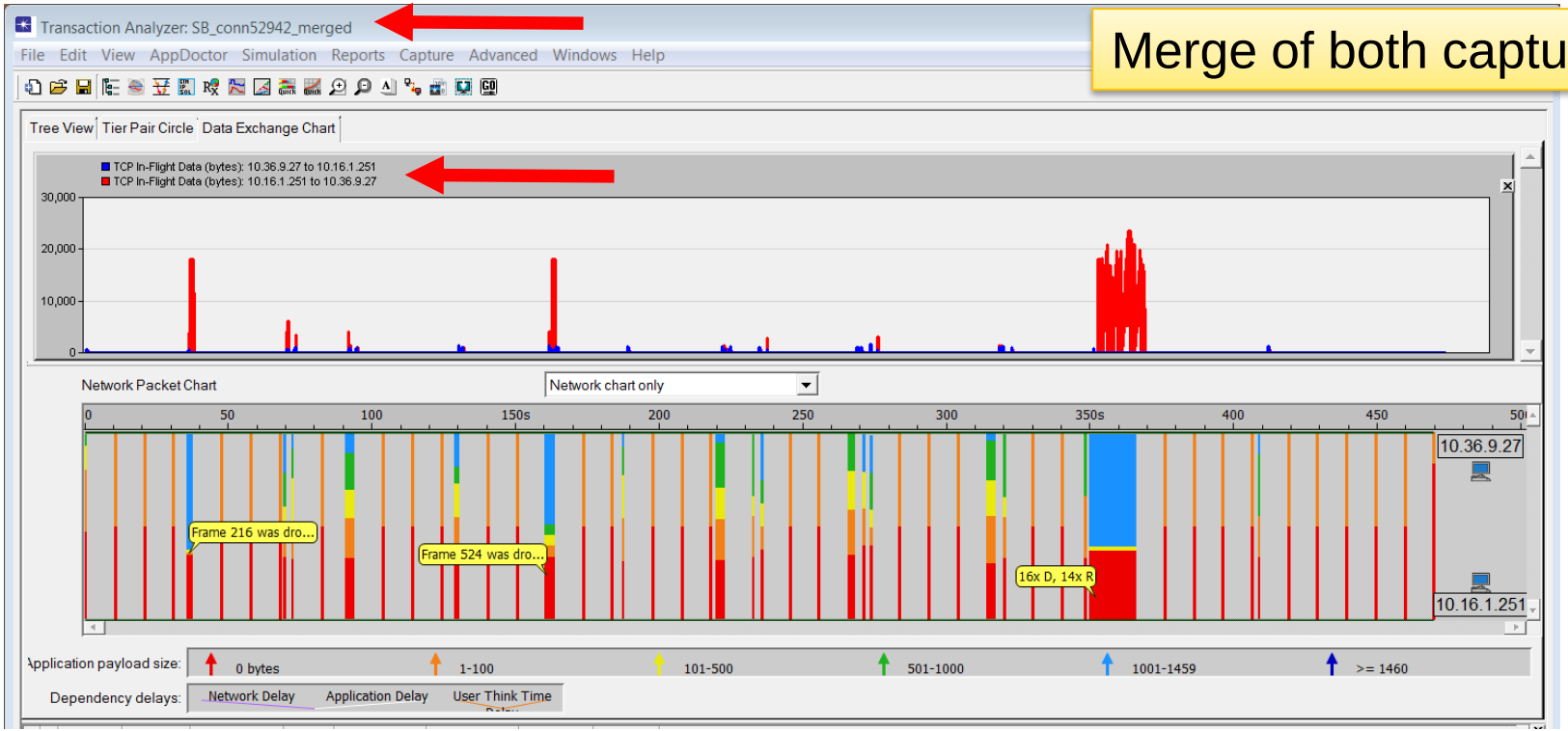
Cost of Drop: # of RTTs?



Frame	Source	Destination	Size	Send Time	Recv Time	Decode	Labels	Decode Summary
987	10.36.9.27	10.16.1.251	60	351.09939	351.16125	TCP		D=443 S=52942 ACK=1807091007 SEQ=2681147298 LEN=0 WIN<<8=65792
990	10.36.9.27	10.16.1.251	60	351.10077	351.16125	TCP		D=443 S=52942 ACK=1807093591 SEQ=2681147298 LEN=0 WIN<<8=65792
993	10.36.9.27	10.16.1.251	60	351.10787	351.16631	TCP		D=443 S=52942 ACK=1807095128 SEQ=2681147298 LEN=0 WIN<<8=65792
996	10.36.9.27	10.16.1.251	60	351.10836	351.16635	TCP		D=443 S=52942 ACK=1807097712 SEQ=2681147298 LEN=0 WIN<<8=65792
999	10.36.9.27	10.16.1.251	60	351.11125	351.17033	TCP		D=443 S=52942 ACK=1807100296 SEQ=2681147298 LEN=0 WIN<<8=65792
1002	10.36.9.27	10.16.1.251	66	351.11551	351.17526	TCP		D=443 S=52942 ACK=1807101588 SEQ=2681147298 LEN=0 WIN<<8=65792
1004	10.36.9.27	10.16.1.251	66	351.11557	351.17526	TCP		D=443 S=52942 ACK=1807101588 SEQ=2681147298 LEN=0 WIN<<8=65792
1006	10.36.9.27	10.16.1.251	66	351.11563	351.17531	TCP		D=443 S=52942 ACK=1807101588 SEQ=2681147298 LEN=0 WIN<<8=65792

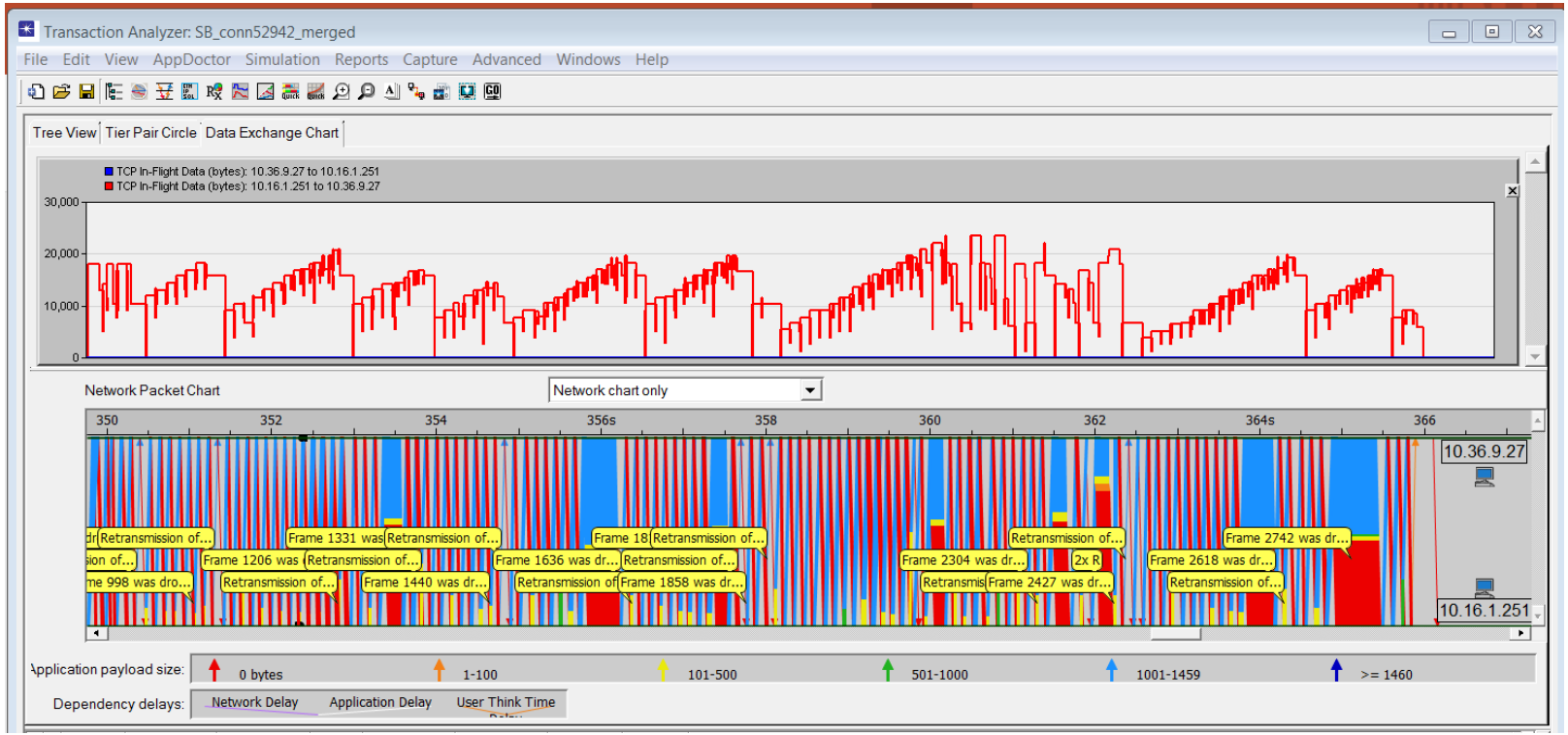
Discussion

Let's Visualize the Entire Connection

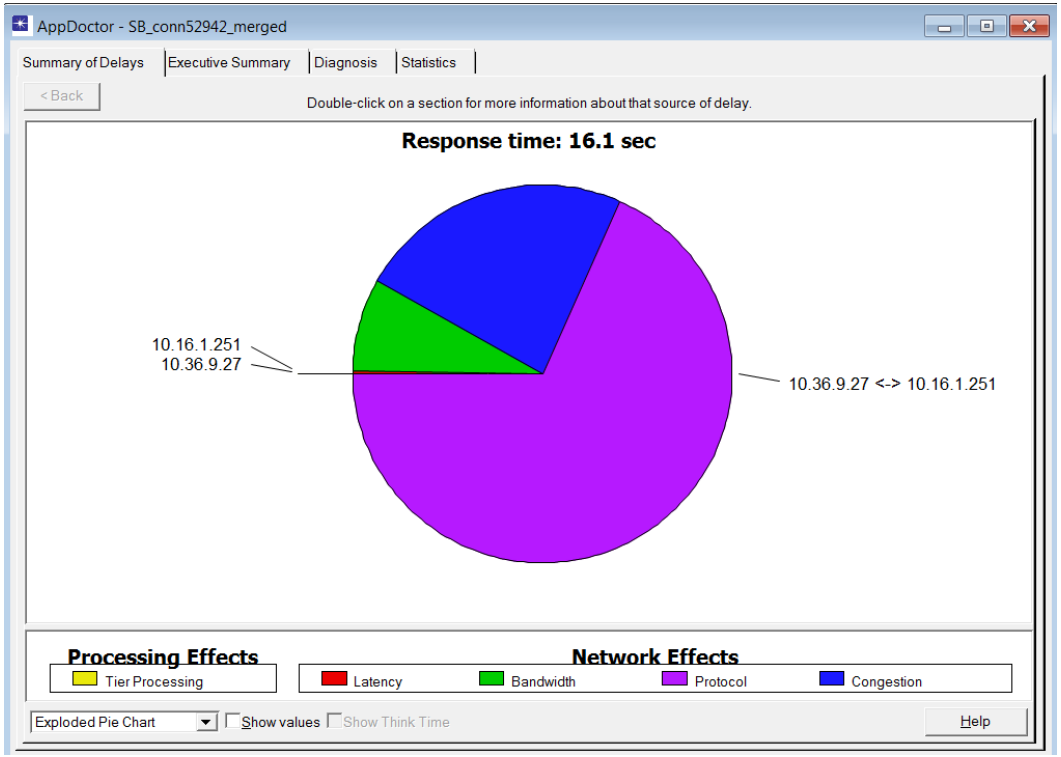


Merge of both captures

...drill into the 1.4MB Xfer



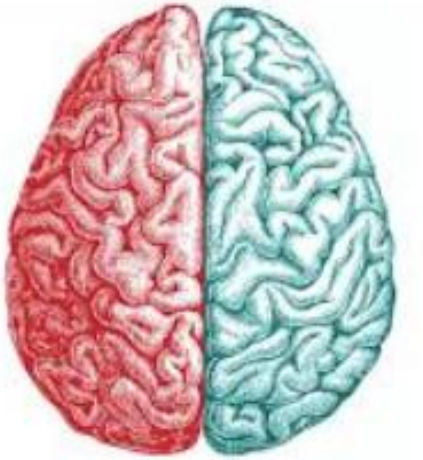
Delay Analysis during Xfer



Final Discussion, Questions, Comments

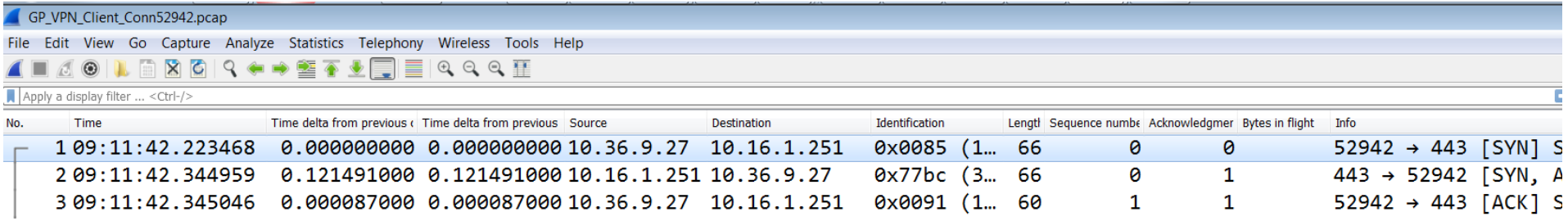
Split Brain Comparisons

- Latency and Congestion Review

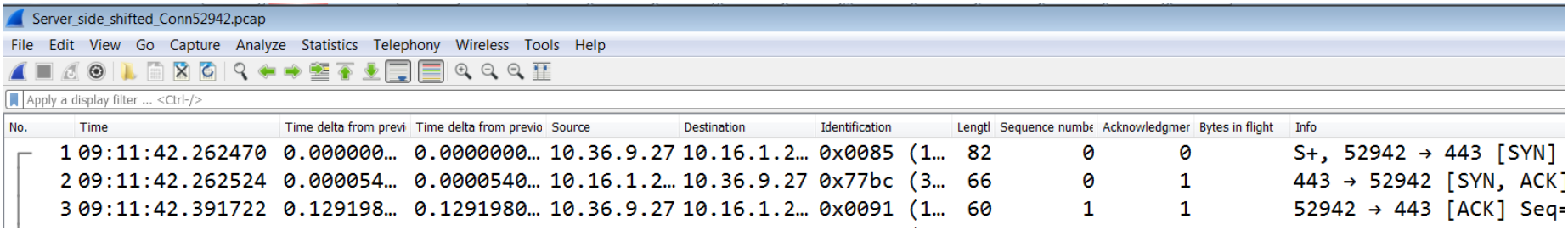


Remember this?

- Client Capture – 121ms



- Server Capture – 129ms



Group Discussion

- What's the definition of latency?
- What's the definition of congestion?
- How can you tell latency from congestion?

One Possible Formula

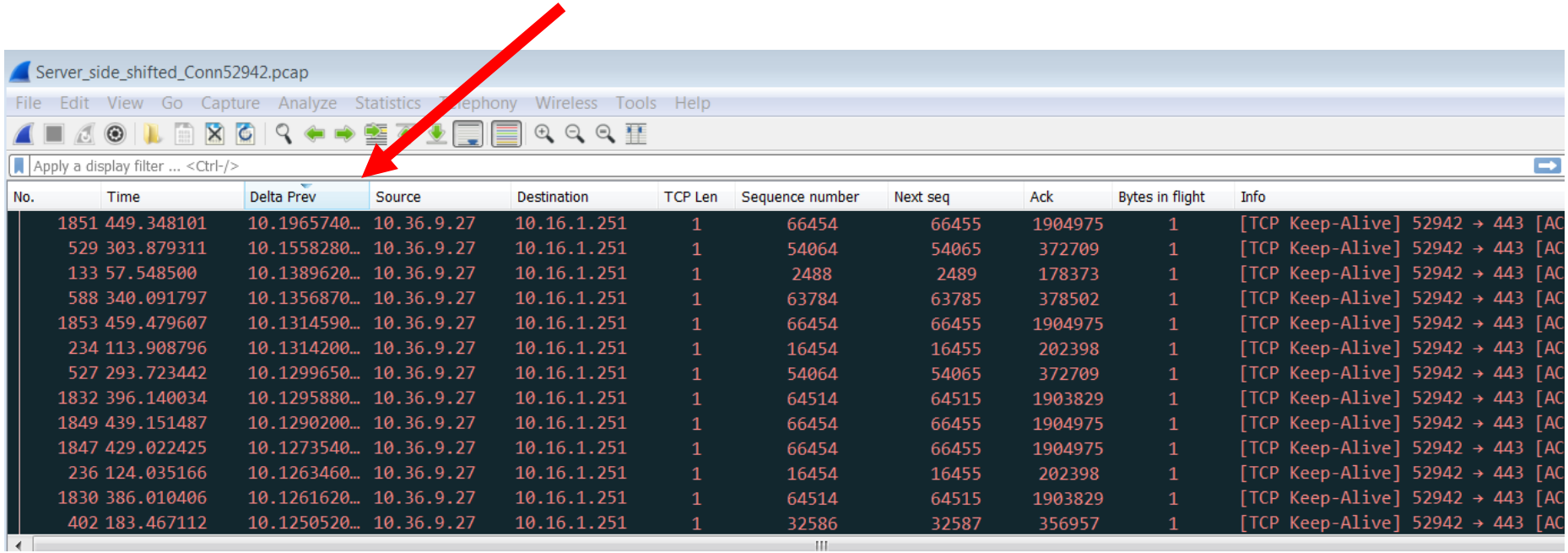
- Total Delay (TD) minus
- (Bandwidth delay + Latency Delay + Protocol Delay)

- TD – (BD + LD + PD) == Congestion

Finding congestion

- How can we find examples of high latency (congestion) in the capture?
- High “prev captured” delta *and* 0 payload
  - Payload > 0 could be used if we find a “piggy-back” ACK

Sort by Delta Prev



Scroll Down to TCP Len ==0

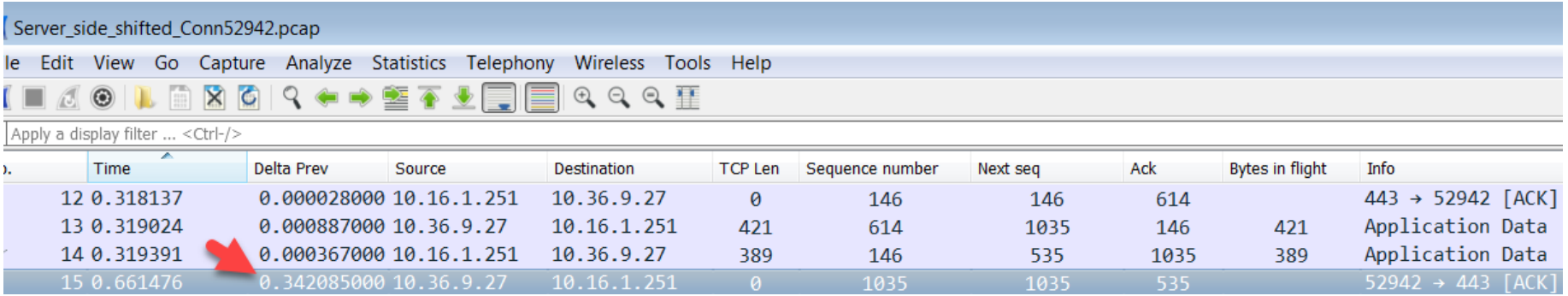
Server_side_shifted_Conn52942.pcap

File Edit View Go Capture Analyze Statistics Telephony Wireless Tools Help

Apply a display filter ... <Ctrl-/>

No.	Time	Delta Prev	Source	Destination	TCP Len	Sequence number	Next seq	Ack	Bytes in flight	Info
243	129.858807	0.912091000	10.16.1.251	10.36.9.27	453	202398	202851	17937	453	Application Data
485	266.005525	0.465380000	10.36.9.27	10.16.1.251	37	45617	45654	366285	37	Application Data
503	267.703699	0.415038000	10.36.9.27	10.16.1.251	37	49753	49790	368097	37	Application Data
15	0.661476	0.342085000	10.36.9.27	10.16.1.251	0	1035	1035	535		52942 → 443 [ACK] Seq=1035
1844	409.110547	0.334864000	10.36.9.27	10.16.1.251	0	66455	66455	1904975		52942 → 443 [ACK] Seq=66455
576	317.047693	0.332477000	10.36.9.27	10.16.1.251	0	61717	61717	377692		52942 → 443 [ACK] Seq=61717
248	130.359245	0.331970000	10.36.9.27	10.16.1.251	0	18971	18971	203944		52942 → 443 [ACK] Seq=18971
472	233.001809	0.329109000	10.36.9.27	10.16.1.251	0	45031	45031	363432		52942 → 443 [ACK] Seq=45031
427	220.077289	0.327739000	10.36.9.27	10.16.1.251	0	37139	37139	358625		52942 → 443 [ACK] Seq=37139
416	187.728667	0.327603000	10.36.9.27	10.16.1.251	0	35849	35849	358124		52942 → 443 [ACK] Seq=35849
163	70.246969	0.325587000	10.36.9.27	10.16.1.251	0	5499	5499	189187		52942 → 443 [ACK] Seq=5499
399	163.546123	0.325216000	10.36.9.27	10.16.1.251	0	32587	32587	356957		52942 → 443 [ACK] Seq=32587
489	266.335429	0.323936000	10.36.9.27	10.16.1.251	0	46651	46651	366738		52942 → 443 [ACK] Seq=46651

Server Cap Shows 342ms RTT



	Time	Delta Prev	Source	Destination	TCP Len	Sequence number	Next seq	Ack	Bytes in flight	Info
12	0.318137	0.000028000	10.16.1.251	10.36.9.27	0	146	146	614		443 → 52942 [ACK]
13	0.319024	0.000887000	10.36.9.27	10.16.1.251	421	614	1035	146	421	Application Data
14	0.319391	0.000367000	10.16.1.251	10.36.9.27	389	146	535	1035	389	Application Data
15	0.661476	0.342085000	10.36.9.27	10.16.1.251	0	1035	1035	535		52942 → 443 [ACK]

- Could this be right? We just established RTT is 120-ish ms
- What is interesting about this ACK?

Client shows 209ms delta?

Server_side_shifted_Conn52942.pcap

Edit View Go Capture Analyze Statistics Telephony Wireless Tools Help

Apply a display filter ... <Ctrl-/>

	Time	Delta Prev	Source	Destination	TCP Len	Sequence number	Next seq	Ack	Bytes in flight	Info
12	0.318137	0.000028000	10.16.1.251	10.36.9.27	0	146	146	614		443 → 52942 [ACK]
13	0.319024	0.000887000	10.36.9.27	10.16.1.251	421	614	1035	146	421	Application Data
14	0.319391	0.000367000	10.16.1.251	10.36.9.27	389	146	535	1035	389	Application Data
15	0.661476	0.342085000	10.36.9.27	10.16.1.251	0	1035	1035	535		52942 → 443 [ACK]

GP_VPN_Client_Conn52942.pcap

File Edit View Go Capture Analyze Statistics Telephony Wireless Tools Help

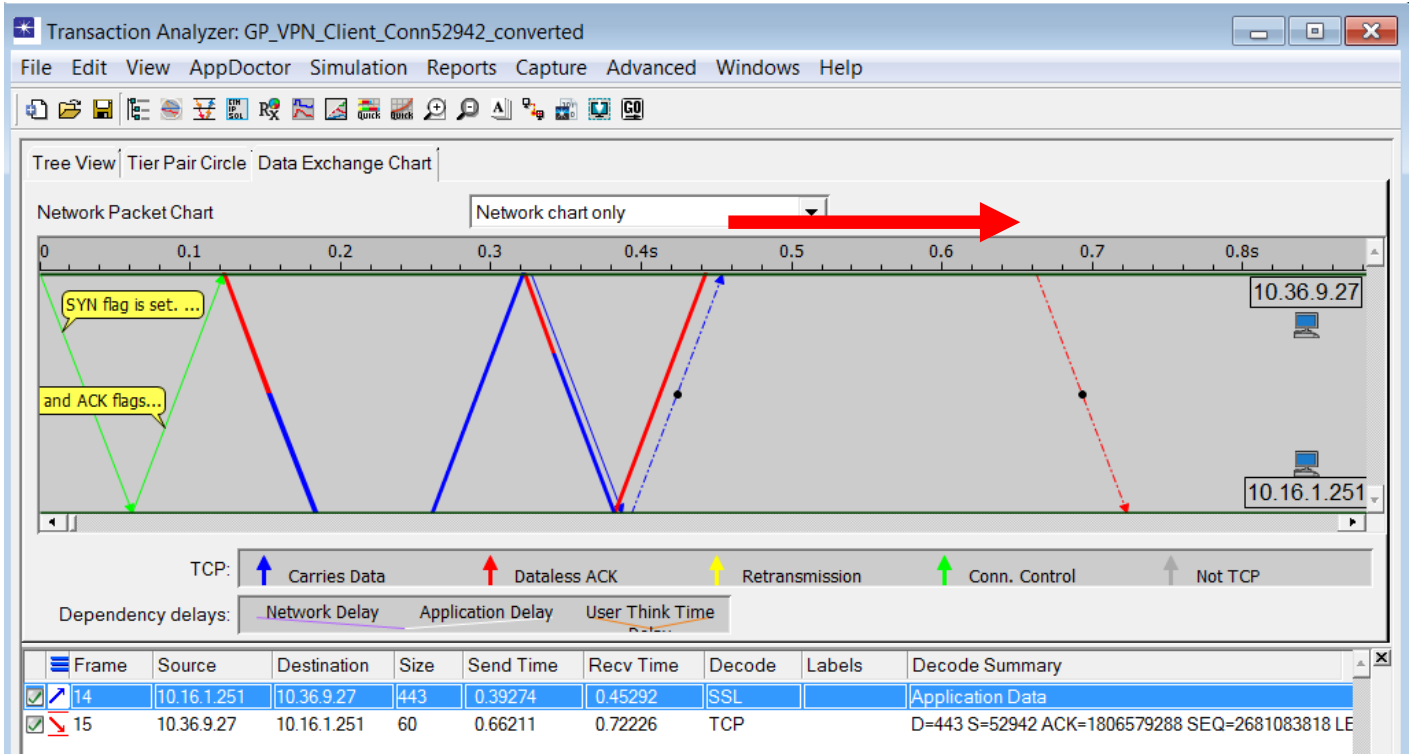
Apply a display filter ... <Ctrl-/>

No.	Time	Delta Prev	Source	Destination	TCP Len	Sequence number	Next seq	Ack	Bytes in flight	Info
10	0.321933	0.000216000	10.36.9.27	10.16.1.251	37	577	614	146	96	Application Data
11	0.326855	0.004922000	10.36.9.27	10.16.1.251	421	614	1035	146	517	Application Data
12	0.442252	0.115397000	10.16.1.251	10.36.9.27	0	146	146	518		[TCP Dup ACK 5#1]
13	0.442287	0.000035000	10.16.1.251	10.36.9.27	0	146	146	614		443 → 52942 [ACK]
14	0.452920	0.010633000	10.16.1.251	10.36.9.27	389	146	535	1035	389	Application Data
15	0.662106	0.209186000	10.36.9.27	10.16.1.251	0	1035	1035	535		52942 → 443 [ACK]

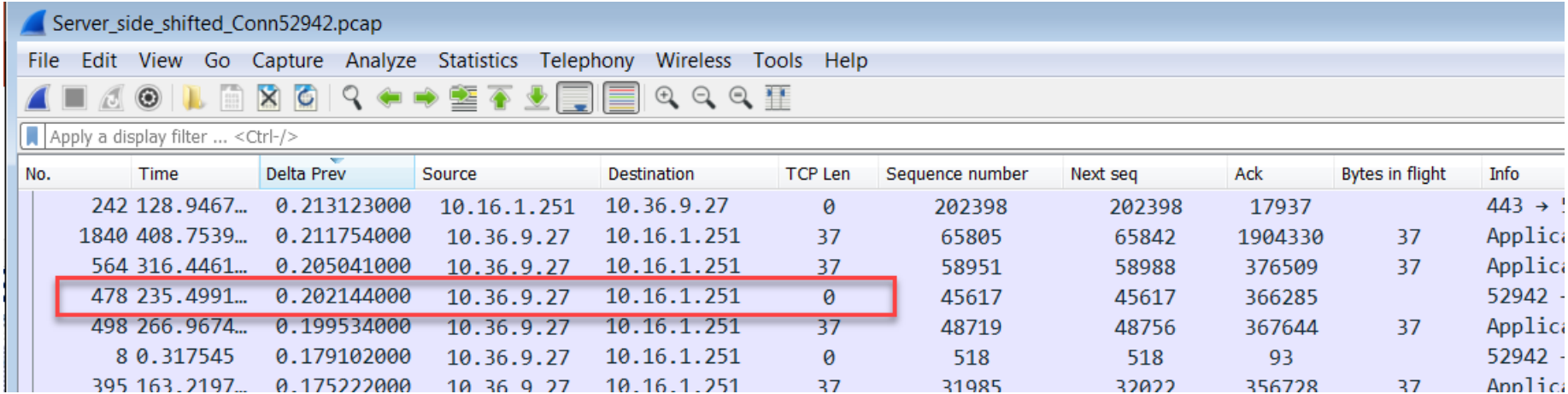
Discussion

- Server shows 342ms delay
- Latency + Delayed ACK Timer + Latency
- Delayed ACK Timer 209ms
- RTT 342ms – 209ms == 133ms
- Is this in the ball park?
- What's the smallest latency we've seen so far in these captures?

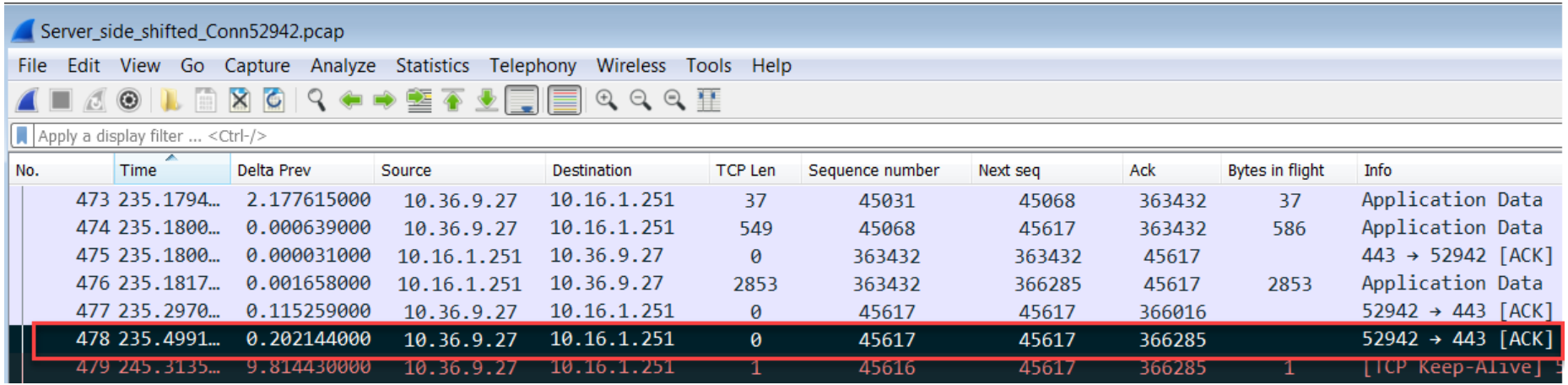
Let's Visualize



Let's look at one more...

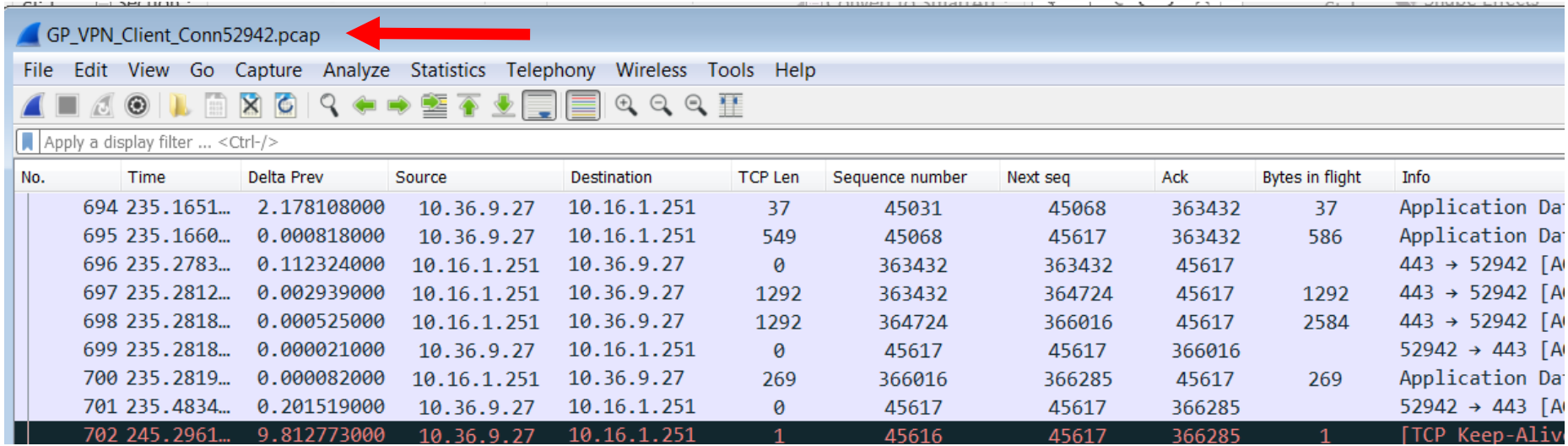


Server Side



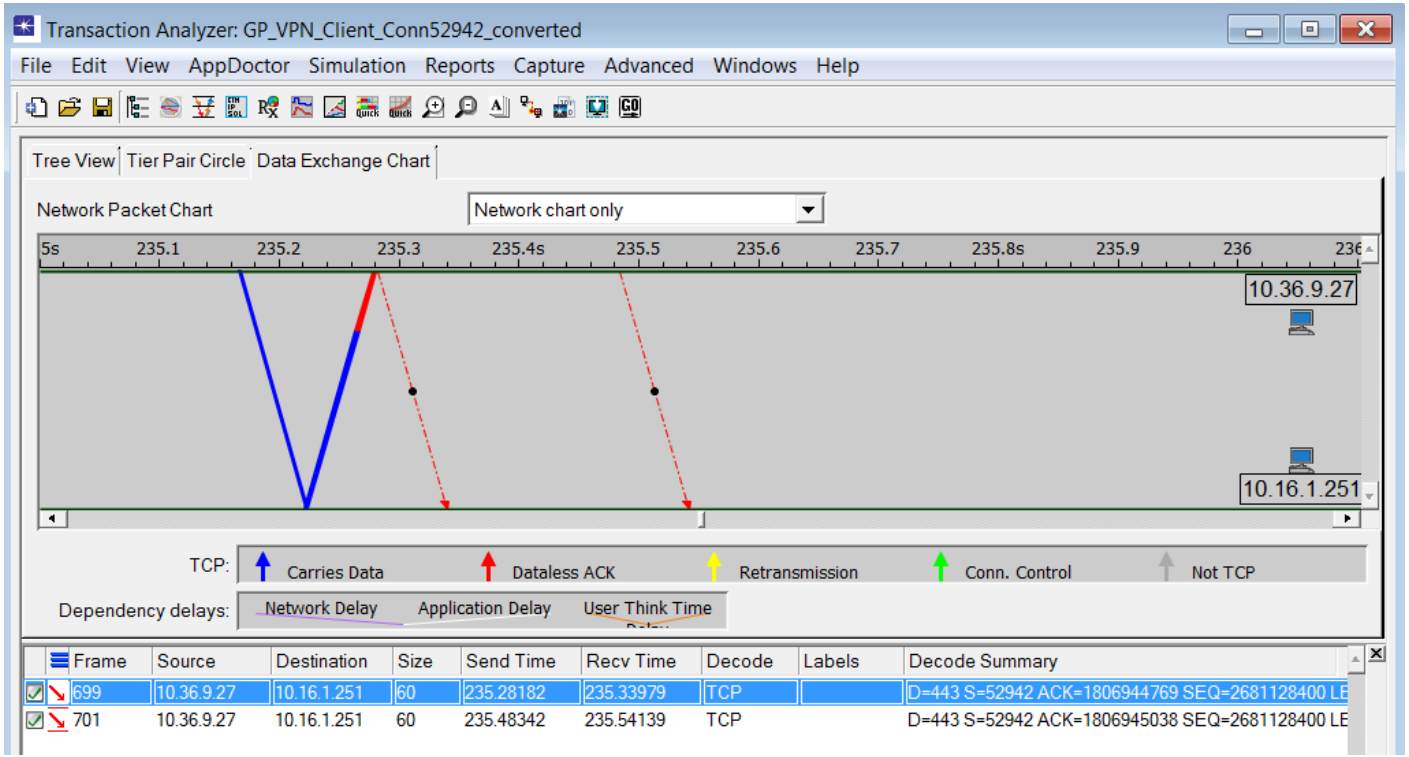
No.	Time	Delta Prev	Source	Destination	TCP Len	Sequence number	Next seq	Ack	Bytes in flight	Info
473	235.1794…	2.177615000	10.36.9.27	10.16.1.251	37	45031	45068	363432	37	Application Data
474	235.1800…	0.000639000	10.36.9.27	10.16.1.251	549	45068	45617	363432	586	Application Data
475	235.1800…	0.000031000	10.16.1.251	10.36.9.27	0	363432	363432	45617		443 → 52942 [ACK]
476	235.1817…	0.001658000	10.16.1.251	10.36.9.27	2853	363432	366285	45617	2853	Application Data
477	235.2970…	0.115259000	10.36.9.27	10.16.1.251	0	45617	45617	366016		52942 → 443 [ACK]
478	235.4991…	0.202144000	10.36.9.27	10.16.1.251	0	45617	45617	366285		52942 → 443 [ACK]
479	245.3135…	9.814430000	10.36.9.27	10.16.1.251	1	45616	45617	366285	1	[TCP Keep-Alive]

Client Side

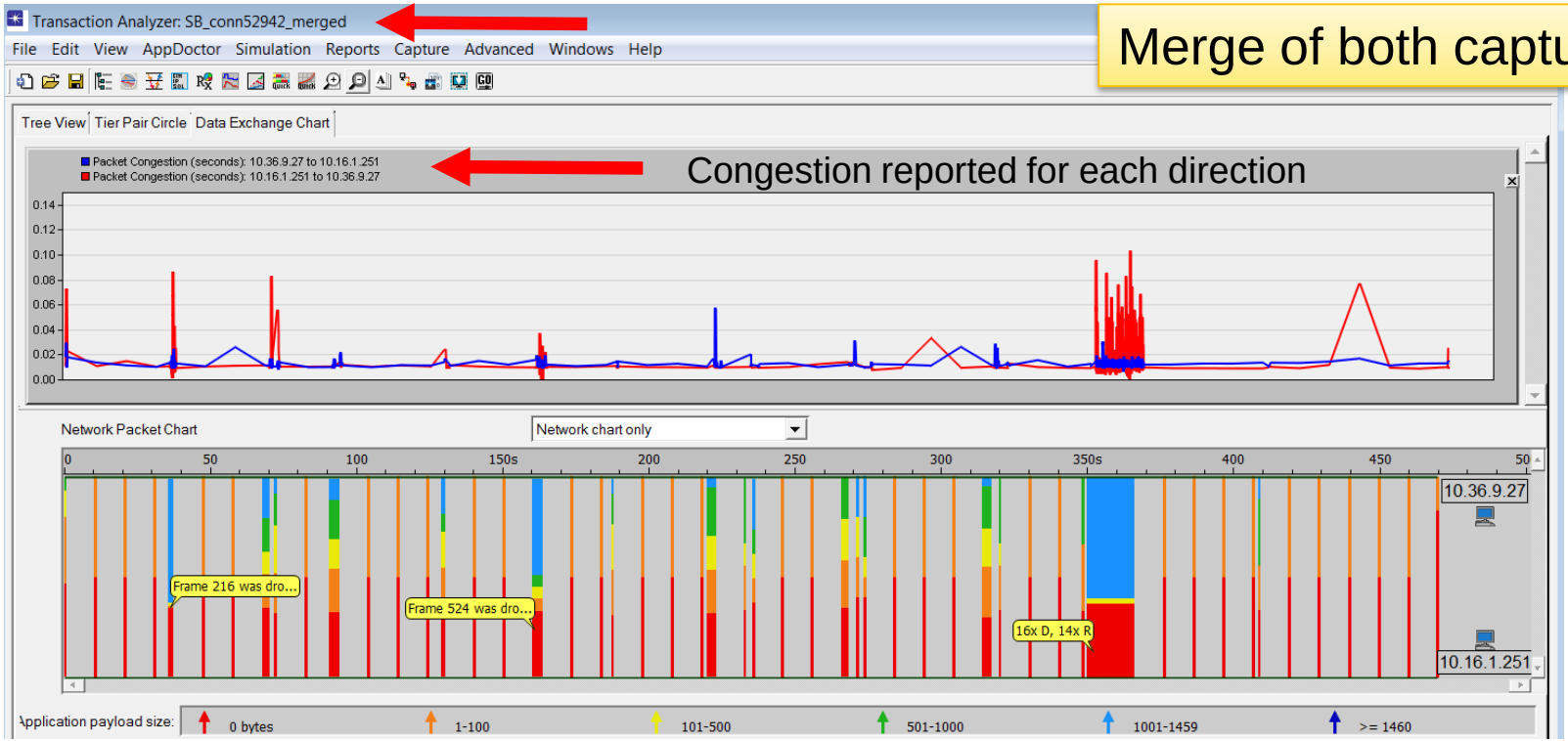


No.	Time	Delta Prev	Source	Destination	TCP Len	Sequence number	Next seq	Ack	Bytes in flight	Info
694	235.1651…	2.178108000	10.36.9.27	10.16.1.251	37	45031	45068	363432	37	Application Da
695	235.1660…	0.000818000	10.36.9.27	10.16.1.251	549	45068	45617	363432	586	Application Da
696	235.2783…	0.112324000	10.16.1.251	10.36.9.27	0	363432	363432	45617		443 → 52942 [A
697	235.2812…	0.002939000	10.16.1.251	10.36.9.27	1292	363432	364724	45617	1292	443 → 52942 [A
698	235.2818…	0.000525000	10.16.1.251	10.36.9.27	1292	364724	366016	45617	2584	443 → 52942 [A
699	235.2818…	0.000021000	10.36.9.27	10.16.1.251	0	45617	45617	366016		52942 → 443 [A
700	235.2819…	0.000082000	10.16.1.251	10.36.9.27	269	366016	366285	45617	269	Application Da
701	235.4834…	0.201519000	10.36.9.27	10.16.1.251	0	45617	45617	366285		52942 → 443 [A
702	245.2961…	9.812773000	10.36.9.27	10.16.1.251	1	45616	45617	366285	1	[TCP Keep-Aliv

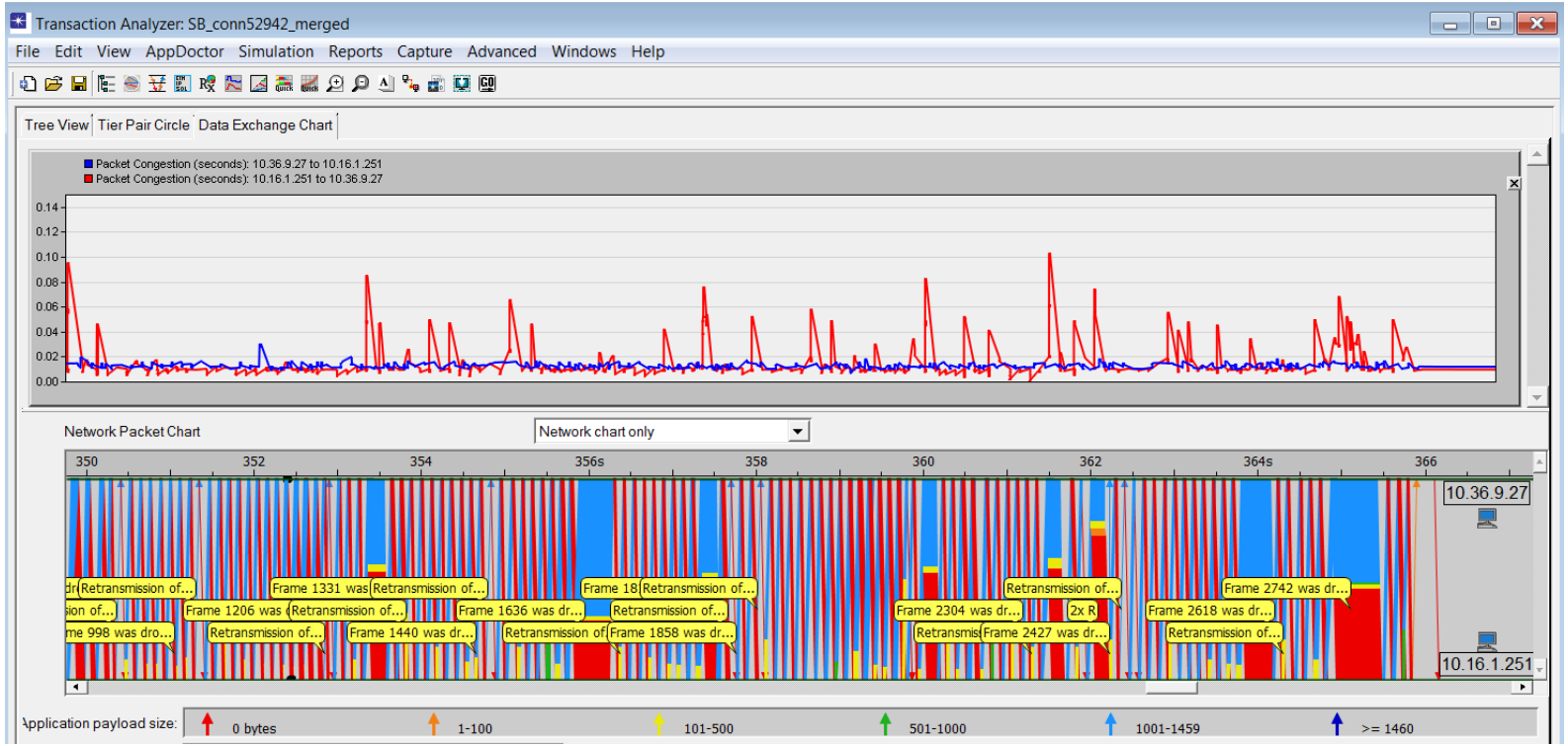
Visualized


Transaction Analyzer: GP_VPN_Client_Conn52942_converted
File Edit View AppDoctor Simulation Reports Capture Advanced Windows Help
Tree View
Tier Pair Circle
Data Exchange Chart
Network Packet Chart
Network chart only
5s
235.1
235.2
235.3
235.4s
235.5
235.6
235.7
235.8s
235.9
236
10.36.9.27
10.16.1.251
TCP:
Carries Data
Dataless ACK
Retransmission
Conn. Control
Not TCP
Dependency delays:
Network Delay
Application Delay
User Think Time
Frame Source Destination Size Send Time Recv Time Decode Labels Decode Summary
699 10.36.9.27 10.16.1.251 60 235.28182 235.33979 TCP D=443 S=52942 ACK=1806944769 SEQ=2681128400 LE
701 10.36.9.27 10.16.1.251 60 235.48342 235.54139 TCP D=443 S=52942 ACK=1806945038 SEQ=2681128400 LE

10,000 ft/m View



Zoom-in to File Download



Discussion

- Any surprises about the amount of congestion?
- Does this view help to explain how congestion impacts performance?

Post Session – Thank You!

- Thank you for viewing this bonus section after Sharkfest Euro 2018 Conference!

End of Session