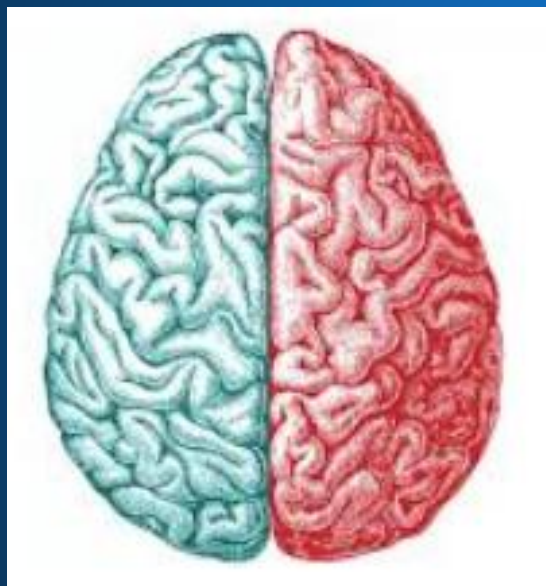




# SharkFest '19 Europe



## Session #11 - TCP Split Brain - Part I



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

**John Pittle**

Riverbed Technologies  
Performance Management  
Strategist  
[jpittle@riverbed.com](mailto:jpittle@riverbed.com)



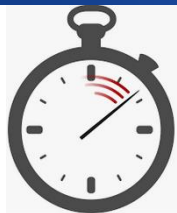
# Premise: TCP Split Brain



- 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



# Upfront Contract



- 75 Minutes x 2
- Your Agenda
- My Agenda



# Your Agenda



- Fine tune your Wireshark Skills
- Fine tune your TCP Skills
- Learn more about what you can expect to see in Sender vs. Receiver captures
- Improve your performance troubleshooting skills



# My Agenda



- 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 advanced visualizations that help reinforce the end point behavior we will be discussing



# About me?



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





# One of my Favorite Quotes

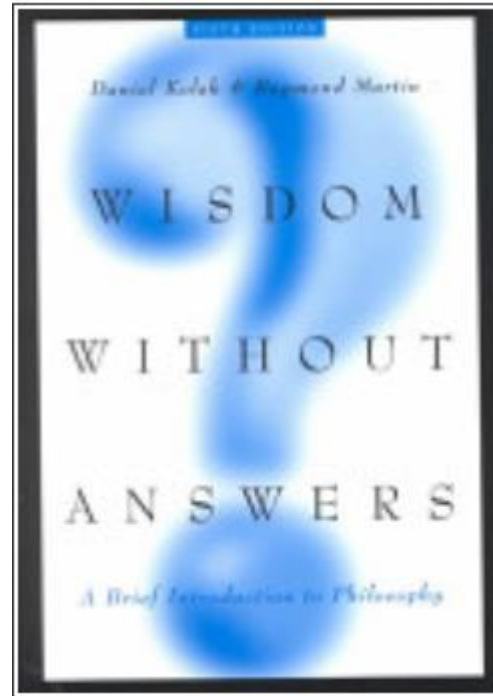


Education is not the filling of a pail, but  
the lighting of a fire.

William Butler Yeats



# One of my Favorite Books



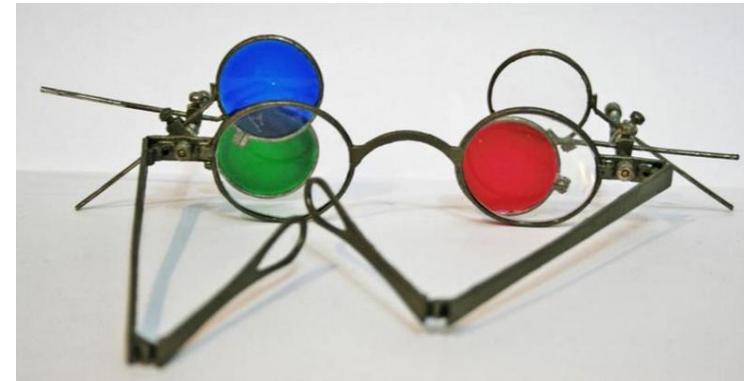




# My Ask of You



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





# Session Resources



- Pre-filtered end point capture files for the TCP connection of interest
- Wireshark Profiles
- Comparison Summary Spreadsheet
- Wireshark 2.6.3
- Riverbed Transaction Analyzer 17.6 - for Visualization and Advanced Analytics



# Comparison Summary



## TCP Split Brain Comparison Summary (Sender vs. Receiver)

Item <input type="checkbox"/>	Topic <input type="checkbox"/>	Summary <input type="checkbox"/>	Sender <input type="checkbox"/>	Receiver <input type="checkbox"/>



# Part I Topics



- Background on app, topology, and symptoms
- Compare and Contrast (aka Split Brain)
  - 3-way Handshake
  - Latency
  - TCP State
  - Expert Info
  - Fragment Overlaps (Part I)



# Part II Topics



- Continue where we left off
- Compare and contrast...
  - Fragment Overlaps (Part II)
  - Bytes in Flight
  - Bonus – If time is available
- Session Wrap-Up





# Troubleshooting 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

Approx. 111ms RTT



443

10.16.1.251

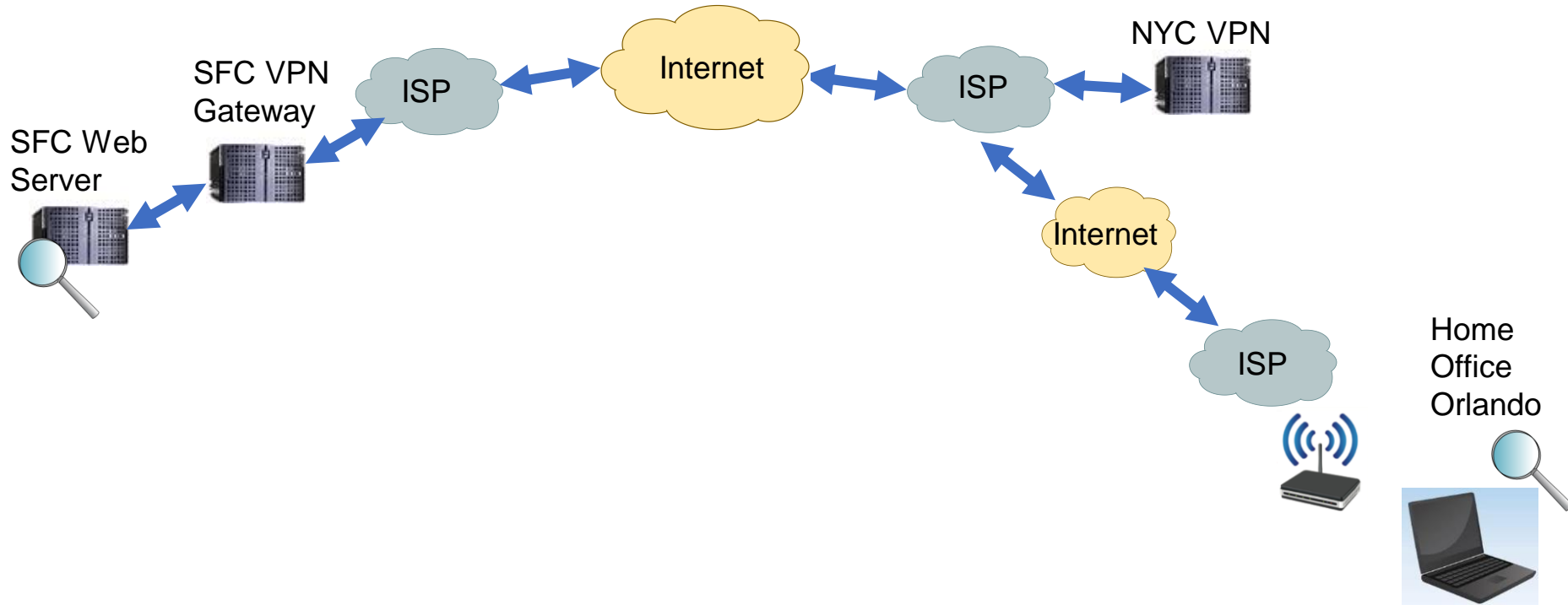
52942



10.36.9.27



# Actual Topology





# Pop Quiz #1



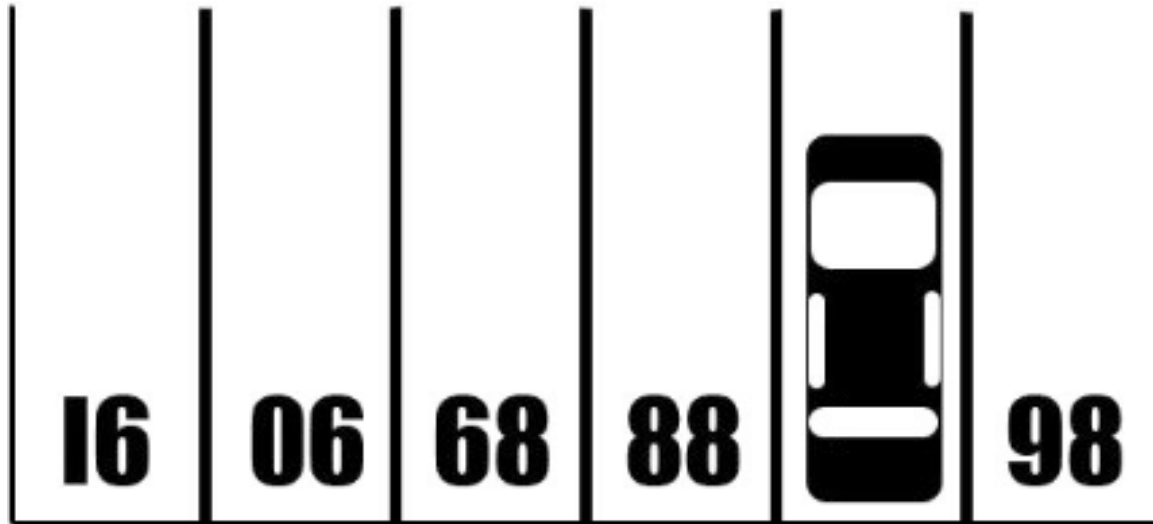
- Ready for our first Quiz?



# Pop Quiz



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

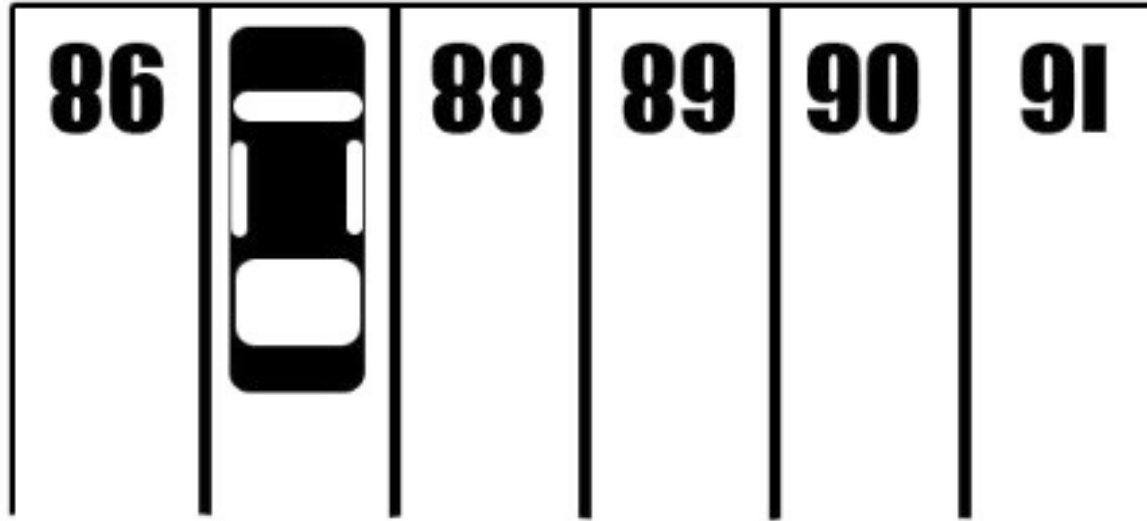




# Perspective



How about now?





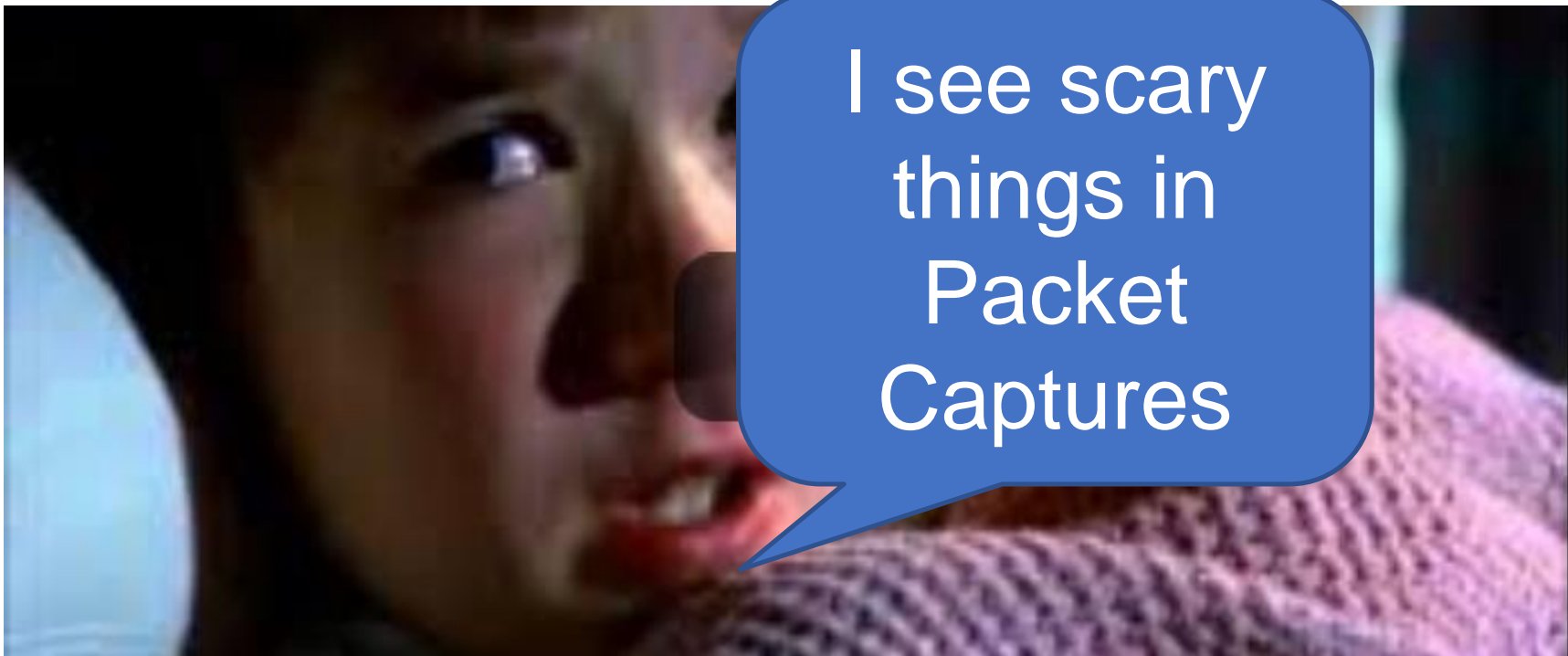
# TCP End Point Behavior



- 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 in parallel on each host
- 3<sup>rd</sup> party actors can also have an affect on behavior

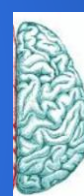


# Sixth Sense (1999)





# Comparisons



- 3-way handshake
- Latency
- TCP State
- Expert Info
- Fragment Overlaps / OOS / Retransmissions







# Split Brain Comparisons



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





# 3-Way Handshake - Client



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
1	0.000000	0.000000000	10.36.9.27	10.16.1.251	0	0	0	0		52942 → 443 [SYN]
2	0.121491	0.121491000	10.16.1.251	10.36.9.27	0	0	0	1		443 → 52942 [SYN]
3	0.121578	0.000087000	10.36.9.27	10.16.1.251	0	1	1	1		52942 → 443 [ACK]
4	0.124883	0.003305000	10.36.9.27	10.16.1.251	517	1	518	1	517	Client Hello

Acknowledgment number: 0  
1000 .... = Header Length: 32 bytes (8)

- ▶ **Flags: 0x002 (SYN)**
- Window size value: 8192  
[Calculated window size: 8192]
- Checksum: 0xac33 [unverified]  
[Checksum Status: Unverified]
- Urgent pointer: 0
- ▶ **Options: (12 bytes), Maximum segment size, No-Operation (NOP), Window scale, No-Operation (NOP), No-Operation (NOP), SACK permitted**
  - ▶ TCP Option - Maximum segment size: 1360 bytes
  - ▶ TCP Option - No-Operation (NOP)
  - ▶ TCP Option - Window scale: 8 (multiply by 256)
  - ▶ TCP Option - No-Operation (NOP)
  - ▶ TCP Option - No-Operation (NOP)
  - ▶ TCP Option - SACK permitted
- ▶ [Timestamps]



# 3-Way Handshake - Client



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
1	0.000000	0.000000000	10.36.9.27	10.16.1.251	0	0	0	0		52942 → 443 [SYN]
2	0.121491	0.121491000	10.16.1.251	10.36.9.27	0	0	0	1		443 → 52942 [SYN]
3	0.121578	0.000087000	10.36.9.27	10.16.1.251	0	1	1	1		52942 → 443 [ACK]
4	0.124883	0.003305000	10.36.9.27	10.16.1.251	517	1	518	1	517	Client Hello

Acknowledgment number: 0  
1000 .... = Header Length: 32 bytes (8)

▶ **Flags: 0x002 (SYN)**

Window size value: 8192  
[Calculated window size: 8192]  
Checksum: 0xac33 [unverified]  
[Checksum Status: Unverified]  
Urgent pointer: 0

Options: (12 bytes), Maximum segment size, No-Operation (NOP), Window scale, No-Operation (NOP), No-Operation (NOP), SACK permitted

- ▶ TCP Option - Maximum segment size: 1360 bytes
- ▶ TCP Option - No-Operation (NOP)
- ▶ TCP Option - Window scale: 8 (multiply by 256)
- ▶ TCP Option - No-Operation (NOP)
- ▶ TCP Option - No-Operation (NOP)
- ▶ TCP Option - SACK permitted

▶ [Timestamps]



# 3-Way Handshake - Server



Server\_side\_shifted\_Conn52942.pcap

File Edit View Go Capture Analyze Statistics Telephony Wireless Tools Help

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

No.	Time	Delta Prev	Source	Destination	TCP Len	Sequence number	Next seq	Ack	Bytes in flight	Info
1	0.000000	0.000000000	10.36.9.27	10.16.1.251	0	0	0	0		S+, 52942 → 443 [
2	0.000054	0.000054000	10.16.1.251	10.36.9.27	0	0	0	1		443 → 52942 [SYN,
3	0.129252	0.129198000	10.36.9.27	10.16.1.251	0	1	1	1		52942 → 443 [ACK]

Acknowledgment number: 1 (relative ack number)  
1000 .... = Header Length: 32 bytes (8)

▶ **Flags: 0x012 (SYN, ACK)**

Window size value: 8192  
[Calculated window size: 8192]  
Checksum: 0x1f70 [unverified]  
[Checksum Status: Unverified]  
Urgent pointer: 0

Options: (12 bytes), Maximum segment size, No-Operation (NOP), Window scale, No-Operation (NOP), No-Operation (NOP), SACK permitted

- ▶ TCP Option - Maximum segment size: 1460 bytes
- ▶ TCP Option - No-Operation (NOP)
- ▶ TCP Option - Window scale: 8 (multiply by 256)
- ▶ TCP Option - No-Operation (NOP)
- ▶ TCP Option - No-Operation (NOP)
- ▶ TCP Option - SACK permitted





# Learn More about MSS



tcp mss



About 1,330,000 results (0.53 seconds)

The maximum segment size (**MSS**) is a parameter of the options field of the **TCP** header that specifies the largest amount of data, specified in bytes, that a computer or communications device can receive in a single **TCP** segment.



[Maximum segment size - Wikipedia](https://en.wikipedia.org/wiki/Maximum_segment_size)

[https://en.wikipedia.org/wiki/Maximum\\_segment\\_size](https://en.wikipedia.org/wiki/Maximum_segment_size)

About Featured Snippets Feedback

People also ask

What is TCP adjust MSS?



## Maximum segment size



The maximum segment size is a parameter of the options field of the TCP header that specifies the largest amount of data, specified in bytes, that a computer or communications device can receive in a single TCP segment. It does not count the TCP header or the IP header. [Wikipedia](#)

Feedback



# 1<sup>st</sup> Leg Completed



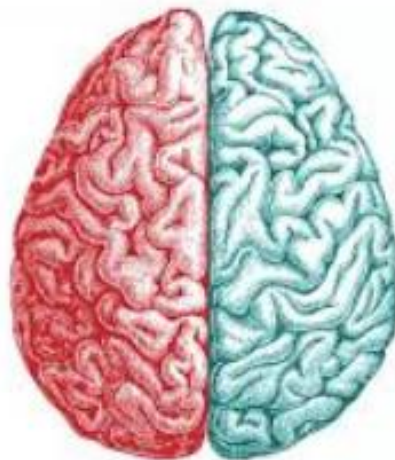




# Split Brain Comparisons



- Latency Checks







# Latency Check



- Client Capture – 121ms

No.	Time	Time delta from previous capture point	Time delta from previous interface capture time	Source	Destination	Identification	Length	Sequence number	Acknowledgment number	Bytes in flight	Info
1	09:11:42.223468	0.000000000	0.000000000	10.36.9.27	10.16.1.251	0x0085 (1...	66	0	0	0	52942 → 443 [SYN] Seq=...
2	09:11:42.344959	0.121491000	0.121491000	10.16.1.251	10.36.9.27	0x77bc (3...	66	0	1	1	443 → 52942 [SYN, ACK] Seq=...
3	09:11:42.345046	0.000087000	0.000087000	10.36.9.27	10.16.1.251	0x0091 (1...	60	1	1	1	52942 → 443 [ACK] Seq=...

- Server Capture – 129ms

No.	Time	Time delta from previous capture point	Time delta from previous interface capture time	Source	Destination	Identification	Length	Sequence number	Acknowledgment number	Bytes in flight	Info
1	09:11:42.262470	0.000000...	0.000000...	10.36.9.27	10.16.1.2...	0x0085 (1...	82	0	0	0	S+, 52942 → 443 [SYN] Seq=...
2	09:11:42.262524	0.000054...	0.0000540...	10.16.1.2...	10.36.9.27	0x77bc (3...	66	0	1	1	443 → 52942 [SYN, ACK] Seq=...
3	09:11:42.391722	0.129198...	0.1291980...	10.36.9.27	10.16.1.2...	0x0091 (1...	60	1	1	1	52942 → 443 [ACK] Seq=...



# Latency Check



- Client Capture – 121ms

No.	Time	Time delta from previous capture	Time delta from previous packet	Source	Destination	Identification	Length	Sequence number	Acknowledgment	Bytes in flight	Info
1	09:11:42.223468	0.000000000	0.000000000	10.36.9.27	10.16.1.251	0x0085 (1...	66	0	0	0	52942 → 443 [SYN] Seq=...
2	09:11:42.344959	0.121491000	0.121491000	10.16.1.251	10.36.9.27	0x77bc (3...	66	0	1	1	443 → 52942 [SYN, ACK] Seq=...
3	09:11:42.345046	0.000087000	0.000087000	10.36.9.27	10.16.1.251	0x0091 (1...	60	1	1	1	52942 → 443 [ACK] Seq=...

- Server Capture – 129ms

No.	Time	Time delta from previous capture	Time delta from previous packet	Source	Destination	Identification	Length	Sequence number	Acknowledgment	Bytes in flight	Info
1	09:11:42.26247	0.000000...	0.000000...	10.36.9.27	10.16.1.2...	0x0085 (1...	82	0	0	0	S+, 52942 → 443 [SYN] Seq=...
2	09:11:42.26252	0.000054...	0.0000540...	10.16.1.2...	10.36.9.27	0x77bc (3...	66	0	1	1	443 → 52942 [SYN, ACK] Seq=...
3	09:11:42.39172	0.129198...	0.1291980...	10.36.9.27	10.16.1.2...	0x0091 (1...	60	1	1	1	52942 → 443 [ACK] Seq=...



# Pop Quiz



- If you've been given a packet capture, and you don't know if it's the server or the client, how can tell?



# Identify Role from SYN



GP\_VPN\_Client\_Conn52942.pcap

File Edit View Go Capture Analyze Statistics Telephony Wireless Tools Help

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

No.	Time	Time delta from previous	Time delta from previous	Source	Destination	Identification	Length	Sequence number	Acknowledgment	Bytes in flight	Info
1	09:11:42.223468	0.000000000	0.000000000	10.36.9.27	10.16.1.251	0x0085 (1...	66	0	0	0	52942 → 443 [SYN] S
2	09:11:42.344959	0.121491000	0.121491000	10.16.1.251	10.36.9.27	0x77bc (3...	66	0	1	1	443 → 52942 [SYN, A
3	09:11:42.345046	0.000087000	0.000087000	10.36.9.27	10.16.1.251	0x0091 (1...	60	1	1	1	52942 → 443 [ACK] S

Server\_side\_shifted\_Conn52942.pcap

File Edit View Go Capture Analyze Statistics Telephony Wireless Tools Help

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

No.	Time	Time delta from previ	Time delta from previo	Source	Destination	Identification	Length	Sequence number	Acknowledgment	Bytes in flight	Info
1	09:11:42.26247	0.000000...	.0000000...	10.36.9.27	10.16.1.2...	0x0085 (1...	82	0	0	0	S+, 52942 → 443 [SYN]
2	09:11:42.26252	0.000054...	.0000540...	10.16.1.2...	10.36.9.27	0x77bc (3...	66	0	1	1	443 → 52942 [SYN, ACK]
3	09:11:42.39172	0.129198...	.1291980...	10.36.9.27	10.16.1.2...	0x0091 (1...	60	1	1	1	52942 → 443 [ACK] Seq=



# Identify Role from SYN



If SYN+ACK has high time delta, then capture came from client

GP\_VPN\_Client\_Conn52942.pcap

File Edit View Go Capture Analyze Statistics Telephony Wireless Tools Help

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

No.	Time	Time delta from previous	Time delta from previous	Source	Destination	Identification	Length	Sequence number	Acknowledgment	Bytes in flight	Info
1	09:11:42.223468	0.000000000	0.000000000	10.36.9.27	10.16.1.251	0x0085 (1...	66	0	0	0	52942 → 443 [SYN] S
2	09:11:42.344959	0.121491000	0.121491000	10.16.1.251	10.36.9.27	0x77bc (3...	66	0	1	1	443 → 52942 [SYN, A
3	09:11:42.345046	0.000087000	0.000087000	10.36.9.27	10.16.1.251	0x0091 (1...	60	1	1	1	52942 → 443 [ACK] S

If ACK has high time delta, then capture came from client

Server\_side\_shifted\_Conn52942.pcap

File Edit View Go Capture Analyze Statistics Telephony Wireless Tools Help

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

No.	Time	Time delta from previ	Time delta from previo	Source	Destination	Identification	Length	Sequence number	Acknowledgment	Bytes in flight	Info
1	09:11:42.26247	0.000000...	0.000000...	10.36.9.27	10.16.1.2...	0x0085 (1...	82	0	0	0	S+, 52942 → 443 [SYN]
2	09:11:42.26252	0.000054...	0.0000540...	10.16.1.2...	10.36.9.27	0x77bc (3...	66	0	1	1	443 → 52942 [SYN, ACK]
3	09:11:42.39172	0.129198...	0.1291980...	10.36.9.27	10.16.1.2...	0x0091 (1...	60	1	1	1	52942 → 443 [ACK] Seq-



# Wireshark i(nitial)RTT



GP\_VPN\_Client\_Conn52942.pcap

File Edit View Go Capture Analyze Statistics Telephony Wireless Tools Help

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

No.	Time	Delta Prev	Source	Destination	TCP Len	Sequence number	Next seq	Ack
1	0.000000	0.000000000	10.36.9.27	10.16.1.251	0	0	0	0
2	0.121491	0.121491000	10.16.1.251	10.36.9.27	0	0	0	1
3	0.121578	0.000087000	10.36.9.27	10.16.1.251	0	1	1	1
4	0.124883	0.003305000	10.36.9.27	10.16.1.251	517	1	518	1

[Next sequence number: 1 (relative sequence number)]  
Acknowledgment number: 1 (relative ack number)  
0101 .... = Header Length: 20 bytes (5)  
Flags: 0x010 (ACK)  
Window size value: 257  
[Calculated window size: 65792]  
[Window size scaling factor: 256]  
Checksum: 0x6ba1 [unverified]  
[Checksum Status: Unverified]  
Urgent pointer: 0  
SEQ/ACK analysis  
[This is an ACK to the segment in frame: 2]  
[The RTT to ACK the segment was: 0.000087000 seconds]  
[iRTT: 0.121578000 seconds]  
[Timestamps]

GP\_VPN\_Client\_Conn52942.pcap | Packets: 2818 · Displayed: 2818 (100.0%) | Profile: SB-SACK

0:15



# Comparison Chart



TCP Split Brain Comparison Summary (Sender vs. Receiver)				
Item	Topic	Summary	Sender	Receiver
1	SYN Options	MSS, Scaling, TimeStamps, SACK	Negotiation & Adapt, MSS=1460, WS=8, SACK	Negotiation & Adapt, MSS=1360, WS=8, SACK
2	Latency	Can be different in each direction	Client iRTT == 121ms	Server iRTT == 129ms



# 2nd Leg Completed







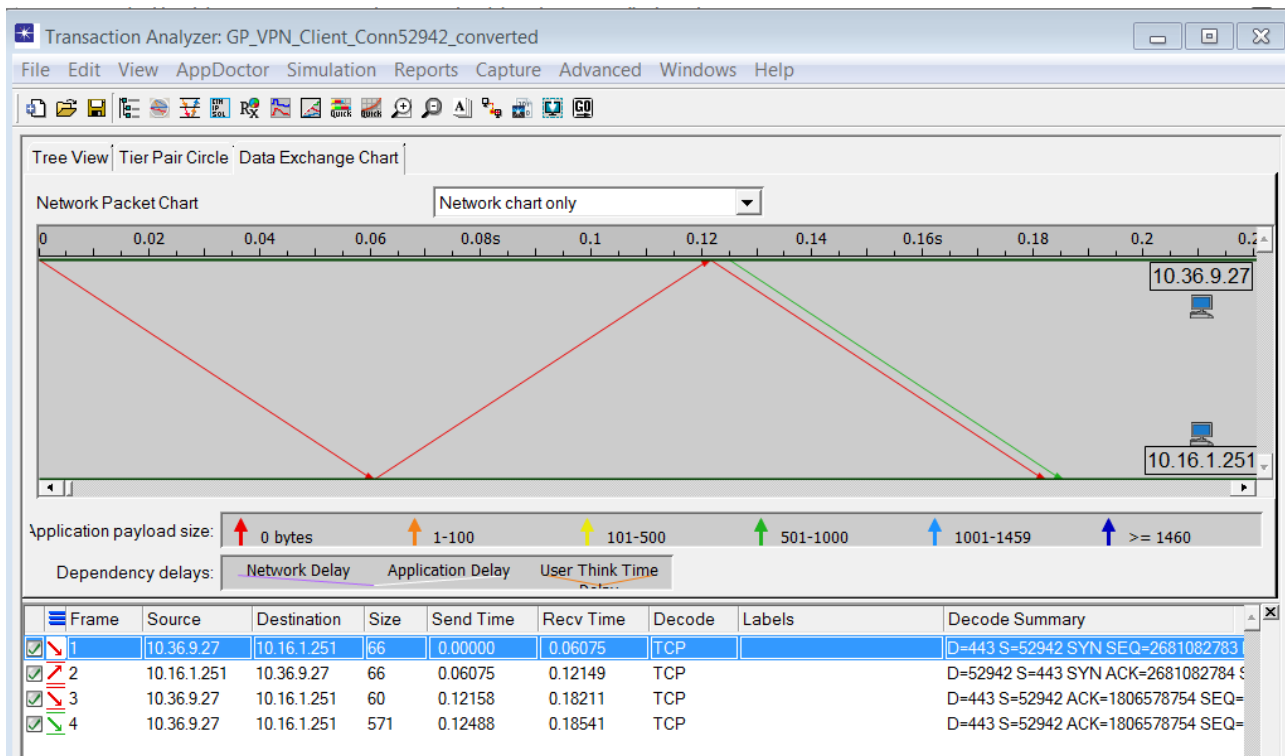
# What if we could visualize the traffic exchanges?



- Traffic bouncing between hosts
- Frequency, density, duration over time
- Transfer time
- Expert analytics overlaid with the visualized traffic
- If only there were such a capability...

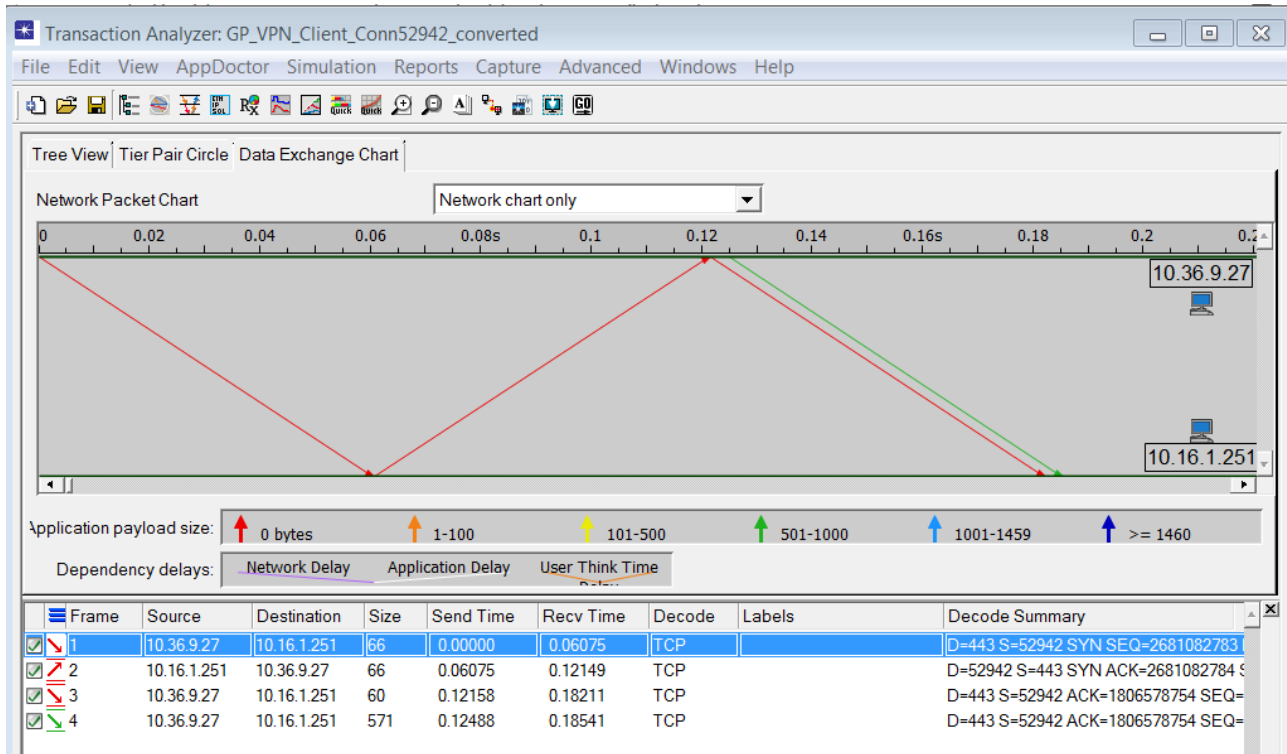


# Visualize Traffic over Time



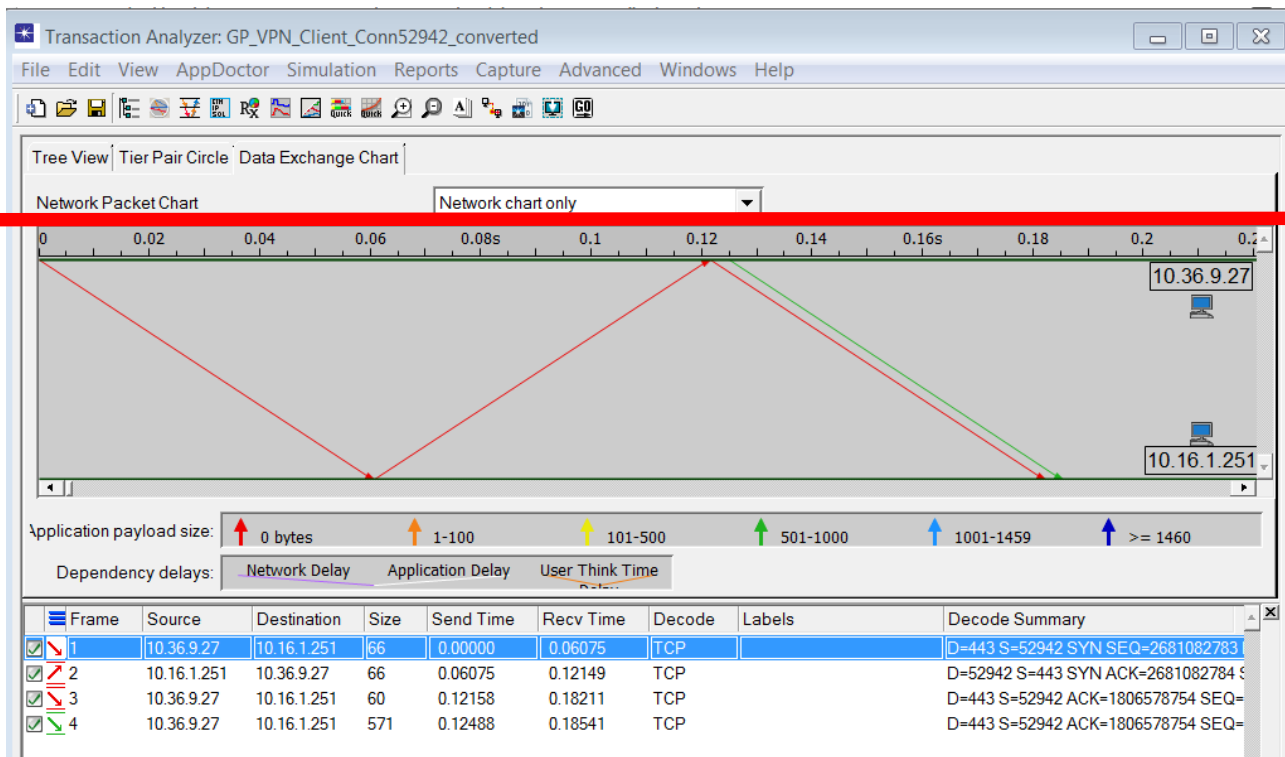


# Quick Orientation



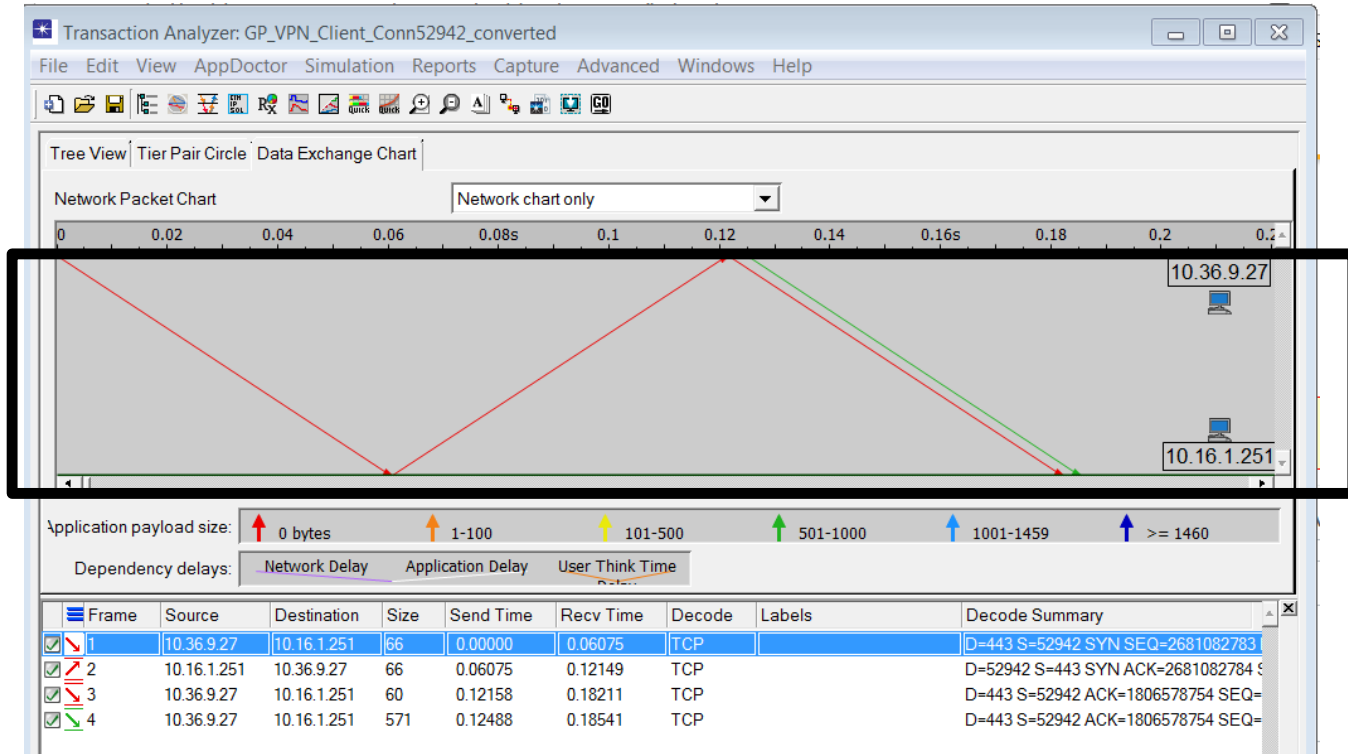


# Time Moves Left to Right



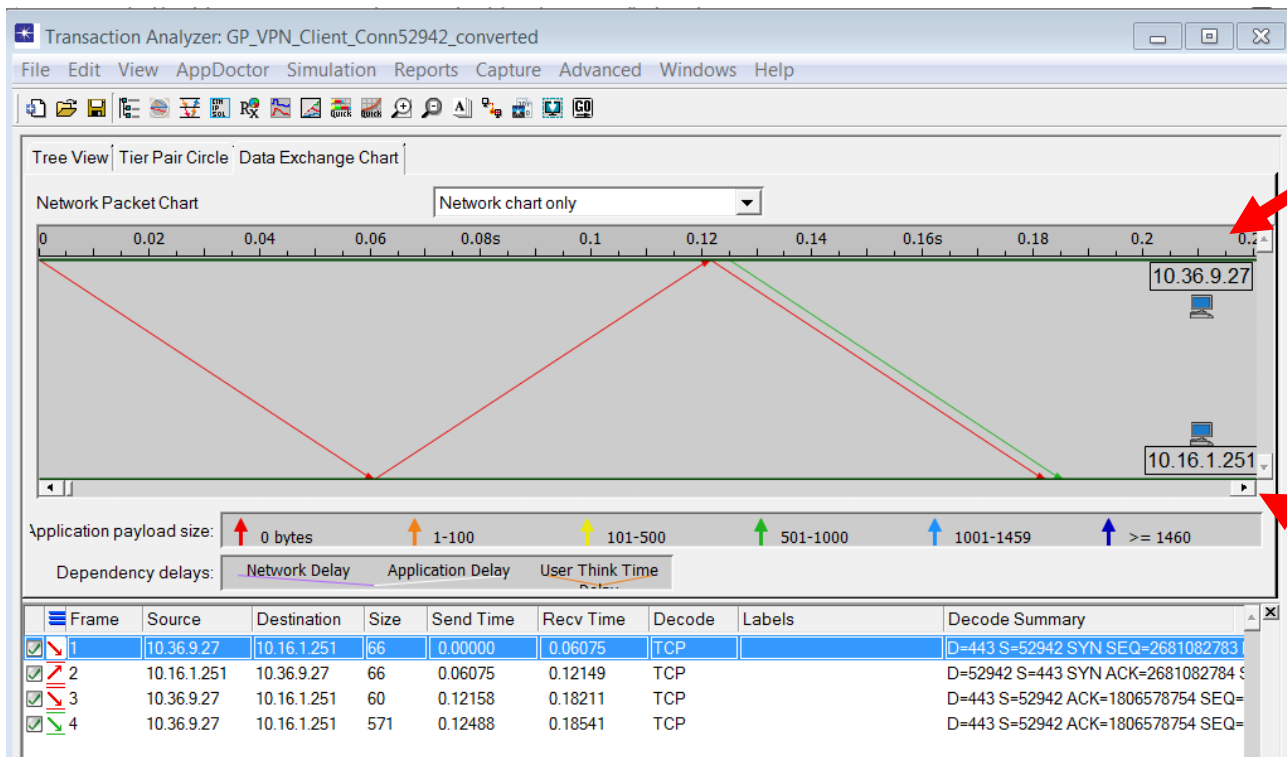


# Swimlane





# Top and Bottom of Swimlane



Client

Server



# Optional Summary Decodes



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

Application payload size: 0 bytes, 1-100, 101-500, 501-1000, 1001-1459, >= 1460

Dependency delays: Network Delay, Application Delay, User Think Time

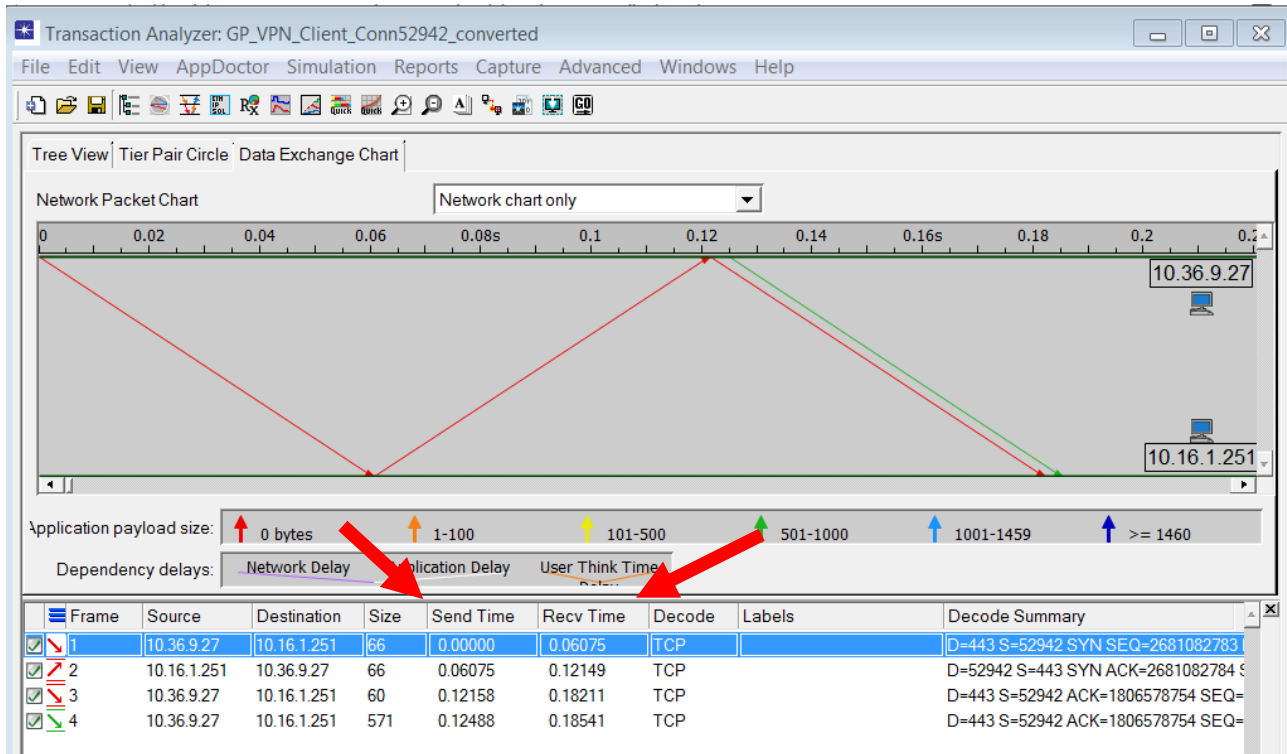
Frame	Source	Destination	Size	Send Time	Recv Time	Decode	Labels	Decode Summary
1	10.36.9.27	10.16.1.251	66	0.00000	0.06075	TCP		D=443 S=52942 SYN SEQ=2681082783
2	10.16.1.251	10.36.9.27	66	0.06075	0.12149	TCP		D=52942 S=443 SYN ACK=2681082784
3	10.36.9.27	10.16.1.251	60	0.12158	0.18211	TCP		D=443 S=52942 ACK=1806578754 SEQ=
4	10.36.9.27	10.16.1.251	571	0.12488	0.18541	TCP		D=443 S=52942 ACK=1806578754 SEQ=

Based on what you select in the swimlane





# Send Time / Recv Time







# Estimated vs. Actual Times



- Essential aspect of TCP Split Brain
- Each capture has the actual times a packet is sent or received from / to the host running the capture
- 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...
- ... and we have an idea of latency...
- ...we can make a reasonable estimate about the time it was transmitted from the other host



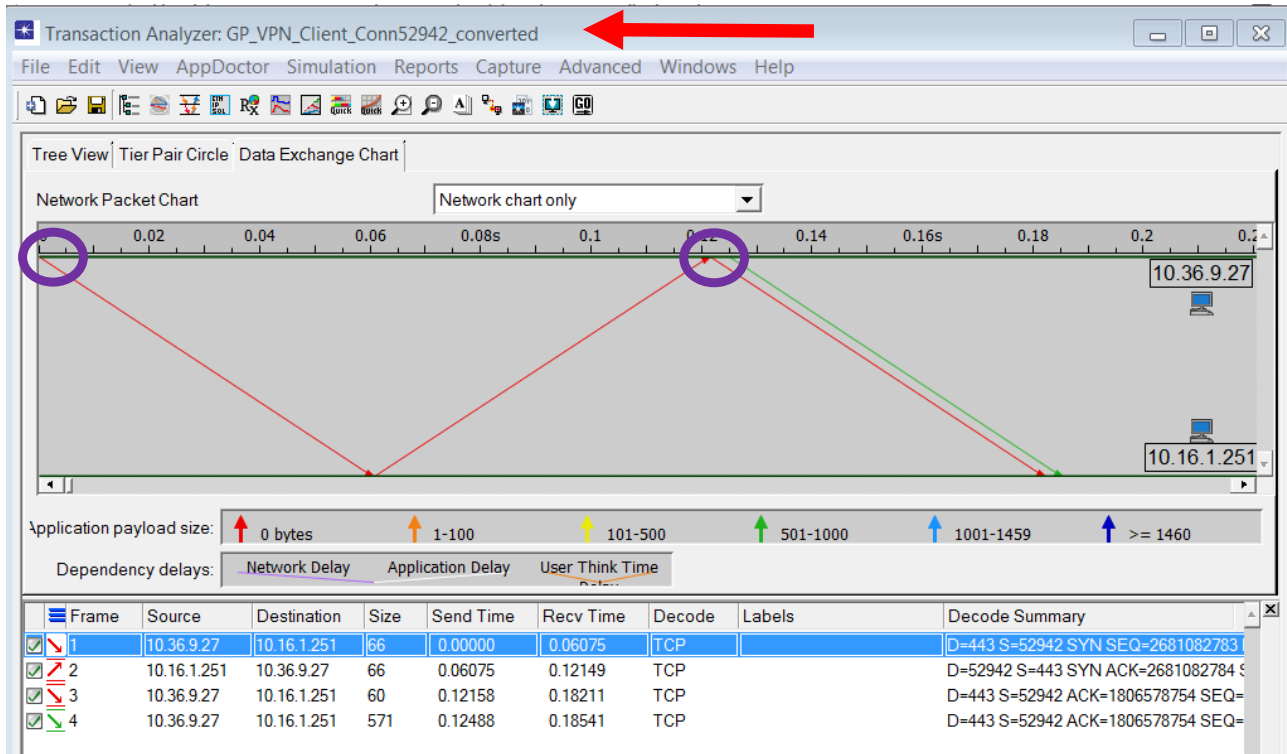
# Estimated vs. Actual Times



- If we know when a packet was transmitted...
- ...and we have an idea of latency...
- ...we can make a reasonable estimate about the time it was received at other host

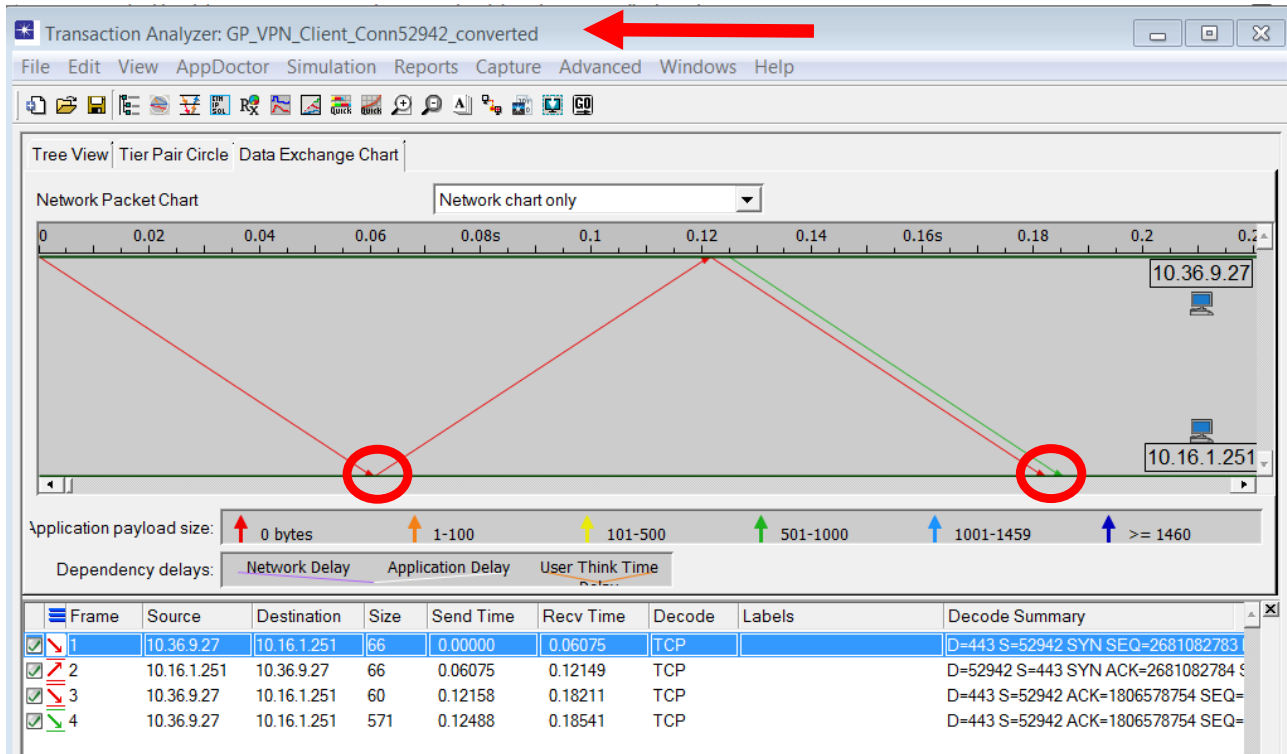


# Example: What we know





# 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



TCP A		TCP B
1. CLOSED		LISTEN
2. SYN-SENT	--> <SEQ=100><CTL=SYN>	--> SYN-RECEIVED
3. ESTABLISHED	<-- <SEQ=300><ACK=101><CTL=SYN,ACK>	<-- SYN-RECEIVED
4. ESTABLISHED	--> <SEQ=101><ACK=301><CTL=ACK>	--> ESTABLISHED
5. ESTABLISHED	--> <SEQ=101><ACK=301><CTL=ACK><DATA>	--> ESTABLISHED

Basic 3-Way Handshake for Connection Synchronization



# Stopwatch to Freeze Time



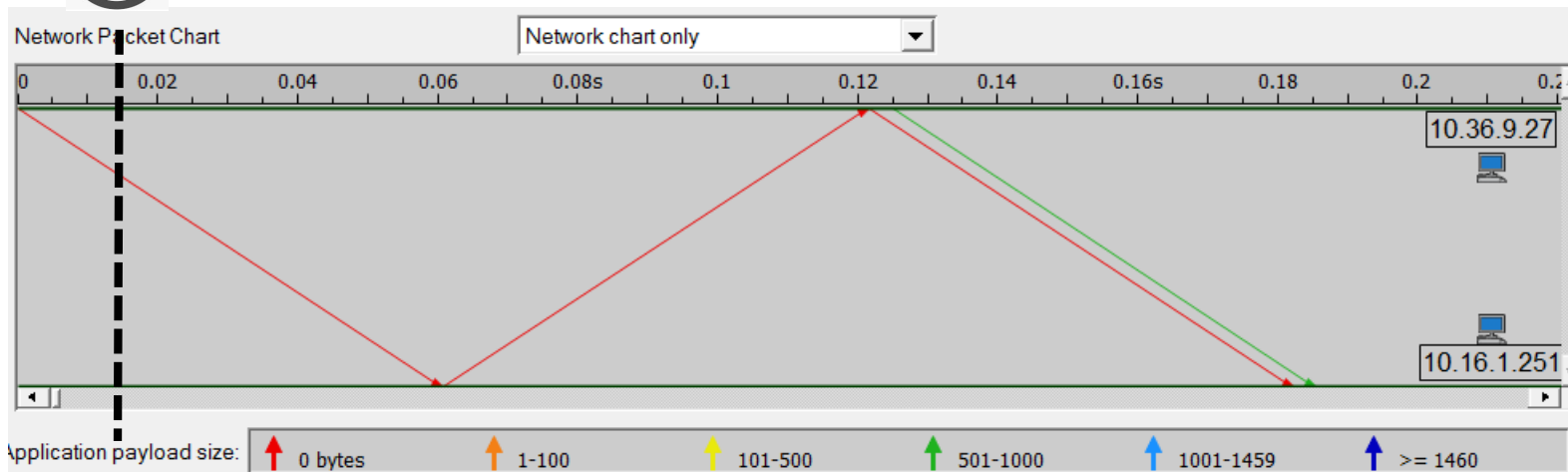




# Split Brain – State Discussion



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

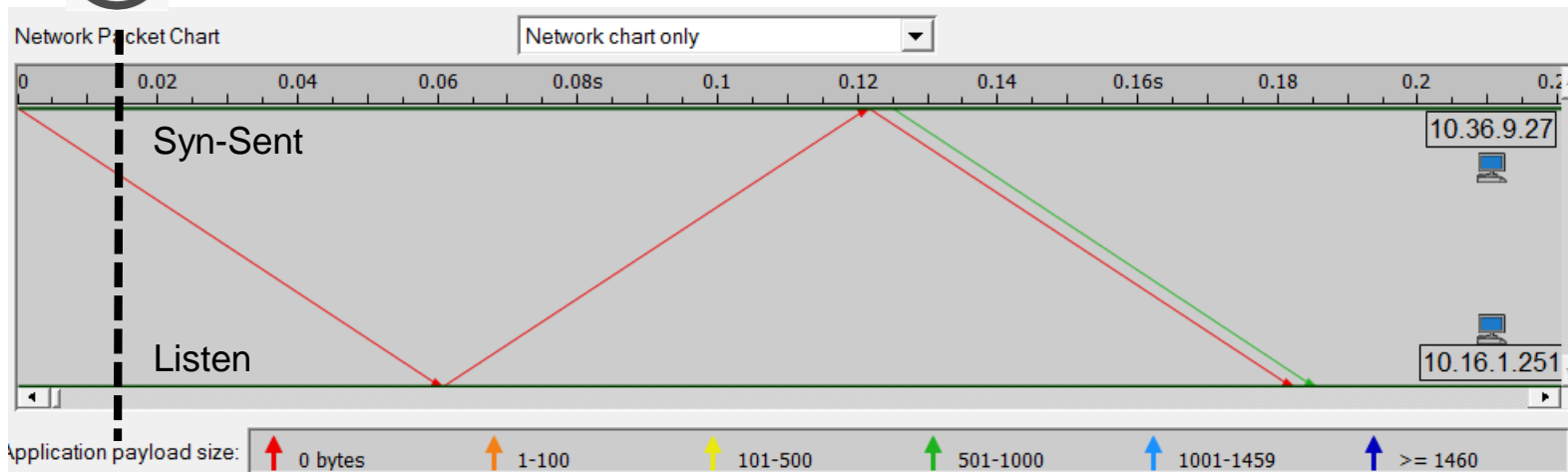




# Split Brain – State Discussion



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



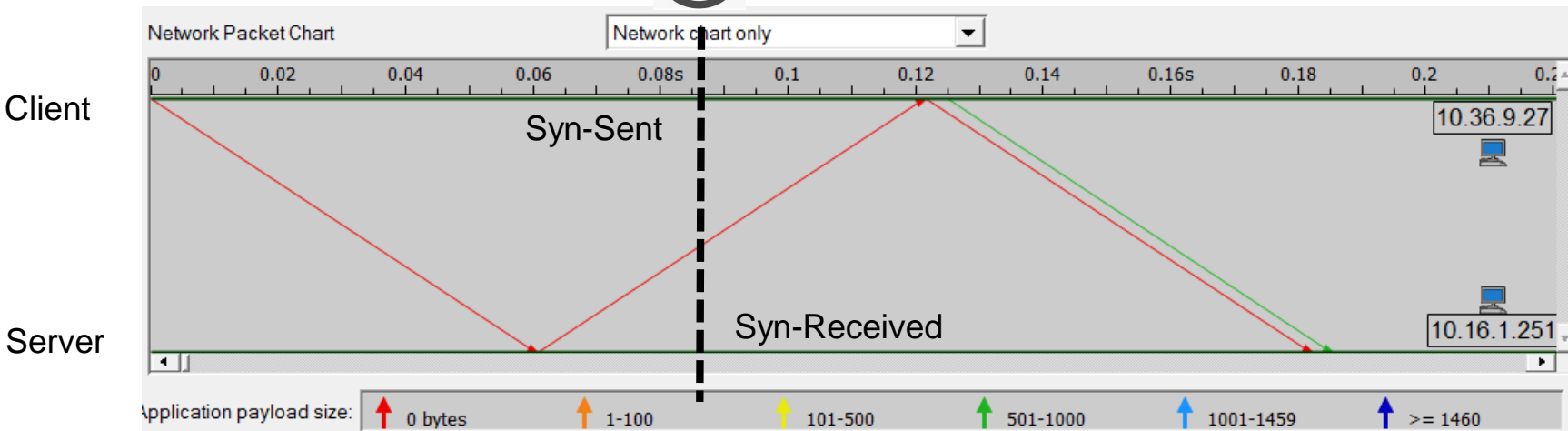


# Split Brain – State Discussion



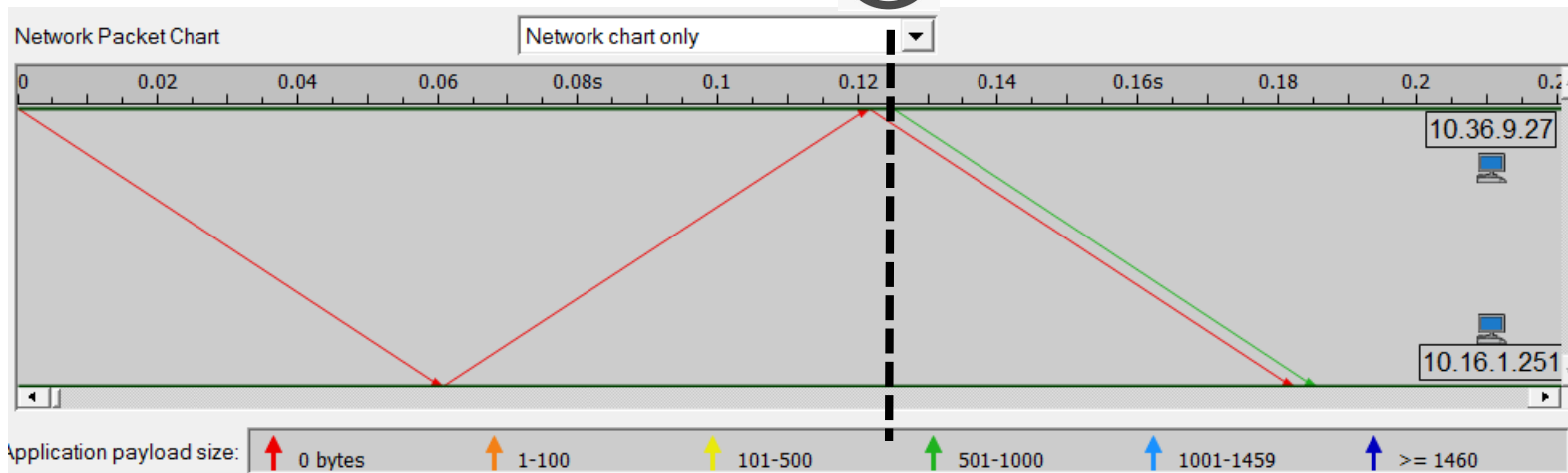


# Split Brain – State Discussion



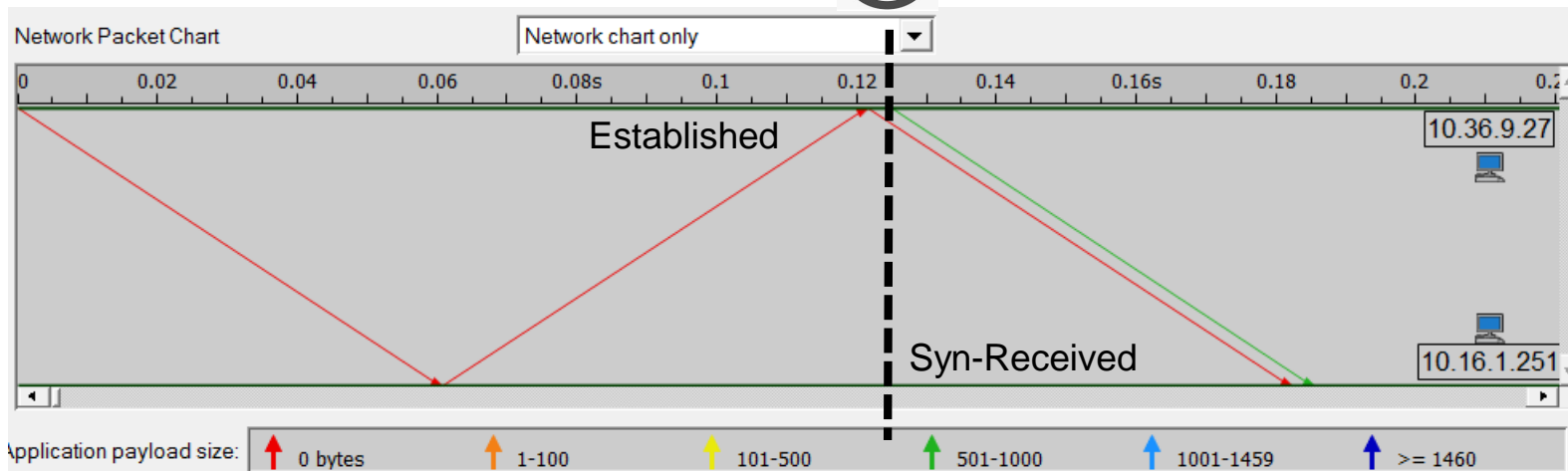


# Split Brain – State Discussion



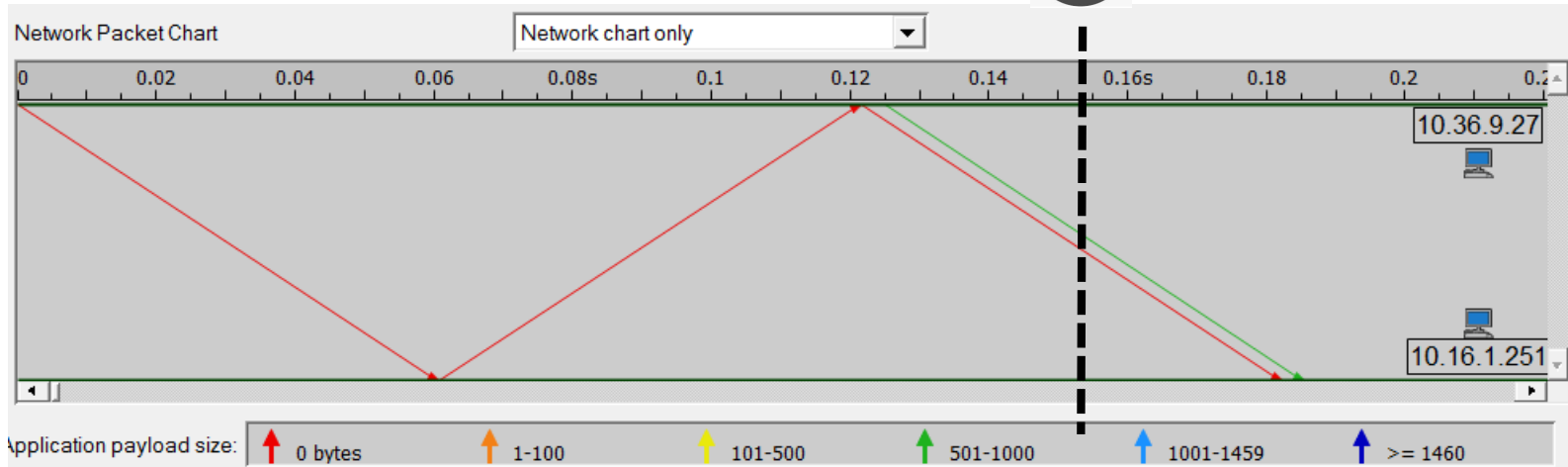


# Split Brain – State Discussion



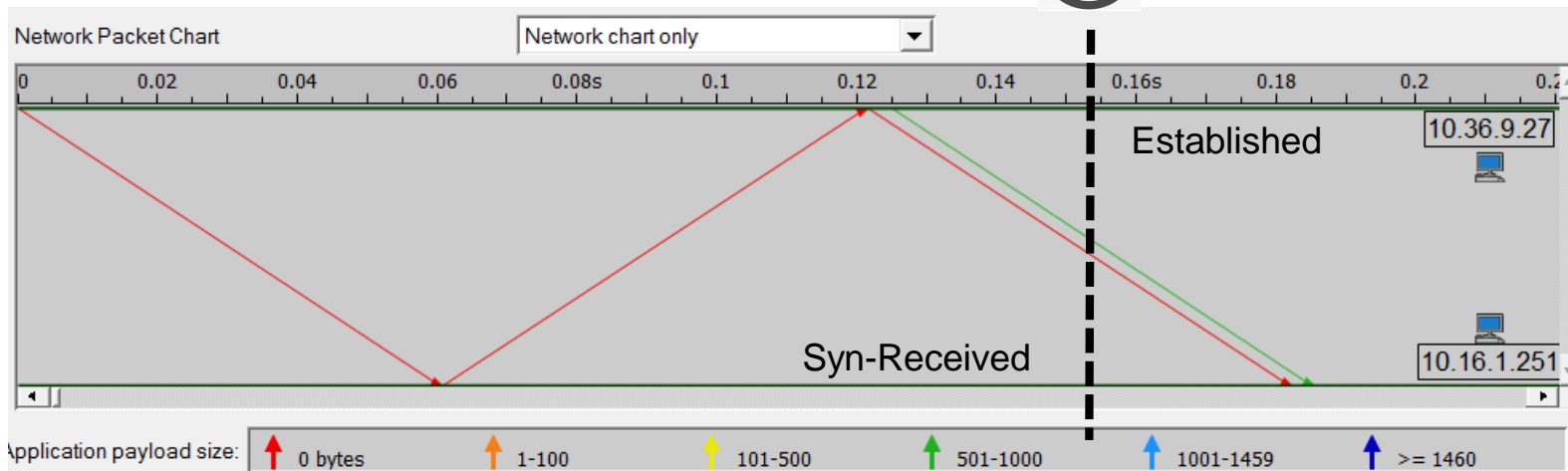


# Split Brain – State Discussion





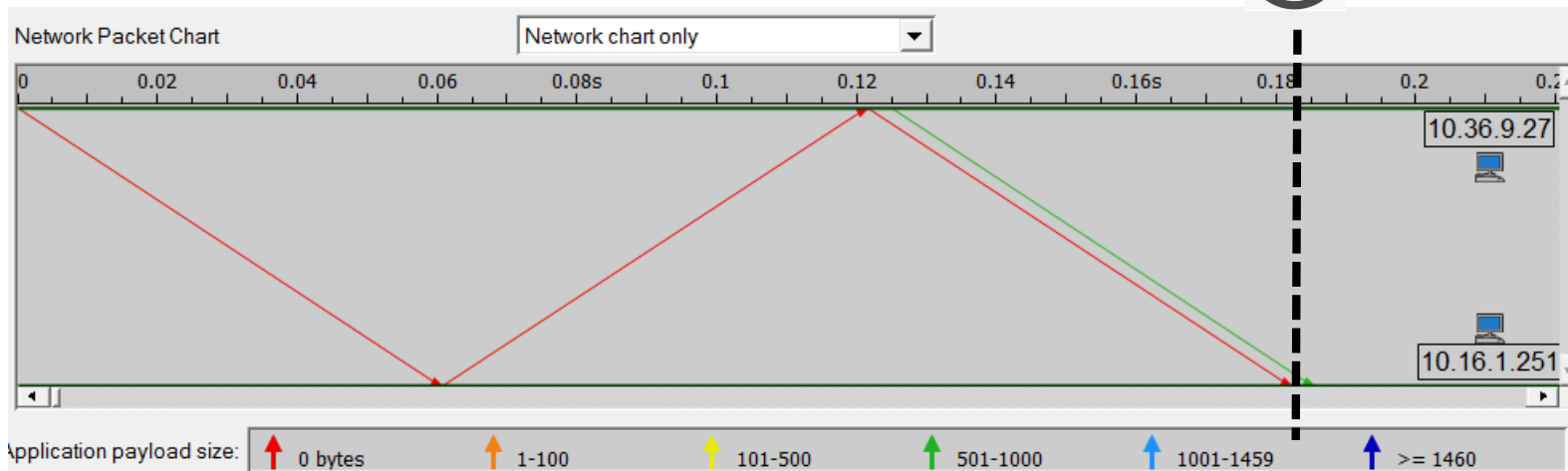
# Split Brain – State Discussion





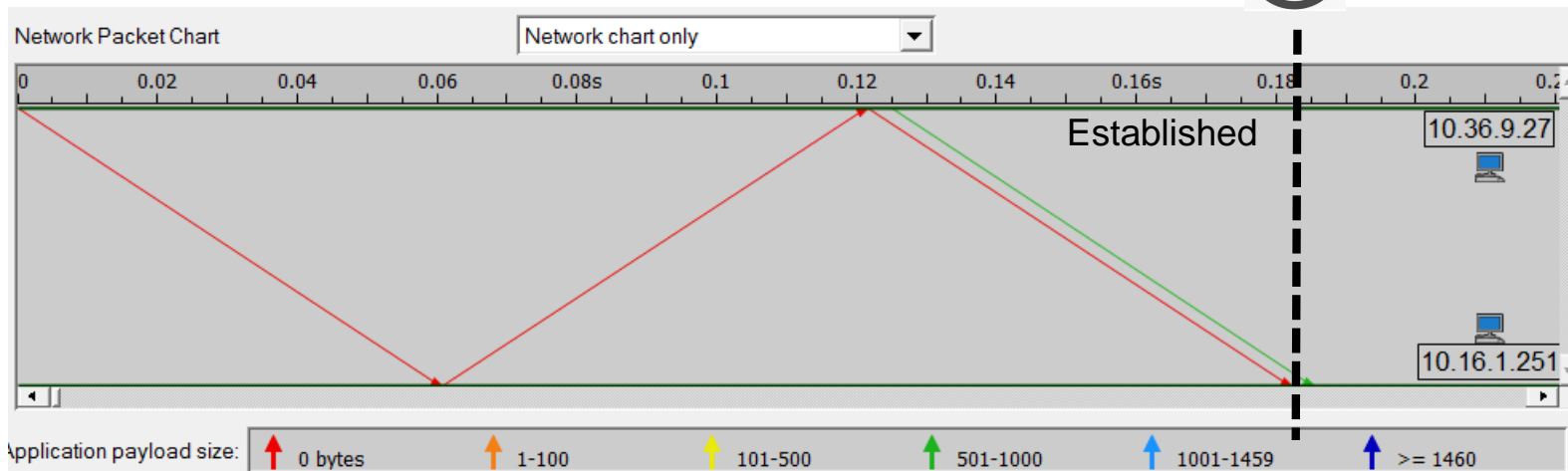


# Split Brain – State Discussion





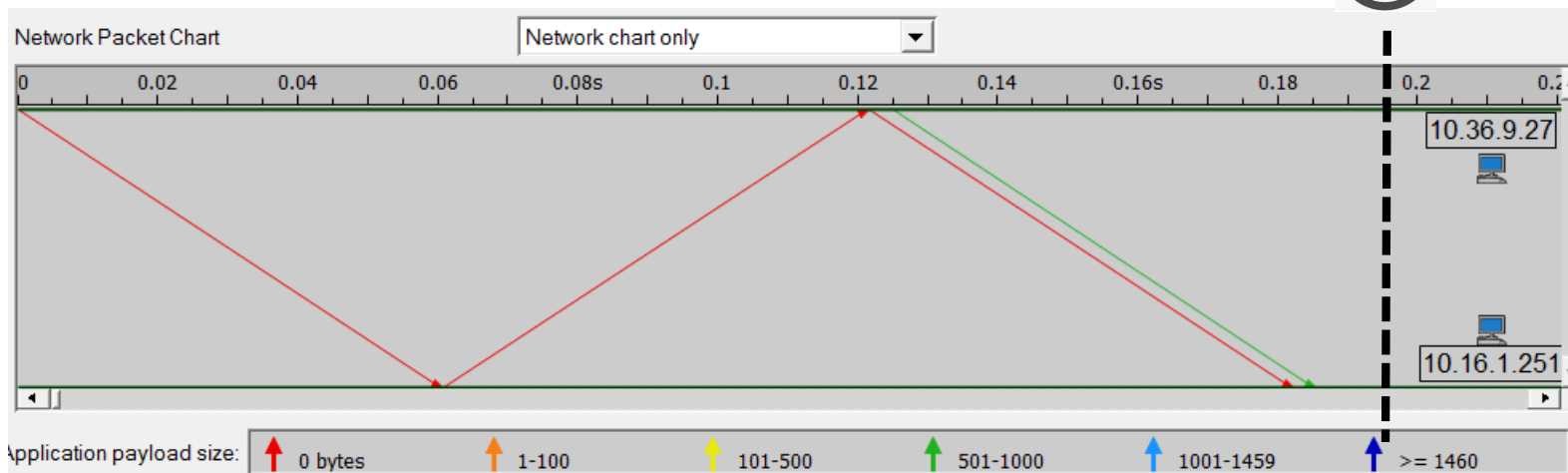
# Split Brain – State Discussion



Established

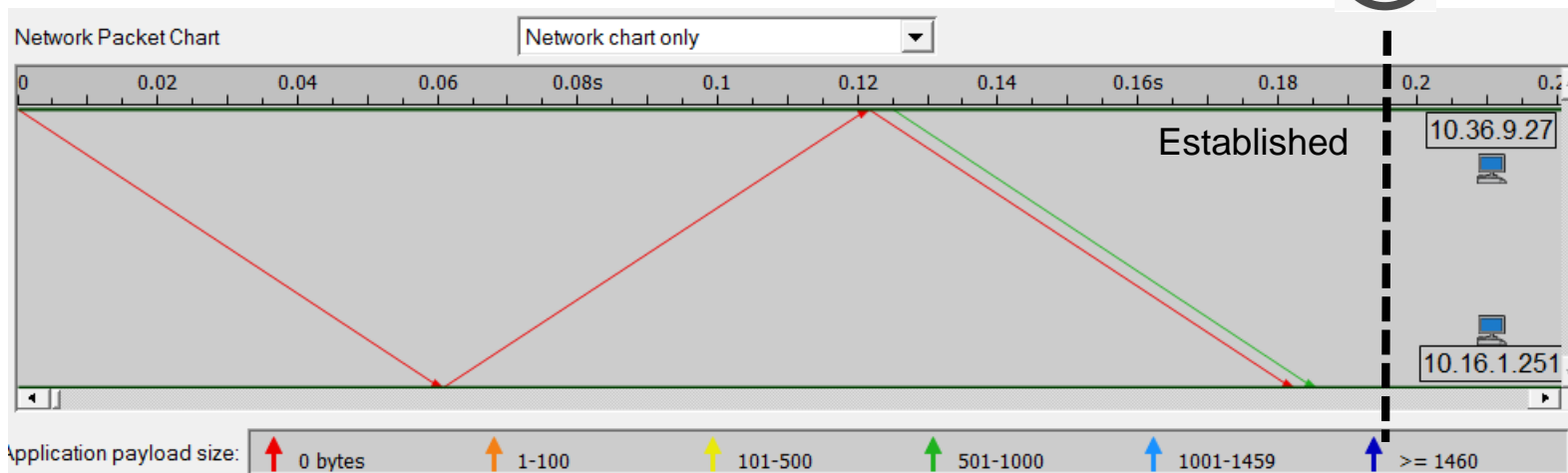


# Split Brain – State Discussion





# Split Brain – State Discussion



Established



# 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...



# Comparison Summary



TCP Split Brain Comparison Summary (Sender vs. Receiver)				
Item	Topic	Summary	Sender	Receiver
1	SYN Options	MSS, Scaling, TimeStamps, SACK	Negotiation & Adapt, MSS=1460, WS=8, SACK	Negotiation & Adapt, MSS=1360, WS=8, SACK
2	Latency	Can be different in each direction	Client iRTT == 121ms	Server iRTT == 129ms
3	TCP State	Different States at startup and shutdown; state change only occurs when packet sent / received	Closed, SYN-Sent, Established	Listen, SYN-Received, Established



# 3rd Leg Completed

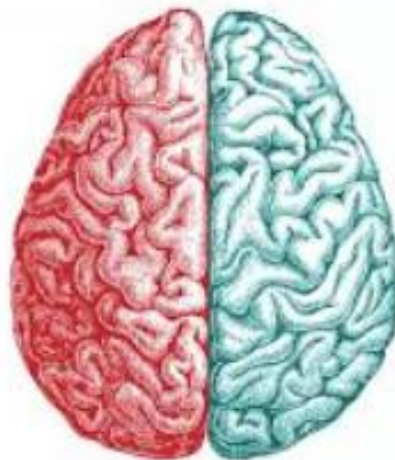




# Split Brain Comparisons



- Next we'll compare Wireshark Expert Info for Client vs. Server capture for a single connection







# Client 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

\* Same results in Wireshark 3.0.5



# Server Conn52942



Wireshark · Expert Information · Server\_side\_shifted\_Conn52942.pcap

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

\* Same results in Wireshark 3.0.5



# 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



Wireshark · Expert Information · GP\_VPN\_Client\_Conn52942.pcap

Severity	Summary	Group	Protocol	Count
Error	TLSCiphertext length MUST NOT exceed $2^{14} + 2048$	Protocol	SSL	2

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

\* Protocol shows as TLS in Wireshark 3.0.5



# Wireshark Updates



- Wireshark 2.6.3 was used for this course
- Found same Expert Info in 3.0.5
- Future versions may handle OOS & SSL reassembly with more tolerance



# Discussion: Fragment Overlap



Wireshark · Expert Information · Server\_side\_shifted\_Conn52942.pcap

Severity	Summary	Group	Protocol	Count
Error	New fragment overlaps old data (retransmission?)	Malformed	TCP	8

- 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?



# Comparison Summary



TCP Split Brain Comparison Summary (Sender vs. Receiver)				
Item	Topic	Summary	Sender	Receiver
4	Expert Info			
4.1	Duplicate ACK	Should be close to the same on both captures		
4.2	OOS		Not expected on sender; but retrans could be interpreted as OOS	Will be higher than sender
4.3	Retransmissions		Will be higher than receiver; could also be flagged as OOS	Might be flagged as OOS
4.4	Previous Segment Not Captured	Can be different	Not expected on sender unless there's a capture integrity issue	Expected on receiver when there's packet loss or OOS
4.5	SSL Errors	Could be side effect of reassembly in presence of OOS	Not expected	Likely to be reassembly issue



# 4th Leg Completed





# Let's drill into one of these...



In the neighborhood of packet #714

Wireshark · Expert Information · Server\_side\_shifted\_Conn52942.pcap

Severity	Summary	Group	Protocol
▲ Error	New fragment overlaps old data (retransmission?)	Malformed	TCP
714	[TCP Fast Retransmission] 443 → 52942 [PSH, ACK] Seq=5...	Malformed	TCP
845	[TCP Fast Retransmission] 443 → 52942 [PSH, ACK] Seq=6...	Malformed	TCP
926	[TCP Fast Retransmission] 443 → 52942 [PSH, ACK] Seq=7...	Malformed	TCP
987	[TCP Fast Retransmission] 443 → 52942 [PSH, ACK] Seq=8...	Malformed	TCP
1231	[TCP Fast Retransmission] 443 → 52942 [PSH, ACK] Seq=1...	Malformed	TCP
1581	[TCP Fast Retransmission] 443 → 52942 [PSH, ACK] Seq=1...	Malformed	TCP



This will be a great opportunity for additional Split Brain comparisons



# Split Brain Comparisons



- Drill down into Fragment Overlap Details – Part I







# Pop Quiz



- What's one of the best things about Wireshark?





# Pop Quiz



- OK..., besides the Developers?
- Totally flexible columns, views, and profiles!!



# Configuration Profile Help



The screenshot shows a help window titled "Wireshark User's Guide". The window has a toolbar with icons for Hide, Previous, Next, Back, Print, and Options. On the left, there is a "Contents" pane with a search box and a tree view of the document's structure. The main content area displays the text for section 10.6, "Configuration Profiles".

## 10.6. Configuration Profiles

Configuration Profiles can be used to configure and use more than one set of preferences and configurations. Select the *Configuration Profiles...* menu item from the *Edit* menu, or simply press Shift-Ctrl-A; and Wireshark will pop up the Configuration Profiles dialog box as shown in [Figure 10.9. "The configuration profiles dialog box"](#). It is also possible to click in the "Profile" part of the statusbar to popup a menu with available Configuration Profiles ([Figure 3.22. "The Statusbar with a configuration profile menu"](#)).

Configuration files stored in the Profiles:

- Preferences (preferences) ([Section 10.5. "Preferences"](#))
- Capture Filters (cfilters) ([Section 6.6. "Defining and saving filters"](#))
- Display Filters (dfilters) ([Section 6.6. "Defining and saving filters"](#))
- Coloring Rules (colorfilters) ([Section 10.3. "Packet colorization"](#))
- Disabled Protocols (disabled\_protos) ([Section 10.4.1. "The "Enabled Protocols" dialog box"](#))
- User Accessible Tables:
  - Custom HTTP headers (custom\_http\_header\_fields)
  - Custom IMF headers (imf\_header\_fields)



# 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



Server\_side\_shifted\_Conn52942.pcap

File Edit View Go Capture Analyze Statistics Telephony Wireless Tools Help

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

Focus is on Frame #714

No.	Time	Source	Destination	Protocol	Length	Info
697	350.989481	10.16.1.251	10.36.9.27	TCP	2638	443 → 52942 [ACK] Seq=518959 Ack=64515 Win=130816 Len=2584 [TCP segment of a reassembled P...
698	350.989502	10.36.9.27	10.16.1.251	TCP	60	52942 → 443 [ACK] Seq=64515 Ack=507101 Win=65792 Len=0
699	350.989514	10.16.1.251	10.36.9.27	TLSv1	2623	Application Data
700	350.989537	10.36.9.27	10.16.1.251	TCP	60	52942 → 443 [ACK] Seq=64515 Ack=509670 Win=65792 Len=0
701	350.989907	10.16.1.251	10.36.9.27	TCP	3930	443 → 52942 [ACK] Seq=524112 Ack=64515 Win=130816 Len=3876 [TCP segment of a reassembled P...
702	351.096868	10.36.9.27	10.16.1.251	TCP	60	52942 → 443 [ACK] Seq=64515 Ack=512254 Win=65792 Len=0
703	351.096869	10.36.9.27	10.16.1.251	TCP	60	52942 → 443 [ACK] Seq=64515 Ack=514838 Win=65792 Len=0
704	351.096921	10.16.1.251	10.36.9.27	TLSv1	5207	Application Data
705	351.101927	10.36.9.27	10.16.1.251	TCP	60	52942 → 443 [ACK] Seq=64515 Ack=516375 Win=65792 Len=0
706	351.101974	10.36.9.27	10.16.1.251	TCP	60	52942 → 443 [ACK] Seq=64515 Ack=518959 Win=65792 Len=0
707	351.102241	10.16.1.251	10.36.9.27	TCP	3930	443 → 52942 [ACK] Seq=533141 Ack=64515 Win=130816 Len=3876 [TCP segment of a reassembled P...
708	351.105948	10.36.9.27	10.16.1.251	TCP	60	52942 → 443 [ACK] Seq=64515 Ack=521543 Win=65792 Len=0
709	351.105972	10.16.1.251	10.36.9.27	TLSv1	1591	Application Data
710	351.110879	10.36.9.27	10.16.1.251	TCP	66	52942 → 443 [ACK] Seq=64515 Ack=522835 Win=65792 Len=0 SLE=524112 SRE=525404
711	351.110881	10.36.9.27	10.16.1.251	TCP	66	[TCP Dup ACK 710#1] 52942 → 443 [ACK] Seq=64515 Ack=522835 Win=65792 Len=0 SLE=524112 SRE=525404
712	351.110929	10.36.9.27	10.16.1.251	TCP	66	[TCP Dup ACK 710#2] 52942 → 443 [ACK] Seq=64515 Ack=522835 Win=65792 Len=0 SLE=524112 SRE=525404
713	351.214132	10.36.9.27	10.16.1.251	TCP	66	[TCP Dup ACK 710#3] 52942 → 443 [ACK] Seq=64515 Ack=522835 Win=65792 Len=0 SLE=524112 SRE=525404
714	351.214191	10.16.1.251	10.36.9.27	TCP	1346	[TCP Fast Retransmission] 443 → 52942 [PSH, ACK] Seq=522835 Ack=64515 Win=130816 Len=1292 [TCP segment of a reassembled P...
715	351.214233	10.36.9.27	10.16.1.251	TCP	66	[TCP Dup ACK 710#4] 52942 → 443 [ACK] Seq=64515 Ack=522835 Win=65792 Len=0 SLE=524112 SRE=525404
716	351.214235	10.36.9.27	10.16.1.251	TCP	66	[TCP Dup ACK 710#5] 52942 → 443 [ACK] Seq=64515 Ack=522835 Win=65792 Len=0 SLE=524112 SRE=525404



# View with Classic Profile



Server\_side\_shifted\_Conn52942.pcap

File Edit View Go Capture Analyze Statistics Telephony Wireless Tools Help

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

No.	Time	Source	Destination	Protocol	Length	Info
697	350.989481	10.16.1.251	10.36.9.27	TCP	2638	443 → 52942 [ACK] Seq=518959 Ack=64515 Win=130816 Len=2584 [TCP segment of a reassembled P...
698	350.989502	10.36.9.27	10.16.1.251	TCP	60	52942 → 443 [ACK] Seq=64515 Ack=507101 Win=65792 Len=0
699	350.989514	10.16.1.251	10.36.9.27	TLSv1	2623	Application Data
700	350.989537	10.36.9.27	10.16.1.251	TCP	60	52942 → 443 [ACK] Seq=5...
701	350.989907	10.16.1.251	10.36.9.27	TCP	3930	443 → 52942 [ACK] Seq=5...
702	351.096868	10.36.9.27	10.16.1.251	TCP	60	52942 → 443 [ACK] Seq=6...
703	351.096869	10.36.9.27	10.16.1.251	TCP	60	52942 → 443 [ACK] Seq=6...
704	351.096921	10.16.1.251	10.36.9.27	TLSv1	5207	Application Data
705	351.101927	10.36.9.27	10.16.1.251	TCP	60	52942 → 443 [ACK] Seq=6...
706	351.101974	10.36.9.27	10.16.1.251	TCP	60	52942 → 443 [ACK] Seq=6...
707	351.102241	10.16.1.251	10.36.9.27	TCP	3930	443 → 52942 [ACK] Seq=5...
708	351.105948	10.36.9.27	10.16.1.251	TCP	60	52942 → 443 [ACK] Seq=64515 Ack=521543 Win=65792 Len=0
709	351.105972	10.16.1.251	10.36.9.27	TLSv1	1591	Application Data
710	351.110879	10.36.9.27	10.16.1.251	TCP	66	52942 → 443 [ACK] Seq=64515 Ack=522835 Win=65792 Len=0 SLE=524112 SRE=525404
711	351.110881	10.36.9.27	10.16.1.251	TCP	66	[TCP Dup ACK 710#1] 52942 → 443 [ACK] Seq=64515 Ack=522835 Win=65792 Len=0 SLE=524112 SRE=...
712	351.110929	10.36.9.27	10.16.1.251	TCP	66	[TCP Dup ACK 710#2] 52942 → 443 [ACK] Seq=64515 Ack=522835 Win=65792 Len=0 SLE=524112 SRE=...
713	351.214132	10.36.9.27	10.16.1.251	TCP	66	[TCP Dup ACK 710#3] 52942 → 443 [ACK] Seq=64515 Ack=522835 Win=65792 Len=0 SLE=524112 SRE=...
714	351.214191	10.16.1.251	10.36.9.27	TCP	1346	[TCP Fast Retransmission] 443 → 52942 [PSH, ACK] Seq=522835 Ack=64515 Win=130816 Len=1292 [F...
715	351.214233	10.36.9.27	10.16.1.251	TCP	66	[TCP Dup ACK 710#4] 52942 → 443 [ACK] Seq=64515 Ack=522835 Win=65792 Len=0 SLE=524112 SRE=...
716	351.214235	10.36.9.27	10.16.1.251	TCP	66	[TCP Dup ACK 710#5] 52942 → 443 [ACK] Seq=64515 Ack=522835 Win=65792 Len=0 SLE=524112 SRE=...

What columns do we need in our view to better understand Fragment Overlap?





# Profile SB-Seq Analysis



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	Protocol	Length	Sequence number	Next sequence number	Info
699	350.989514	0.000012000	10.16.1.251	10.36.9.27	TLSv1	2623	521543	524112	Application Data
700	350.989537	0.000023000	10.36.9.27	10.16.1.251	TCP	60	64515	64515	52942 → 443 [ACK] Seq=64515 Ack=509670 Win=6579:
701	350.989907	0.000370000	10.16.1.251	10.36.9.27	TCP	3930	524112	527988	443 → 52942 [ACK] Seq=524112 Ack=64515 Win=1308:
702	351.096868	0.106961000	10.36.9.27	10.16.1.251	TCP	60	64515	64515	52942 → 443 [ACK] Seq=64515 Ack=512254 Win=6579:
703	351.096869	0.000001000	10.36.9.27	10.16.1.251	TCP	60	64515	64515	52942 → 443 [ACK] Seq=64515 Ack=514838 Win=6579:
704	351.096921	0.000052000	10.16.1.251	10.36.9.27	TLSv1	5207	527988	533141	Application Data
705	351.101927	0.005006000	10.36.9.27	10.16.1.251	TCP	60	64515	64515	52942 → 443 [ACK] Seq=64515 Ack=516375 Win=6579:
706	351.101974	0.000047000	10.36.9.27	10.16.1.251	TCP	60	64515	64515	52942 → 443 [ACK] Seq=64515 Ack=518959 Win=6579:
707	351.102241	0.000267000	10.16.1.251	10.36.9.27	TCP	3930	533141	537017	443 → 52942 [ACK] Seq=533141 Ack=64515 Win=1308:
708	351.105948	0.003707000	10.36.9.27	10.16.1.251	TCP	60	64515	64515	52942 → 443 [ACK] Seq=64515 Ack=521543 Win=6579:
709	351.105972	0.000024000	10.16.1.251	10.36.9.27	TLSv1	1591	537017	538554	Application Data
710	351.110879	0.004907000	10.36.9.27	10.16.1.251	TCP	66	64515	64515	52942 → 443 [ACK] Seq=64515 Ack=522835 Win=6579:
711	351.110881	0.000002000	10.36.9.27	10.16.1.251	TCP	66	64515	64515	[TCP Dup ACK 710#1] 52942 → 443 [ACK] Seq=64515
712	351.110929	0.000048000	10.36.9.27	10.16.1.251	TCP	66	64515	64515	[TCP Dup ACK 710#2] 52942 → 443 [ACK] Seq=64515
713	351.214132	0.103203000	10.36.9.27	10.16.1.251	TCP	66	64515	64515	[TCP Dup ACK 710#3] 52942 → 443 [ACK] Seq=64515
714	351.214191	0.000059000	10.16.1.251	10.36.9.27	TCP	1346	522835	524127	[TCP Fast Retransmission] 443 → 52942 [PSH, ACK
715	351.214233	0.000042000	10.36.9.27	10.16.1.251	TCP	66	64515	64515	[TCP Dup ACK 710#4] 52942 → 443 [ACK] Seq=64515



# Pop Quiz



- What are Jumbo frames?







# Pop Quiz



- What are Jumbo frames?
- What are Super Jumbo Frames?





# Pop Quiz



- When is a Jumbo frame not a Jumbo Frame?



# Are these "Jumbo" frames?



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	Protocol	Length	Sequence number	Next sequence number	Info
699	350.989514	0.000012000		10.16.1.251	10.36.9.27	TLSv1	2623	521543	524112	Application Data
700	350.989537	0.000023000		10.36.9.27	10.16.1.251	TCP	60	64515	64515	52942 → 443 [ACK] Seq=64515 Ack=509670 Win=6579:
701	350.989907	0.000370000		10.16.1.251	10.36.9.27	TCP	3930	524112	527988	443 → 52942 [ACK] Seq=524112 Ack=64515 Win=1308:
702	351.096868	0.106961000		10.36.9.27	10.16.1.251	TCP	60	64515	64515	52942 → 443 [ACK] Seq=64515 Ack=512254 Win=6579:
703	351.096869	0.000001000		10.36.9.27	10.16.1.251	TCP	60	64515	64515	52942 → 443 [ACK] Seq=64515 Ack=514838 Win=6579:
704	351.096921	0.000052000		10.16.1.251	10.36.9.27	TLSv1	5207	527988	533141	Application Data
705	351.101927	0.005006000		10.36.9.27	10.16.1.251	TCP	60	64515	64515	52942 → 443 [ACK] Seq=64515 Ack=516375 Win=6579:
706	351.101974	0.000047000		10.36.9.27	10.16.1.251	TCP	60	64515	64515	52942 → 443 [ACK] Seq=64515 Ack=518959 Win=6579:
707	351.102241	0.000267000		10.16.1.251	10.36.9.27	TCP	3930	533141	537017	443 → 52942 [ACK] Seq=533141 Ack=64515 Win=1308:
708	351.105948	0.003707000		10.36.9.27	10.16.1.251	TCP	60	64515	64515	52942 → 443 [ACK] Seq=64515 Ack=521543 Win=6579:
709	351.105972	0.000024000		10.16.1.251	10.36.9.27	TLSv1	1591	537017	538554	Application Data
710	351.110879	0.004907000		10.36.9.27	10.16.1.251	TCP	66	64515	64515	52942 → 443 [ACK] Seq=64515 Ack=522835 Win=6579:
711	351.110881	0.000002000		10.36.9.27	10.16.1.251	TCP	66	64515	64515	[TCP Dup ACK 710#1] 52942 → 443 [ACK] Seq=64515
712	351.110929	0.000048000		10.36.9.27	10.16.1.251	TCP	66	64515	64515	[TCP Dup ACK 710#2] 52942 → 443 [ACK] Seq=64515
713	351.214132	0.103203000		10.36.9.27	10.16.1.251	TCP	66	64515	64515	[TCP Dup ACK 710#3] 52942 → 443 [ACK] Seq=64515
714	351.214191	0.000059000		10.16.1.251	10.36.9.27	TCP	1346	522835	524127	[TCP Fast Retransmission] 443 → 52942 [PSH, ACK
715	351.214233	0.000042000		10.36.9.27	10.16.1.251	TCP	66	64515	64515	[TCP Dup ACK 710#4] 52942 → 443 [ACK] Seq=64515



# TCP Large Send Offload



Google

tcp large send offload



7



All

Shopping

News

Images

Videos

More

Settings

Tools

About 205,000 results (0.50 seconds)

## Large Send Offload and Network Performance | Peer Wisdom

[www.peerwisdom.org/2013/04/03/large-send-offload-and-network-performance/](http://www.peerwisdom.org/2013/04/03/large-send-offload-and-network-performance/) ▼

Apr 3, 2013 - So what is **Large Send Offload** (also known as Large Segmentation ... that allows the TCP/IP network stack to build a large TCP message of up to ...

## Disabling Large Send Offload – Windows | Peer Wisdom

[www.peerwisdom.org/2013/04/25/disabling-large-send-offload-windows/](http://www.peerwisdom.org/2013/04/25/disabling-large-send-offload-windows/) ▼

Apr 25, 2013 - In an earlier post, I described the **Large Send Offload** (LSO) feature of modern ... I'll start with disabling LSO in the TCP/IP network stack since ...

## Large send offload - Wikipedia

[https://en.wikipedia.org/wiki/Large\\_send\\_offload](https://en.wikipedia.org/wiki/Large_send_offload) ▼

In computer networking, **large send offload** (LSO) is a technique for increasing egress ... The technique is also called **TCP segmentation offload** (TSO) when applied to **TCP**, or generic segmentation offload (GSO). The inbound counterpart of ...

## Large Send Offload causes performance and slowdown issues

<https://www.bitdefender.com/.../large-send-offload-causes-performance-and-slowdown...> ▼

**Large Send Offload** causes performance and slowdown issues ... initialization or when an interface appears as a Plug and Play event, the TCP/IP driver will ...

## Large send offload



In computer networking, large send offload is a technique for increasing egress throughput of high-bandwidth network connections by reducing CPU overhead. It works by passing a multipacket buffer to the network interface card. The NIC then splits this buffer into separate packets.  
[Wikipedia](#)

[Feedback](#)



# 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 running on a blade chassis technology (aka Blade Server)?



# TCP Large Receive Offload





[Web](#) [Images](#) [Videos](#) [News](#)

All Regions ▾ Safe Search: Moderate ▾ Any Time ▾


### Large receive offload - Wikipedia

In computer networking, **large receive offload** (LRO) is a technique for increasing inbound throughput of high-bandwidth network connections by reducing CPU overhead.

W [https://en.wikipedia.org/wiki/Large\\_receive\\_offload](https://en.wikipedia.org/wiki/Large_receive_offload)


### Large Receive Offload (LRO) Support for ... - VMware VROOM! Blog

**Large Receive Offload** (LRO) is a technique to reduce the CPU time for processing TCP packets that arrive from the network at a high rate. LRO reassembles incoming packets into larger ones (but fewer packets) to deliver them to the network stack of the system.

 <https://blogs.vmware.com/performance/2015/06/vmxnet3-lro.html>


### Large Receive Offload - VMware Docs Home

**Large Receive Offload** Use **Large Receive Offload** (LRO) to reduce the CPU overhead for processing packets that arrive from the network at a high rate. LRO reassembles incoming network packets into larger buffers and transfers the resulting larger but fewer packets to the network stack of the host or virtual machine.

 <https://docs.vmware.com/en/VMware-vSphere/6.5/com.vmware.vsphere...>

### How To Enable Large Receive Offload (LRO) | Mellanox ...

**large-receive-offload**: on Generic **Receive Offload** (GRO) When "hw\_lro" flag cannot be found on a new kernel (LRO type is hardware), packets aggregation can be done using the GRO feature via ethtool.

 <https://community.mellanox.com/docs/DOC-2814>

### Large receive offload

In computer networking, large receive offload is a technique for increasing inbound throughput of high-bandwidth network connections by reducing CPU overhead. It works by aggregating multiple incoming packets from a single stream into a larger buffer before they are passed higher up the networking stack, thus reducing the number of packets that have to be processed. Linux implementations generally use LRO in conjunction with the New API to also reduce the number of interrupts.

W [More at Wikipedia](#)

[Feedback](#)



# Back to Fragment Overlap







# Here is the overlap



FRAME #714

Server\_side\_shifted\_Conn52942.pcap

File Edit View Go Capture Analyze Statistics Telephony Wireless Tools Help

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

No.	Time	Delta Prev	Source	Destination	TCP Len	Sequence number	Next seq	Ack	Bytes in flight	Info
699	350.989514	0.000012000	10.16.1.251	10.36.9.27	2569	521543	524112	64515	17011	Application Data
700	350.989537	0.000023000	10.36.9.27	10.16.1.251	0	64515	64515	509670		52942 → 443 [ACK] Seq=645
701	350.989907	0.000370000	10.16.1.251	10.36.9.27	3876	524112	527988	64515	18318	443 → 52942 [ACK] Seq=524
702	351.096868	0.106961000	10.36.9.27	10.16.1.251	0	64515	64515	512254		52942 → 443 [ACK] Seq=645
703	351.096869	0.000001000	10.36.9.27	10.16.1.251	0	64515	64515	514838		52942 → 443 [ACK] Seq=645
704	351.096921	0.000052000	10.16.1.251	10.36.9.27	5153	527988	533141	64515	18303	Application Data
705	351.101927	0.005006000	10.36.9.27	10.16.1.251	0	64515	64515	516375		52942 → 443 [ACK] Seq=645
706	351.101974	0.000047000	10.36.9.27	10.16.1.251	0	64515	64515	518959		52942 → 443 [ACK] Seq=645
707	351.102241	0.000267000	10.16.1.251	10.36.9.27	3876	533141	537017	64515	18058	443 → 52942 [ACK] Seq=533
708	351.105948	0.003707000	10.36.9.27	10.16.1.251	0	64515	64515	521543		52942 → 443 [ACK] Seq=645
709	351.105972	0.000024000	10.16.1.251	10.36.9.27	1537	537017	538554	64515	17011	Application Data
710	351.110879	0.004907000	10.36.9.27	10.16.1.251	0	64515	64515	522835		52942 → 443 [ACK] Seq=645
711	351.110881	0.000002000	10.36.9.27	10.16.1.251	0	64515	64515	522835		[TCP Dup ACK 710#1] 52942
712	351.110929	0.000048000	10.36.9.27	10.16.1.251	0	64515	64515	522835		[TCP Dup ACK 710#2] 52942
713	351.214132	0.103203000	10.36.9.27	10.16.1.251	0	64515	64515	522835		[TCP Dup ACK 710#3] 52942
714	351.214191	0.000059000	10.16.1.251	10.36.9.27	1292	522835	524127	64515	15719	[TCP Fast Retransmission]
715	351.214233	0.000042000	10.36.9.27	10.16.1.251	0	64515	64515	522835		[TCP Dup ACK 710#4] 52942





# Server Side Stream Visual



Transmit Original Segment

SEQ 521543

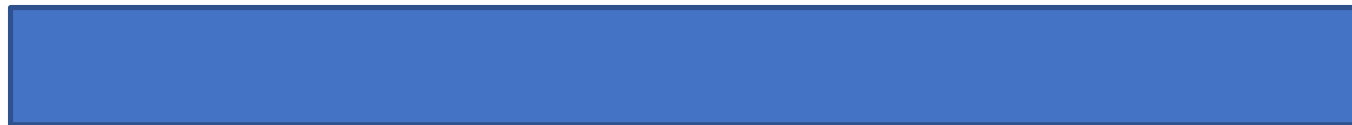
524111



2,569 Stream Bytes

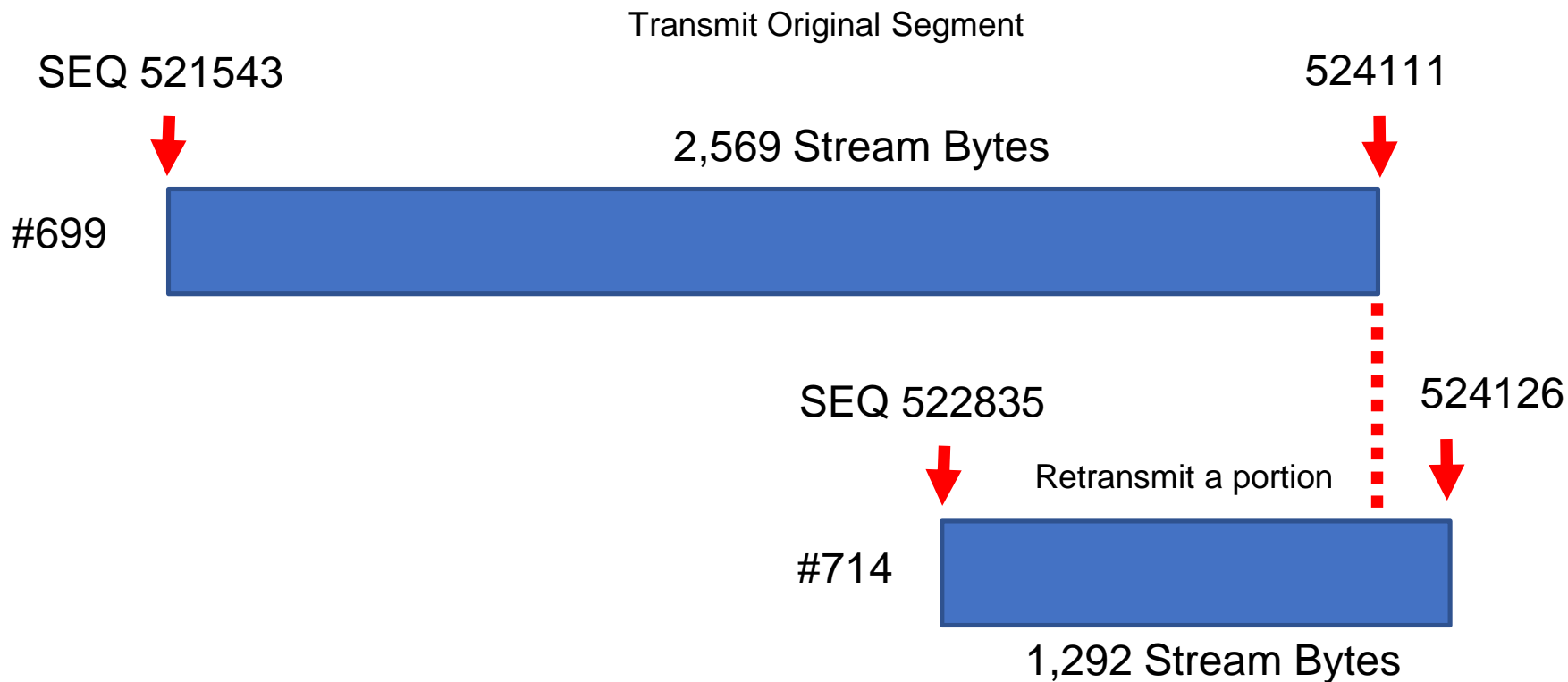


#699





# 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



521543



2,569 Stream Bytes

524111





# Several Observations



GP\_VPN\_Client\_Conn52942.pcap

File Edit View Go Capture Analyze Statistics Telephony Wireless Tools Help

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

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] 443 → 52942 [ACK] Seq=524112 Ack=64515.
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] Seq=64515 Ack=522835.
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



GP\_VPN\_Client\_Conn52942.pcap

File Edit View Go Capture Analyze Statistics Telephony Wireless Tools Help

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

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] 443 → 52942 [ACK] Seq=524112 Ack=64515
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] Seq=64515 Ack=522835
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?



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	200	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...

- 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...



# Break Time



- 15 Minute Break
- When we return we'll switch to Session #14
- TCP Split Brain Part II Deck





# 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 “sender perspective” and “receiver perspective” to fully interpret complex situations



# Summary Results



TCP Split Brain Comparison Summary (Sender vs. Receiver)				
Item	Topic	Summary	Sender	Receiver
1	SYN Options	MSS, Scaling, TimeStamps, SACK	Negotiation & Adapt, MSS=1460, WS=8, SACK	Negotiation & Adapt, MSS=1360, WS=8, SACK
2	Latency	Can be different in each direction	Client IRTT == 121ms	Server IRTT == 129ms
3	TCP State	Different States at startup and shutdown; state change only occurs when packet sent / received	Closed, SYN-Sent, Established	Listen, SYN-Received, Established
4	Expert Info			
4.1	Duplicate ACK	Should be close to the same on both captures		
4.2	OOS		Not expected on sender; but retrans could be interpreted as OOS	Will be higher than sender
4.3	Retransmissions		Will be higher than receiver; could also be flagged as OOS	Might be flagged as OOS
4.4	Previous Segement Not Captured	Can be different	Not expected on sender unless there's a capture integrity issue	Expected on receiver when there's packet loss or OOS
4.5	SSL Errors	Could be side effect of reassembly in presence of OOS	Not expected	Likely to be reassembly issue
4.6	Fragment Overlap	Most likely caused by Segmentation Offoad	Most likely to show up on Sender's capture	Receiver could flag overlap using SACK field
5	Frame Sizes	Can be different	Effects of LSO / IP Fragmentation	Effects of LRO
6	Display Time Delta	Very unique to each endpoint	ACKs will usually apply to segments sent much earlier in time	Interpretation can seem confusing, especially when lots of packets are in flight
7	Bytes In-Flight	OOS and retransmissions may impact calculation	Should be pretty accurate	Generally not very interesting from receiver's capture
8	RTT2ACK		Can include receiver's Delayed ACK Timer; never less than 1 RTT	Generally Super Fast! When less than Delayed ACK Timer, could reflect degraded condition for TCP stack
9	Congestion	Requires advanced analytics		
10	Service Response Time		Client: May include latency + protocol delay + congestion	Server: should be most accurate



# Final 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...
- Please join us for Part II after the break



# Thank You!



- For your participation!
- Hope you found this session informative and insightful
- Hope you'll join us for Part II – after the break
- We'll continue our Split Brain deep dive into compare Bytes in Flight, RTT2ACK, followed by Congestion

