The Network is Slow!

Finding the Root Cause of Slow Application Performance

Lorna Robertshaw

9 November 2017

Goal of this class

- Many classes at SharkFest teach nuances of TCP, spotting retransmissions, expert level analysis...
- This class teaches some (relatively) simple, repeatable ways to quickly identify or rule out common performance problems
- Focus is on HTTPS/TLS
- Assumption is that you can’t always decrypt or capture in many places

About me…

- Lorna Robertshaw
- Based in Leicester, UK
- 16 years experience at OPNET and Riverbed, as trainer and SE for APM/NPM products
- Currently funemployed
- Lots of experience with:
    - Riverbed Transaction Analyzer (ACE)
    - Packet Analyzer (Pilot)
    - AppResponse (ARX, Shark)
- Moderate experience with Wireshark
- https://www.linkedin.com/in/lornarobertshaw/

Pareto Principle

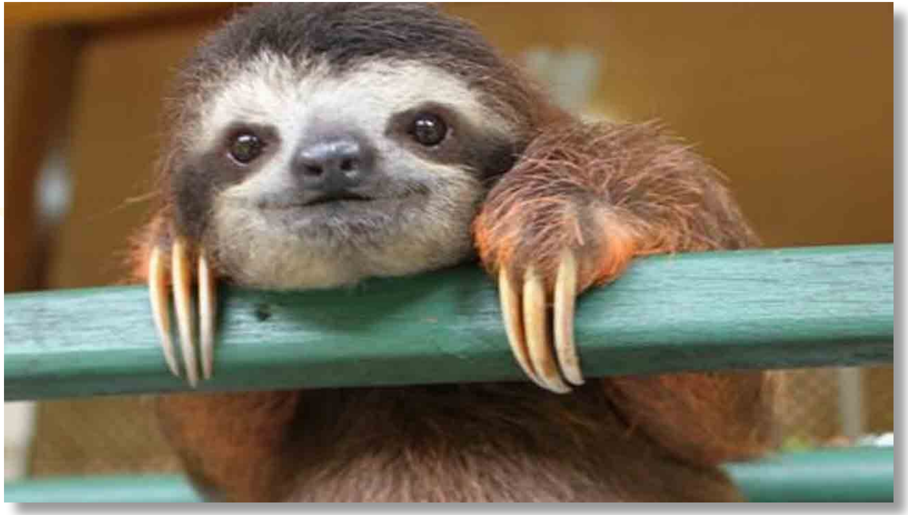
- 80% of the results come from 20% of the effort
- **80%* of slow application troubleshooting cases do not require expert level packet analysis**
- However…
  - The network always gets the blame, so network engineers use network tools
  - Visibility is becoming more of a challenge
  - Packet analysis gets results
- My proposal: rather than dive into detailed TCP analysis, do low effort analysis first

* 100%** of percentages in this presentation are completely made up

** Except that one, that one is accurate

Common Problem #1: Large Client/Server Delay(s)

- Symptoms: Big gap(s) in application layer communication between client and server, but continued TCP communication (ACKs, Keepalives, etc.)
- Goal: Show decisively that delay was not on network, but on server side or client side
- If client side, try to determine what caused the delay



Common Problem #2: Huge/complex/chatty communications

- Examples:
  - Enormous web pages
  - Client establishing connections with large number of servers
  - Sending too much data given what application is trying to accomplish
  - Thousands of application turns (chatty application)
- Goal: illustrate how much data is being sent and that normal/typical conditions may result in slow response times

Common Problem #3: Timeouts and Failures

- DNS Failures/Errors
- Connecting to wrong server
- Failed connections
- Server prematurely terminates connection
- Something times out in backend but transaction still completes

Where to Capture First?

- Best place = local to frontend server
- Easiest place may be on client side

- Strategy for capture on client:
  - Capture wide open and use filters afterwards
  - Isolate single transaction
  - Use markers/pings
  - Web: Capture while using browser's Developer Tools

When Should You Analyze a Capture on the Client with No Filter?

- To quickly validate the IPs/ports/protocols in use
- When only certain users experience problem
- When filtered captures don’t show the problem
- When server side or filtered capture shows no network or server issues

Case Study: PEBKAC Problem

- Carol is in Maryland, using thick-client Oracle system hosted in New York Data Center
- "Every few minutes, Oracle freezes, then starts working again."
- Other users not experiencing this problem
- Data Center side Riverbed AppResponse (packet capture and analysis) appliance shows no issues with Oracle Server/network traffic
- Desktop support can't find issue on Carol's PC

Case Study: PEBKAC Problem

- Set up continuous packet capture on Carol's PC with no capture filter
- Carol calls back when problem next happens
- Import 20 minute time slice into Riverbed Transaction Analyzer

The Culprit …

Carol was storing her music on a remote file server and playing it with Windows Media Player. Each time a new song buffered, her connection with Oracle suffered and Oracle appeared to freeze.

Helpful Tools

Browser Developer Tools

- Statistics:
  - Number of objects on page
  - Number of objects in cache
  - Load time
- List of servers and objects
- Examine biggest objects
- Examine objects that took longest to load
  - Breakdown of Delay
  - Information helpful for finding object load in Wireshark

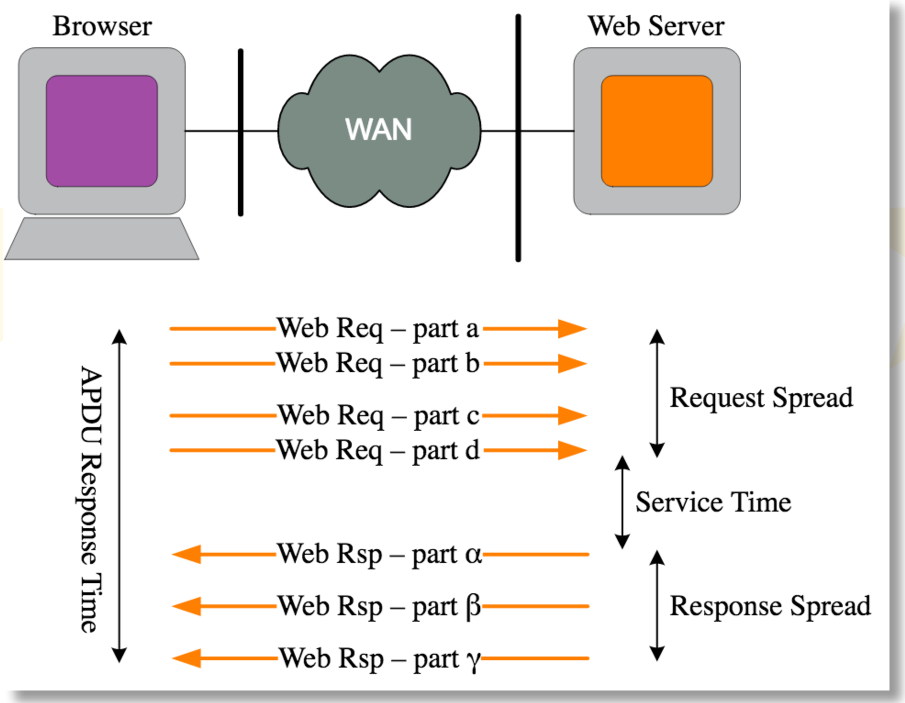
Timings

The Timings tab breaks a network request down into the following subset of the stages defined in the HTTP Archive specification:

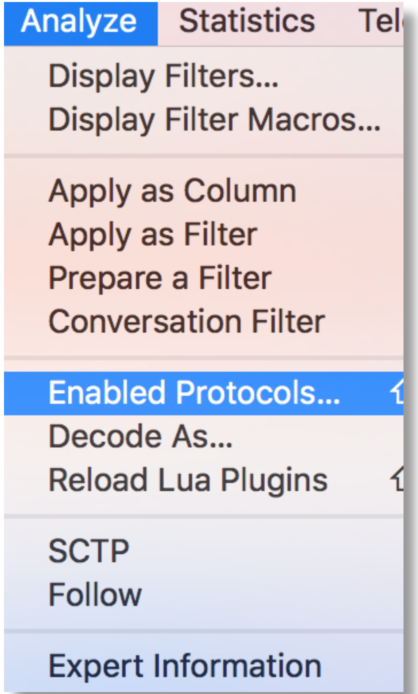
Name	Description
Blocked	Time spent in a queue waiting for a network connection. The browser imposes a limit on the number of simultaneous connections that can be made to a single server. In Firefox this defaults to 6, but can be changed using the `network.http.max-persistent-connections-per-server` preference. If all connections are in use, the browser can't download more resources until a connection is released.
DNS resolution	Time taken to resolve a host name.
Connecting	Time taken to create a TCP connection.
Sending	Time taken to send the HTTP request to the server.
Waiting	Waiting for a response from the server.
Receiving	Time taken to read the entire response from the server (or cache).

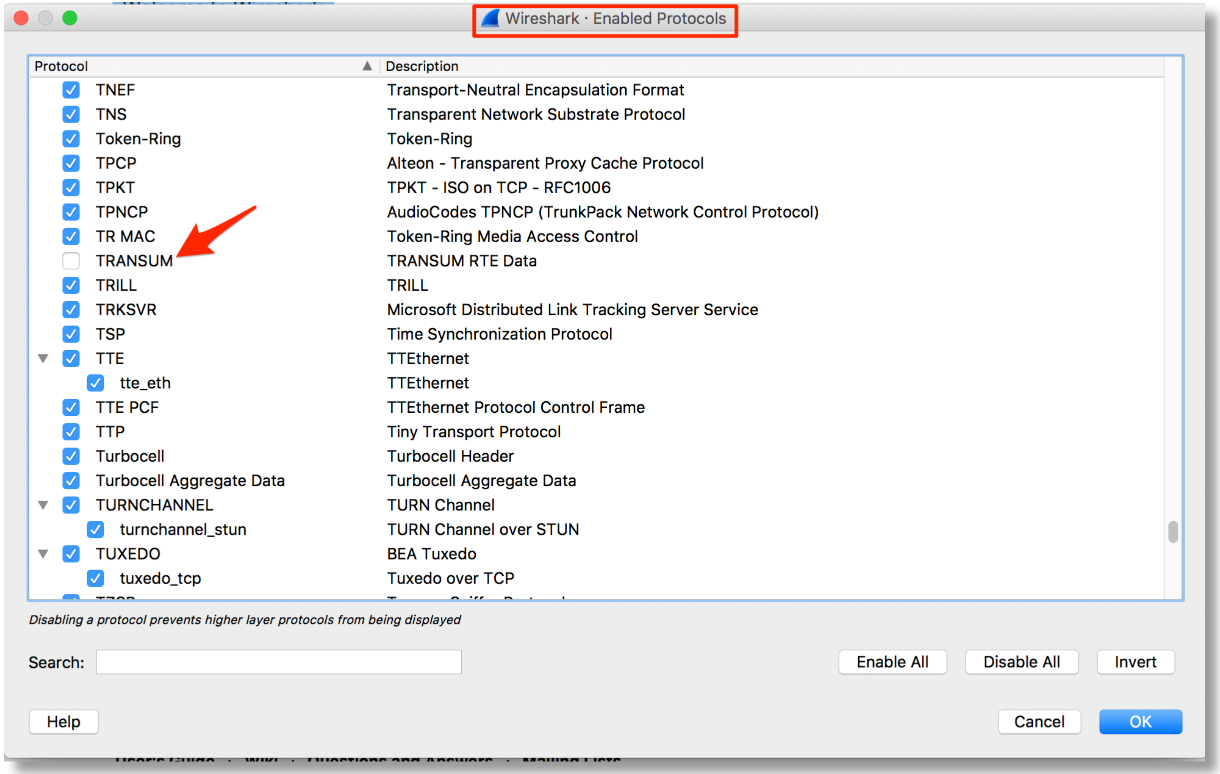
TRANSUM

- New Wireshark option in 2.4
- Plug-in for 2.2
- Provides Response Time metrics for *HTTP, HTTPS, Windows Fileserver SMB2, Microsoft SQL TDS, Oracle SQL TNS, .NET Remoting, SOAP, DCE-RPC (including MS-RPC used by Microsoft Exchange), Kerberos, FTP, TELNET, DNS and many proprietary protocols.*
- *More info at https://community.tribelab.com/*



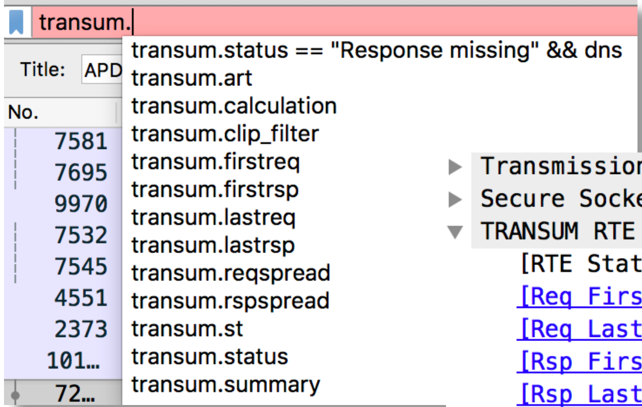
Enable TRANSUM (WS 2.4)





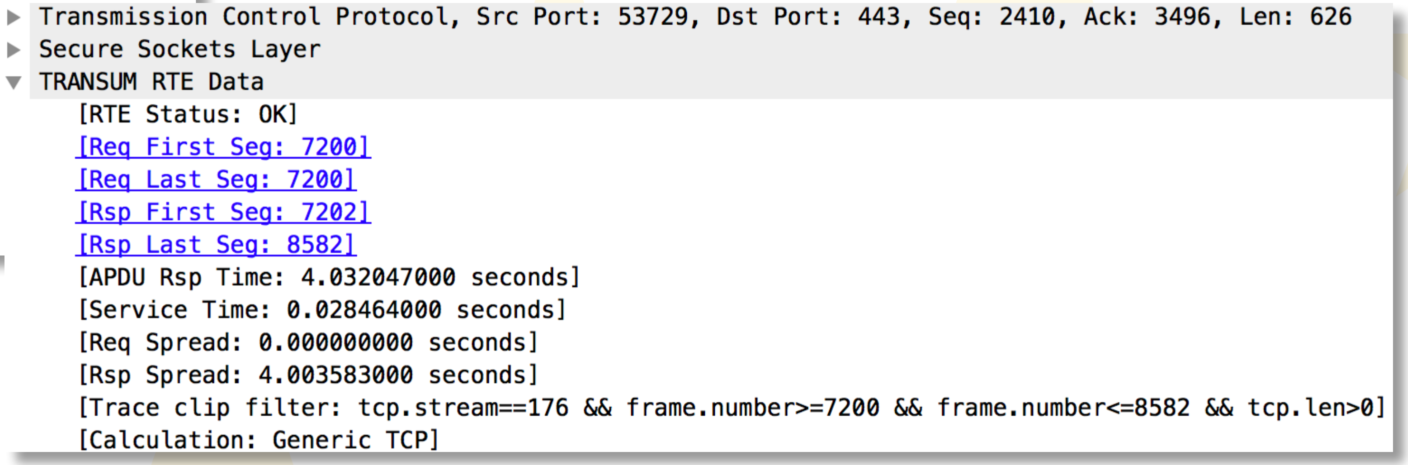
TRANSUM Stats

Display filter/column names

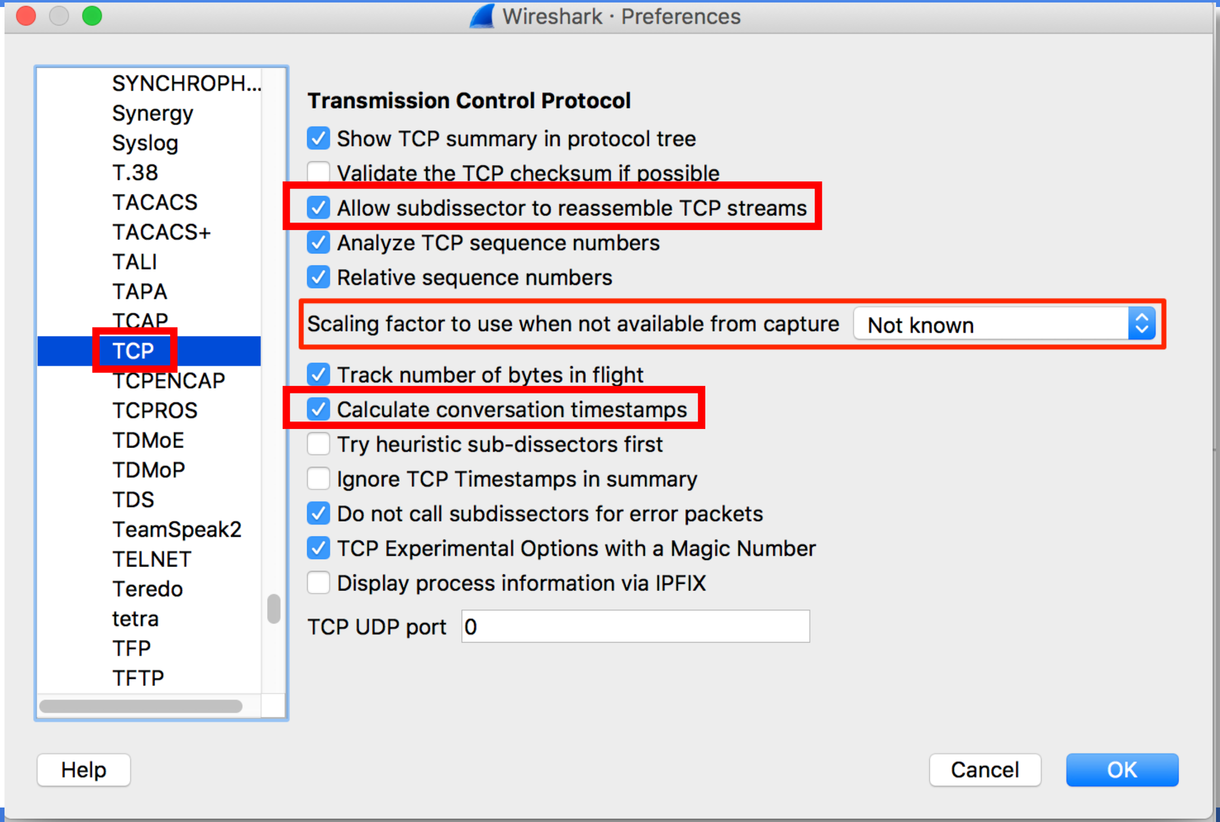


Note: APDU Response Time will be set on the request, not the response

TRANSUM stats in decodes



TCP Protocol Preferences



Preferences are associated with a PROFILE

You need these settings in your active profile for the filters/columns in this class to work

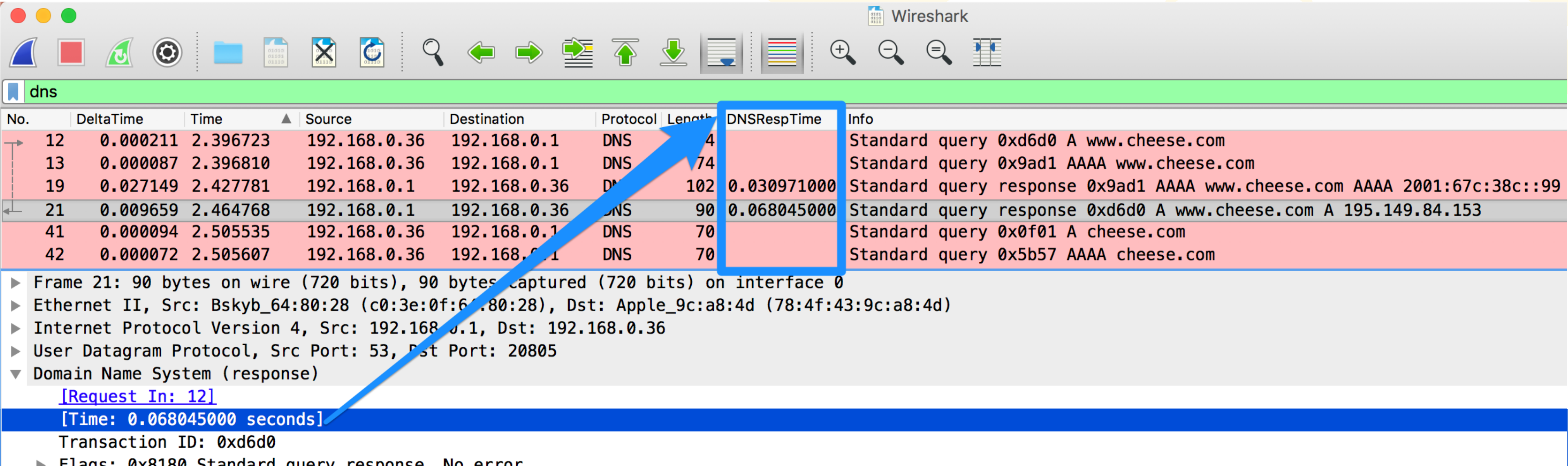
1. Capture Transaction

- Web App:
  - Disable/close unrelated services/apps
  - Clear browser cache
  - Shut down browser
  - Clear DNS cache (optional)
  - Start browser
  - In Wireshark, start capture with no/minimal filter
  - Launch dev tools
  - Load first web page
  - If you need to login/navigate to page of interest, do so
  - Set ping or marker right before transaction of interest
  - Wait for page to load
  - Stop capture

- Non-web app:
  - Log out of app and OS
  - Log into OS
  - Disable/close unrelated services/apps
  - In Wireshark, start capture with no/minimal filter
  - Launch app
  - If you need to login/navigate to begin transaction of interest, do so
  - Set ping or marker right before transaction of interest
  - Wait for transaction to complete
  - Stop capture

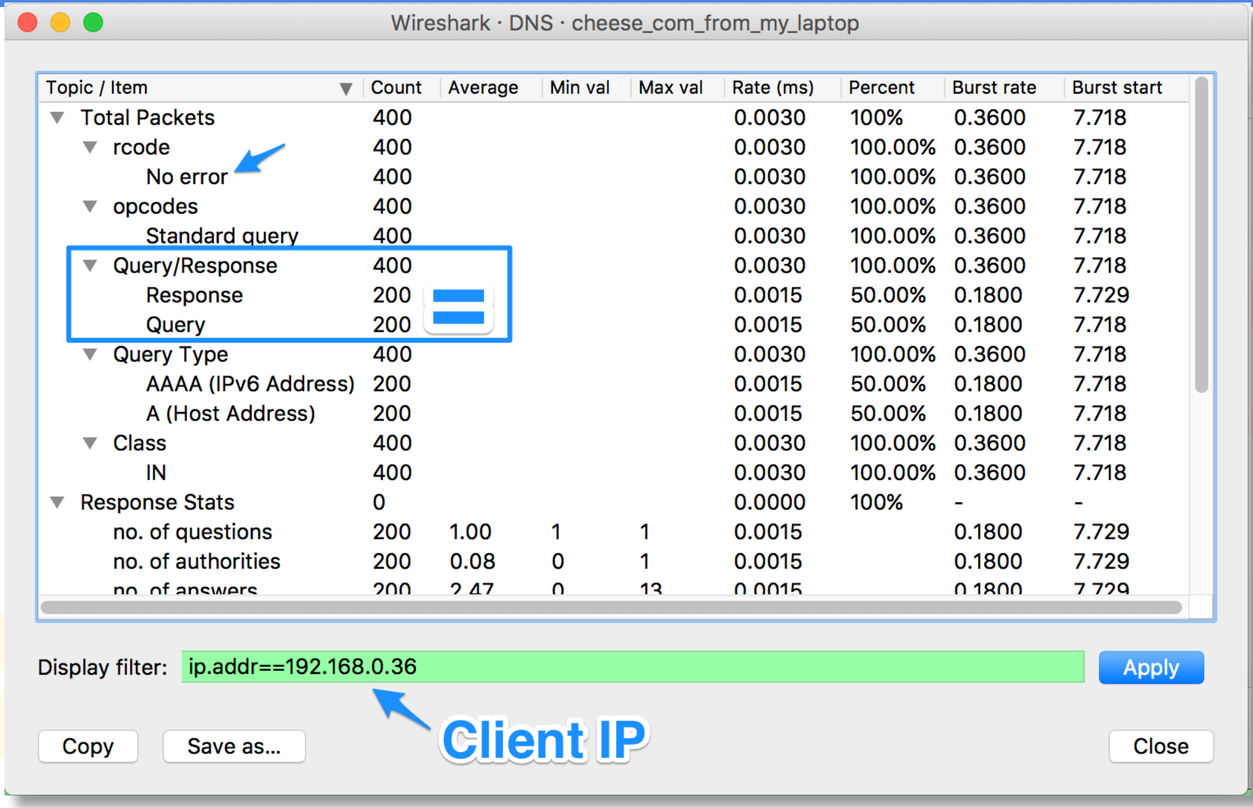
2. DNS Analysis

- Display filter to: **dns**
- Find first request for website/server of interest (ping/marker might help) and note the Epoch Time it was sent
- Find a DNS response > expand DNS layer > right-click [Time: xxx seconds] > Apply As Column
- Sort by this column to see if any DNS queries took a long time (>200ms)



2. DNS Analysis

- Statistics > DNS
- Look for Red Flags…
  - Error rcodes
  - More Queries than Responses
  - Large overall number of queries
- Consider filtering in time and/or by MAC address
- Note: If Wireshark is doing DNS name resolution during the capture, you'll see Query Type: PTR



2. DNS Analysis

- If you have more DNS queries than responses, it may be a timeout
- Each timeout = 1 second delay
- Use this display filter to find requests with no response: `transum.status == "Response missing" && dns`

transum.status == "Response missing" && dns | Expression… | + | SYNs

No.	Time	Source	Destination	Info	APDU Rsp Time	DNSRespTime	Name
1813	6.938919	fd10:ed79:1…	fd10:ed79:1807:…	Standard query 0x5c3f PTR 8.0.0.2.0.0.0.0.0.0.0.0.0.0.0.0.0.3.0.8.…			8.0.0.2.0.0.0.0.0.0.0.0.…
4066	12.000386	Lornas-MBP	SkyRouter.Home	Standard query 0x97ab AAAA sr.symcd.com			sr.symcd.com
5815	20.098154	Lornas-MBP	SkyRouter.Home	Standard query 0x6780 AAAA cks.mynativeplatform.com			cks.mynativeplatform.com
6103	20.240905	Lornas-MBP	SkyRouter.Home	Standard query 0x4b70 AAAA cmi.netseer.com			cmi.netseer.com
6991	21.127877	Lornas-MBP	SkyRouter.Home	Standard query 0x6780 AAAA cookiesync-mynativeplatform-347915877…			cookiesync-mynativeplatf…
6997	21.255238	Lornas-MBP	SkyRouter.Home	Standard query 0x4b70 AAAA cm.netseer.com			cm.netseer.com
104…	43.507570	Lornas-MBP	SkyRouter.Home	Standard query 0x38b3 AAAA cs.ns1p.net			cs.ns1p.net
10…	44.537710	Lornas-MBP	SkyRouter.Home	Standard query 0x38b3 AAAA lb.ns1p.net			lb.ns1p.net

2. DNS Analysis

- Choose a server to investigate, set display filter to: **`dns.qry.name == “server name” || dns.cname == “server name”`**

dns.qry.name== "lb.ns1p.net" || dns.cname == "lb.ns1p.net"

No.	Time	Source	Destination	Info
104…	43.536737	SkyRouter.H…	Lornas-MBP	Standard query response 0xbb47 A cs.ns1p.net CNAME lb.ns1p.net A 178.62.45.99
104…	43.537096	Lornas-MBP	SkyRouter.Home	Standard query 0x38b3 AAAA lb.ns1p.net
105…	44.537710	Lornas-MBP	SkyRouter.Home	Standard query 0x38b3 AAAA lb.ns1p.net

- Add the IP Address to the filter: `dns.qry.name == “server name” || dns.cname == “server name”` **`|| ip.addr == <IP address>`**

dns.qry.name == "lb.ns1p.net" || dns.cname == "lb.ns1p.net" || ip.addr == 178.62.45.99

No.	Time ▲	Source	Destination	Info
104…	43.536737	SkyRouter.H…	Lornas-MBP	Standard query response 0xbb47 A cs.ns1p.net CNAME lb.ns1p.net A 178.62.45.99
104…	43.537096	Lornas-MBP	SkyRouter.Home	Standard query 0x38b3 AAAA lb.ns1p.net
105…	44.537710	Lornas-MBP	SkyRouter.Home	Standard query 0x38b3 AAAA lb.ns1p.net
106…	45.603846	Lornas-MBP	lb.ns1p.net	53828 → 443 [SYN] Seq=0 Win=65535 Len=0 MSS=1460 WS=32 TSval=718539599 TSecr=…
106…	45.603981	Lornas-MBP	lb.ns1p.net	53829 → 443 [SYN] Seq=0 Win=65535 Len=0 MSS=1460 WS=32 TSval=718539599 TSecr=…
106…	45.623579	lb.ns1p.net	Lornas-MBP	443 → 53828 [SYN, ACK] Seq=0 Ack=1 Win=28960 Len=0 MSS=1460 SACK_PERM=1 TSval…
106…	45.623646	Lornas-MBP	lb.ns1p.net	53828 → 443 [ACK] Seq=1 Ack=1 Win=131744 Len=0 TSval=718539618 TSecr=37925430…

- We wasted 2 seconds from initial DNS query to SYN waiting for an IPv6 address!!

3. Remove Irrelevant Traffic

1. Save unfiltered capture in new directory
2. Set display filter to a subset of traffic
3. Optional: quickly "not" it to see what you're throwing away
4. File > Export Specified Packets > "filter_1" > Save
5. File > Open… "filter_1"

- Examples:
  - Filter to `eth.addr == <client MAC address>`
  - Remove UDP/ARP/etc.
  - Remove SMB, TCP connections for other apps

4. Examine/Remove Irrelevant Connections

- Remove connections to other services on client
- Remove irrelevant TCP connections that began before your transaction began

4a. Filter out ongoing TCP connections

- Ongoing: connections that began before your capture began
- Check what you're deleting… To see the first captured packet from all ongoing connections, use this filter:

```
tcp.seq == 1 && tcp.ack == 1 && tcp.window_size_scalefactor ==
-1 && tcp.time_delta == 0
```

- To remove all packets from all ongoing TCP Connections, use this filter:

```
!tcp.window_size_scalefactor == -1
```

tcp.window_size_scalefactor == -1 && tcp.time_delta == 0

No.	DeltaTimeStream	Time	Source	Destination	Protocol	Stream index ▲	Info
11…	0.000000000	1.758055	2a02:c7d:5bfa:…	lhr25s13-in-x03.1e100…	TCP	21	57652 → 443 [ACK]
93…	0.000000000	12.887295	2a02:c7d:5bfa:…	lhr25s13-in-x03.1e100…	TCP	116	57653 → 443 [ACK]
93…	0.000000000	14.190690	162.125.18.133	Lornas-MBP	TLSv…	117	Application Data
10…	0.000000000	39.162551	2a02:c7d:5bfa:…	lhr25s13-in-x0d.1e100…	TCP	145	57644 → 443 [ACK]
10…	0.000000000	39.162552	2a02:c7d:5bfa:…	safebrowsing.googleap…	TCP	146	57643 → 443 [ACK]
10…	0.000000000	39.162552	2a02:c7d:5bfa:…	lhr25s13-in-x03.1e100…	TCP	147	57642 → 443 [ACK]
11…	0.000000000	70.757708	162.125.18.133	Lornas-MBP	TLSv…	154	Application Data

4b. Filter all TCP Connections that began before your transaction started

- Find ping/marker of transaction start, and copy its epoch time
- Set display filter to show handshake of all TCP connections that began after Wireshark started capturing but before your transaction began:
  `frame.time_epoch < *time* && tcp.flags.syn==1`
- Completely unrelated connections? Remove with: `!tcp.stream eq 0`
- Connections related to transaction? Filter out all packets from end of handshake to *time*: `!(tcp.stream eq 0 && tcp.flags.syn == 0 frame.time_epoch < *time*)`

6. Delete tail ends of connections

- Delete all the packets after the transaction end time/marker
- Note: these could still be useful for other analysis (bandwidth used by transaction, overall stats like packets and bytes sent)
- For performance analysis, the tail ends of TCP connections are not needed and can cause false-positive “long delays”

7. Analyse Conversations and Connections

- Statistics > Conversations
- How many servers did the client connect to?

Wireshark · Conversations · no_encrypted_alerts

Ethernet · 1 | IPv4 · 68 | IPv6 · 19 | TCP · 193 | UDP

Address A	Address B	Packets	Bytes	Packets A → B	Bytes A → B	Packets B → A	Bytes B → A
8.41.222.241	Lornas-MBP	36	13 k	13	6617	23	6445
ec2-18-194-27-248.eu-central-1.compute.amazonaws.com	Lornas-MBP	33	12 k	13	7253	20	5076
ec2-18-194-106-16.eu-central-1.compute.amazonaws.com	Lornas-MBP	37	11 k	15	8566	22	3335
a23-2-12-111.deploy.static.akamaitechnologies.com	Lornas-MBP	99	21 k	43	10 k	56	10 k
23.111.9.35	Lornas-MBP	158	110 k	84	104 k	74	5744
a23-212-232-51.deploy.static.akamaitechnologies.com	Lornas-MBP	236	175 k	125	153 k	111	21 k
ec2-34-204-227-165.compute-1.amazonaws.com	Lornas-MBP	45	10 k	23	8039	22	2847
ec2-34-205-22-5.compute-1.amazonaws.com	Lornas-MBP	31	10 k	12	6516	19	4325
ec2-34-205-136-198.compute-1.amazonaws.com	Lornas-MBP	33	14 k	13	8038	20	6279
ec2-34-206-248-96.compute-1.amazonaws.com	Lornas-MBP	20	9284	9	6470	11	2814
ec2-34-226-60-227.compute-1.amazonaws.com	Lornas-MBP	45	28 k	19	13 k	26	14 k
ec2-34-226-228-77.compute-1.amazonaws.com	Lornas-MBP	31	11 k	12	7025	19	4957

7. Analyse Conversations and Connections

- How many TCP Connections were established?
- Do these numbers make sense given what client was doing? (Browsers open 2-6 connections to each web server that has content.)
- Did any connections fail?
- RTT x 3 x # of SSL/TLS connections / # Concurrent Connections = estimate of the overhead of these handshakes on total response time

8. Look for SYN Retransmissions

- `tcp.flags.syn==1 && tcp.flags.ack==0 && tcp.analysis.retransmission`
- After sending a SYN, client waits 1 second for a SYN-ACK. If none is received, client retransmits the SYN.
- Thus, each lost/ignored SYN adds up to 1 second to the overall response time
- Add up the retransmitted SYNs related to your transaction to get the max impact in seconds
- Check each stream to see whether connection eventually succeeded

tcp.flags.syn==1 && tcp.flags.ack==0 && tcp.analysis.retransmission

No.	DeltaTimeStream	Time	Source	Destination	Protocol	Stream	Info
6028	1.040454000	8.467663	Lornas-MBP	log.dmtry.com	TCP	84	[TCP Retransmission] 57746 → 443 [SYN] Seq=0
9803	1.040381000	17.375862	Lornas-MBP	eu-lb-p01-544546018.e…	TCP	127	[TCP Retransmission] 57787 → 443 [SYN] Seq=0
9804	1.040804000	17.375863	Lornas-MBP	beacon-a-v2-596299490…	TCP	122	[TCP Retransmission] 57782 → 443 [SYN] Seq=0
9806	1.041368000	17.377048	Lornas-MBP	uip.semasio.net	TCP	129	[TCP Retransmission] 57789 → 443 [SYN] Seq=0
9903	1.046599000	17.632896	Lornas-MBP	beacon-a-v2-596299490…	TCP	134	[TCP Retransmission] 57794 → 443 [SYN] Seq=0
9960	1.029106000	18.406154	Lornas-MBP	uip.semasio.net	TCP	129	[TCP Retransmission] 57789 → 443 [SYN] Seq=0
100…	1.028381000	19.434535	Lornas-MBP	uip.semasio.net	TCP	129	[TCP Retransmission] 57789 → 443 [SYN] Seq=0
101…	1.018481000	20.453016	Lornas-MBP	uip.semasio.net	TCP	129	[TCP Retransmission] 57789 → 443 [SYN] Seq=0
101…	1.016747000	21.469763	Lornas-MBP	uip.semasio.net	TCP	129	[TCP Retransmission] 57789 → 443 [SYN] Seq=0
109…	1.013525000	55.667297	Lornas-MBP	geo-rtas.btrll.com	TCP	149	[TCP Retransmission] 57807 → 443 [SYN] Seq=0

8. Look for SYN Retransmissions

- Eventually, time between SYN retransmissions increases exponentially

tcp.stream eq 12 | Expression… | + | SY

No.	DeltaTimeStream	Time	Source	Destination	Stream index	Info
122	0.000000000	*REF*	Lornas-MBP	support.unblock-us.com	12	62439 → 443 [SYN] Seq=0 Win=65535 Len
130	1.001452000	1.001452	Lornas-MBP	support.unblock-us.com	12	[TCP Retransmission] 62439 → 443 [SYN
135	1.001045000	2.002497	Lornas-MBP	support.unblock-us.com	12	[TCP Retransmission] 62439 → 443 [SYN
145	1.001158000	3.003655	Lornas-MBP	support.unblock-us.com	12	[TCP Retransmission] 62439 → 443 [SYN
147	1.001351000	4.005006	Lornas-MBP	support.unblock-us.com	12	[TCP Retransmission] 62439 → 443 [SYN
151	1.000904000	5.005910	Lornas-MBP	support.unblock-us.com	12	[TCP Retransmission] 62439 → 443 [SYN
155	2.000341000	7.006251	Lornas-MBP	support.unblock-us.com	12	[TCP Retransmission] 62439 → 443 [SYN
157	4.001442000	11.007693	Lornas-MBP	support.unblock-us.com	12	[TCP Retransmission] 62439 → 443 [SYN
191	8.001730000	19.009423	Lornas-MBP	support.unblock-us.com	12	[TCP Retransmission] 62439 → 443 [SYN
231	16.0077020…	35.017125	Lornas-MBP	support.unblock-us.com	12	[TCP Retransmission] 62439 → 443 [SYN
382	32.0228860…	67.040011	Lornas-MBP	support.unblock-us.com	12	[TCP Retransmission] 62439 → 443 [SYN

Example: TCP Connection Issue

- Problem: Many Office 365 operations are slow for some users
- Methodology:
  - Identify a user who experienced symptoms
  - Goal was to capture entire session, including opening of TCP connections to O365
  - Laptops connected to O365 during Windows login, so we couldn't start capture until partway through
  - Disabled connection to O365 during login, then rebooted laptop, logged in to Windows, started packet capture, then opened Lync to initiate login
  - Stopped capture once presence data had loaded (list of colleagues and availability statuses)
  - Login was slow, so capture contained bad behaviour

Example: TCP Connection Issue

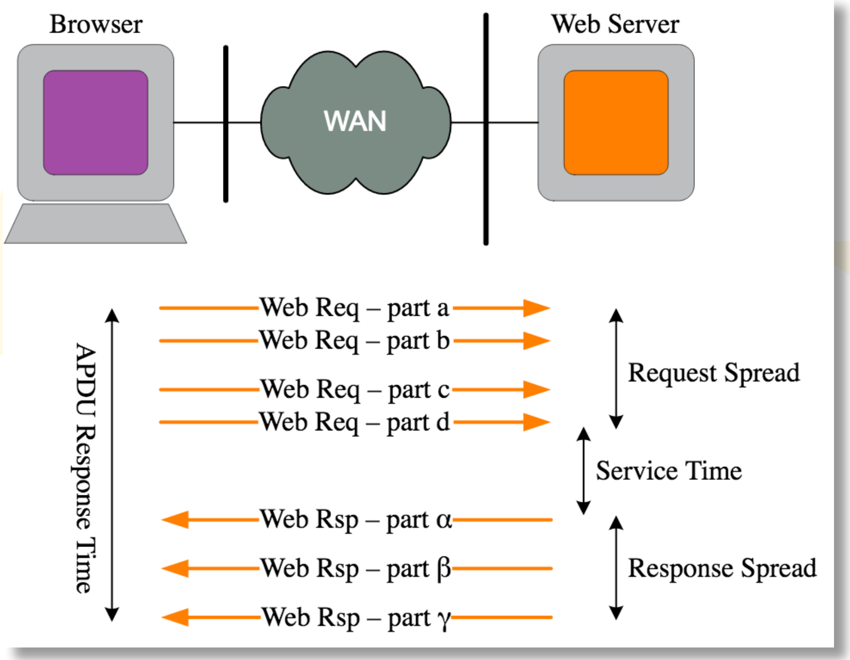
- What packets showed us:
  - Only 5% of TCP SYN messages received a response
  - Client cycled through 16 different server IPs, hoping for a connection
  - Response was usually a redirect to a server with a slightly different name (HTTP 301)
  - Once client found correct server name, login proceeded
  - DNS behavior helped us determine what name was being requested and which DNS servers were giving out these IP addresses.

Example: TCP Connection Issue

- Productivity Impact:
  - Laptop attempts to connect to servers in Microsoft Cloud
  - Three failed attempts take ~1.1 seconds
  - Last week, for requests from <location> alone, Microsoft servers refused over 700,000 connections
  - 77 HOURS in total added to log in times each week
- Solution:
  - Tech support case with Microsoft
  - Apply patch to impacted laptops

9. TRANSUM Response Time Analysis

- Large ADPU Rsp Time: slow overall
- Large Service Time: slow server
- Large Response Spread: slow network, retransmissions, or huge object
- Use with Web Developer Tools to identify name and type of object



9. TRANSUM Response Time Analysis

- If you have TLS traffic….
- TLS Encrypted Alert: Usually a normal notification that the session is stopping. Usually followed by a FIN.
  - These throw off TRANSUM APDU Response Time - remove them and save a separate trace file before doing TRANSUM analysis
  - Could also indicate a problem… but you can't tell, because they are encrypted!

8201	0.000004000	38.684453	s.ns1p.net	Lornas-MBP	TLSv1.2	Encrypted Alert
8202	0.000085000	38.684538	Lornas-MBP	s.ns1p.net	TCP	53830 → 443 [ACK] Seq=755 Ack=36…
8203	0.000001000	38.684539	Lornas-MBP	s.ns1p.net	TCP	53830 → 443 [ACK] Seq=755 Ack=36…
8204	0.000389000	38.684928	Lornas-MBP	s.ns1p.net	TCP	53830 → 443 [FIN, ACK] Seq=755 A…
8205	0.087488000	38.772416	s.ns1p.net	Lornas-MBP	TCP	443 → 53830 [ACK] Seq=3691 Ack=7…

9. TRANSUM Response Time Analysis

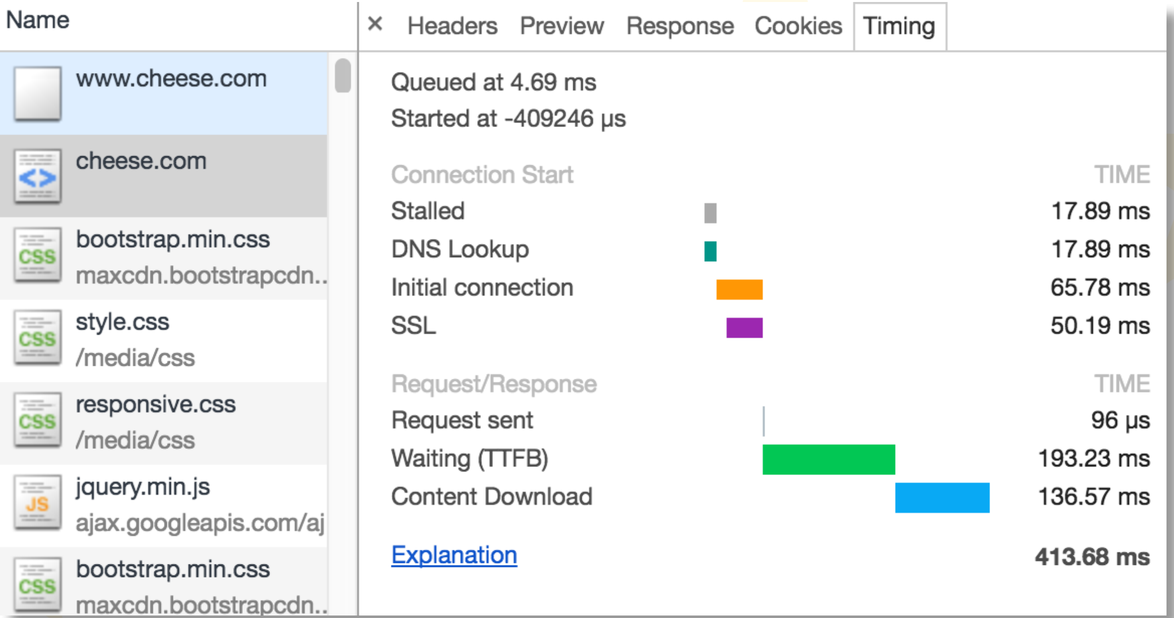
- Sort by APDU Rsp Time
- For each element, examine these columns:

No.	DeltaTimeStream	Time	Source	Destination	Info	APDU Rsp Time	Service Time	Rsp Spread
6546	4.439741000	18.982422	Lornas-MBP	fra-lb10.eu.adsymptotic.com	Application Data	15.090015000	15.090015000	0.000000000
2503	3.144814000	4.246014	Lorna-MBP	safebrowsing.googleapis.com	Application Data	3.326678000	0.102032000	0.001787000
5359	0.604582000	14.045147	Lornas-MBP	fra-lb10.eu.adsymptotic.com	Application Data	3.091182000	3.091182000	0.000000000
5346	2.752087000	14.036787	Lornas-MBP	prod.contextweb.map.fastly…	Application Data	2.869014000	0.040855000	0.000000000
7329	2.621923000	30.846127	Lornas-MBP	d2va07tmah0l23.oxcdn.com	Application Data	2.736500000	0.051695000	0.000004000
947	0.001237000	1.408361	Lornas-MBP	safebrowsing.googleapis.com	Client Hello	1.394638000	0.205031000	1.189607000
1802	0.001719000	2.907692	Lornas-MBP	update.googleapis.com	Application Data	1.222675000	0.040544000	0.016713000
6049	1.011437000	16.400048	Lorna-MBP	stats.l.doubleclick.net	Application Data	1.221680000	0.149851000	0.000109000

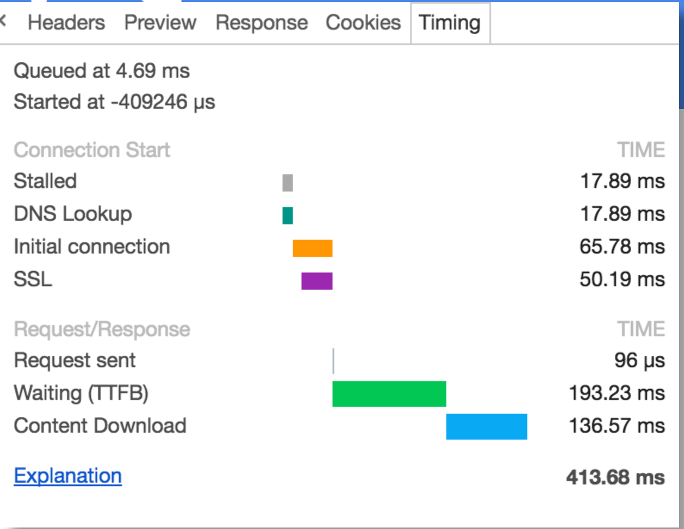
- Follow TCP stream for TCP analysis
- Use Developer Tools to identify object

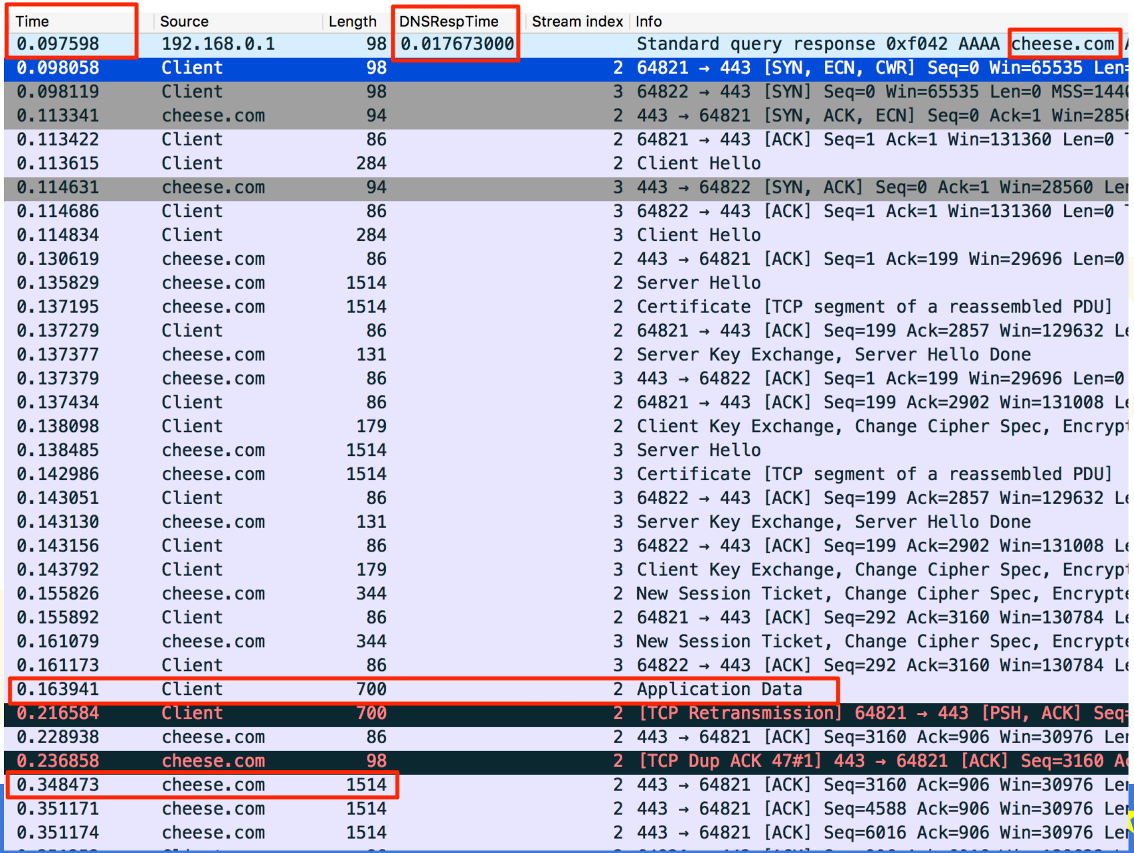
Investigating with Chrome Developer Tools

- Find object of interest (slow, large, etc.)
- Click Timing Tab
- What is main cause of delay?
- Find it in Wireshark!



Investigating with Chrome Developer Tools

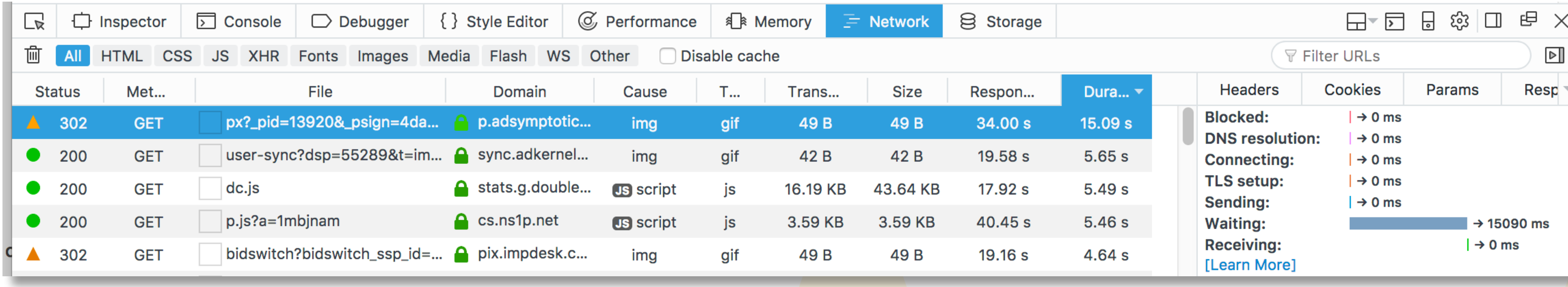


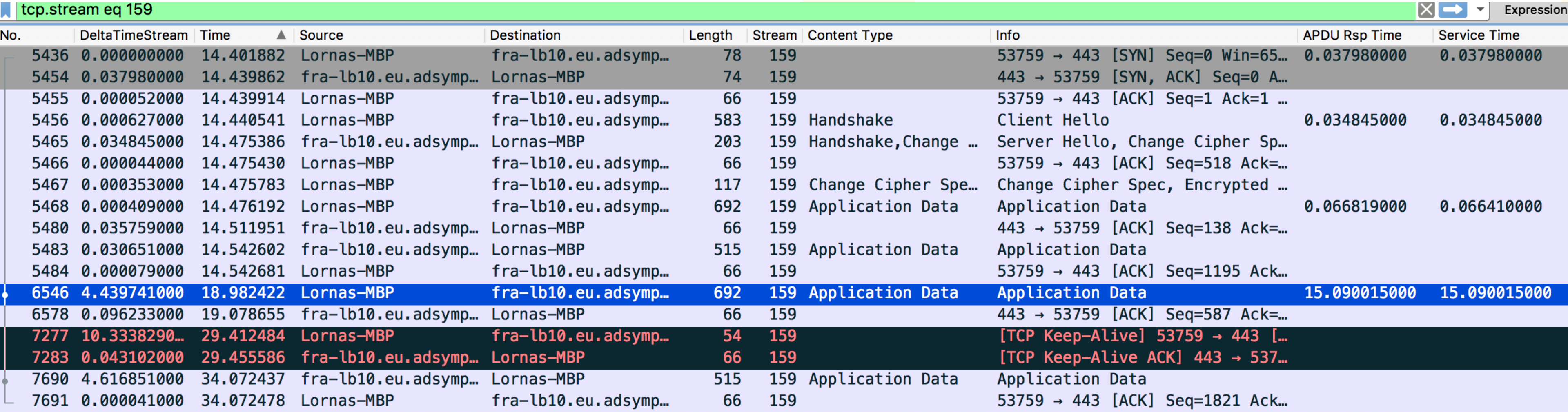


Time	Source	Length	DNSRespTime	Stream index	Info
0.097598	192.168.0.1	98	0.017673000		Standard query response 0xf042 AAAA cheese.com
0.098058	Client	98		2	64821 → 443 [SYN, ECN, CWR] Seq=0 Win=65535 Len
0.098119	Client	98		3	64822 → 443 [SYN] Seq=0 Win=65535 Len=0 MSS=144
0.113341	cheese.com	94		2	443 → 64821 [SYN, ACK, ECN] Seq=0 Ack=1 Win=285
0.113422	Client	86		2	64821 → 443 [ACK] Seq=1 Ack=1 Win=131360 Len=0
0.113615	Client	284		2	Client Hello
0.114631	cheese.com	94		3	443 → 64822 [SYN, ACK] Seq=0 Ack=1 Win=28560 Le
0.114686	Client	86		3	64822 → 443 [ACK] Seq=1 Ack=1 Win=131360 Len=0
0.114834	Client	284		3	Client Hello
0.130619	cheese.com	86		2	443 → 64821 [ACK] Seq=1 Ack=199 Win=29696 Len=0
0.135829	cheese.com	1514		2	Server Hello
0.137195	cheese.com	1514		2	Certificate [TCP segment of a reassembled PDU]
0.137279	Client	86		2	64821 → 443 [ACK] Seq=199 Ack=2857 Win=129632 L
0.137377	cheese.com	131		2	Server Key Exchange, Server Hello Done
0.137379	cheese.com	86		3	443 → 64822 [ACK] Seq=1 Ack=199 Win=29696 Len=0
0.137434	Client	86		2	64821 → 443 [ACK] Seq=199 Ack=2902 Win=131008 L
0.138098	Client	179		2	Client Key Exchange, Change Cipher Spec, Encryp
0.138485	cheese.com	1514		3	Server Hello
0.142986	cheese.com	1514		3	Certificate [TCP segment of a reassembled PDU]
0.143051	Client	86		3	64822 → 443 [ACK] Seq=199 Ack=2857 Win=129632 L
0.143130	cheese.com	131		3	Server Key Exchange, Server Hello Done
0.143156	Client	86		3	64822 → 443 [ACK] Seq=199 Ack=2902 Win=131008 L
0.143792	Client	179		3	Client Key Exchange, Change Cipher Spec, Encryp
0.155826	cheese.com	344		2	New Session Ticket, Change Cipher Spec, Encrypt
0.155892	Client	86		2	64821 → 443 [ACK] Seq=292 Ack=3160 Win=130784 L
0.161079	cheese.com	344		3	New Session Ticket, Change Cipher Spec, Encrypt
0.161173	Client	86		3	64822 → 443 [ACK] Seq=292 Ack=3160 Win=130784 L
0.163941	Client	700		2	Application Data
0.216584	Client	700		2	[TCP Retransmission] 64821 → 443 [PSH, ACK] Seq
0.228938	cheese.com	86		2	443 → 64821 [ACK] Seq=3160 Ack=906 Win=30976 Le
0.236858	cheese.com	98		2	[TCP Dup ACK 47#1] 443 → 64821 [ACK] Seq=3160 A
0.348473	cheese.com	1514		2	443 → 64821 [ACK] Seq=3160 Ack=906 Win=30976 Le
0.351171	cheese.com	1514		2	443 → 64821 [ACK] Seq=4588 Ack=906 Win=30976 Le
0.351174	cheese.com	1514		2	443 → 64821 [ACK] Seq=6016 Ack=906 Win=30976 Le

- Find DNS response for this server in Wireshark (at time 0.097s)
- Add 65ms to this time for Initial Connection
- Scroll down to time 0.163s and see Application Data
- TTFB was 185ms according to Wireshark

Example: Slow Server





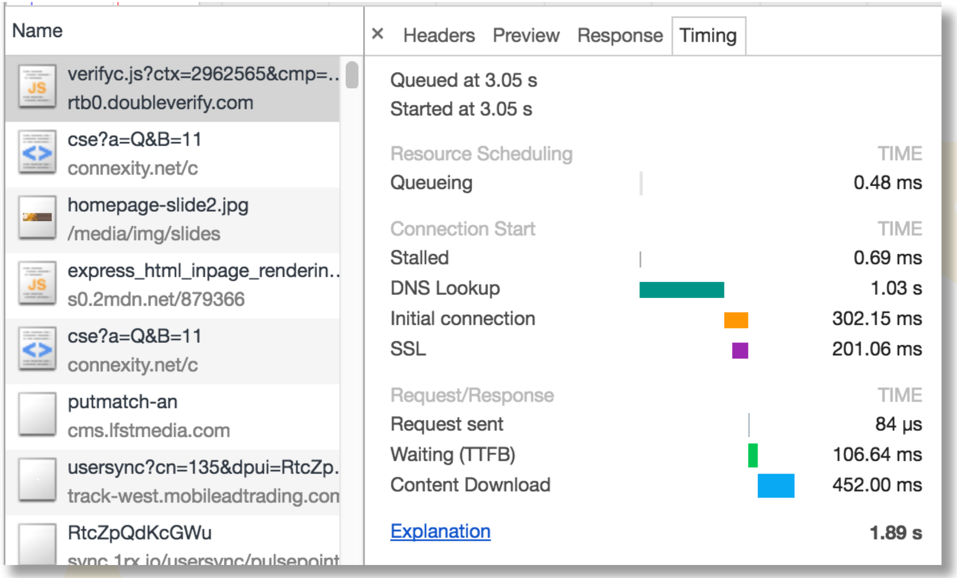
Example 2: Slow Server

tcp.stream eq 55 | Expression… | + | SYNs | Ongoing TCP | MyMactop | TCPD>.5

No.	Time	TCPDelta	Source	Destination	Length	Strear	APDUtime	Service Time	Rsp Spread	RTT	Info
15216	209.706179	5.139509000	Lorna	Server	1514	55					55168 → 443 [ACK] Seq=76829 Ack=536453 Win=131072…
15217	209.706180	0.000001000	Lorna	Server	1514	55					55168 → 443 [ACK] Seq=78277 Ack=536453 Win=131072…
15218	209.706180	0.000000000	Lorna	Server	834	55	21.3545520…	21.354551000	0.000000000		Application Data
15229	209.725793	0.019613000	Server	Lorna	66	55				0.019614000	443 → 55168 [ACK] Seq=536453 Ack=78277 Win=178080…
15230	209.725802	0.000009000	Server	Lorna	66	55				0.019622000	443 → 55168 [ACK] Seq=536453 Ack=79725 Win=178080…
15231	209.725803	0.000001000	Server	Lorna	66	55				0.019623000	443 → 55168 [ACK] Seq=536453 Ack=80493 Win=177440…
15684	220.127368	10.4015650…	Lorna	Server	54	55					[TCP Keep-Alive] 55168 → 443 [ACK] Seq=80492 Ack=…
15687	220.146083	0.018715000	Server	Lorna	66	55					[TCP Window Update] 443 → 55168 [ACK] Seq=536453 …
15719	230.158331	10.0122480…	Lorna	Server	54	55					[TCP Keep-Alive] 55168 → 443 [ACK] Seq=80492 Ack=…
15720	230.176922	0.018591000	Server	Lorna	66	55					[TCP Keep-Alive ACK] 443 → 55168 [ACK] Seq=536453…
15725	231.060731	0.883809000	Server	Lorna	1207	55					Application Data
15726	231.060821	0.000090000	Lorna	Server	66	55				0.000090000	55168 → 443 [ACK] Seq=80493 Ack=537594 Win=129920…
15727	231.070300	0.009479000	Lorna	Server	1514	55					55168 → 443 [ACK] Seq=80493 Ack=537594 Win=131072…
15728	231.070301	0.000001000	Lorna	Server	1514	55					55168 → 443 [ACK] Seq=81941 Ack=537594 Win=131072…
15729	231.070302	0.000001000	Lorna	Server	424	55	5.438696000	5.438687000	0.000007000		Application Data

Example: Long DNS Query

- Chrome says DNS query took over 1 second
- Wireshark shows a DNS request with no response. Request timed out after 1 second and was resent.
- We got an IPv4 response but didn't use it until we heard back about IPv6
- (Find DNS requests with no reply with transum.status == "Response missing" && dns)

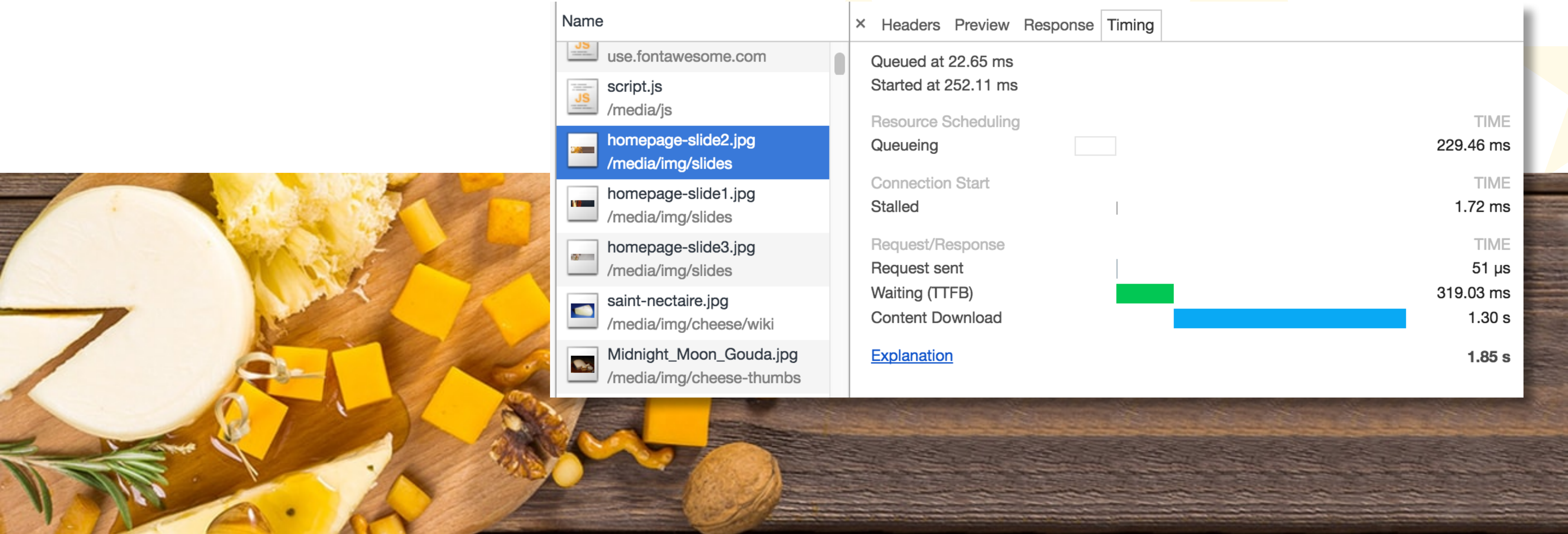


dns.qry.name == "rtb0.doubleverify.com" Expr

No.	DeltaTimeDisp	Time	Source	Destination	DNSRespTime	Info
3007	0.000000	3.544283	Lornas-MBP	SkyRouter.…		Standard query 0x55cd A rtb0.doubleverify.com
3008	0.000085	3.544368	Lornas-MBP	SkyRouter.…		Standard query 0xc41d AAAA rtb0.doubleverify.com
3085	0.051719	3.596087	SkyRouter.…	Lornas-MBP	0.051804000	Standard query response 0x55cd A rtb0.doubleverify.com CNAME bs-geo.dvgtm.ak…
3876	0.958465	4.554552	fd10:ed79:…	fd10:ed79:…		Standard query 0xb0a0 AAAA rtb0.doubleverify.com
3884	0.018465	4.573017	fd10:ed79:…	fd10:ed79:…	0.018465000	Standard query response 0xb0a0 AAAA rtb0.doubleverify.com

Example: Slow Image Load

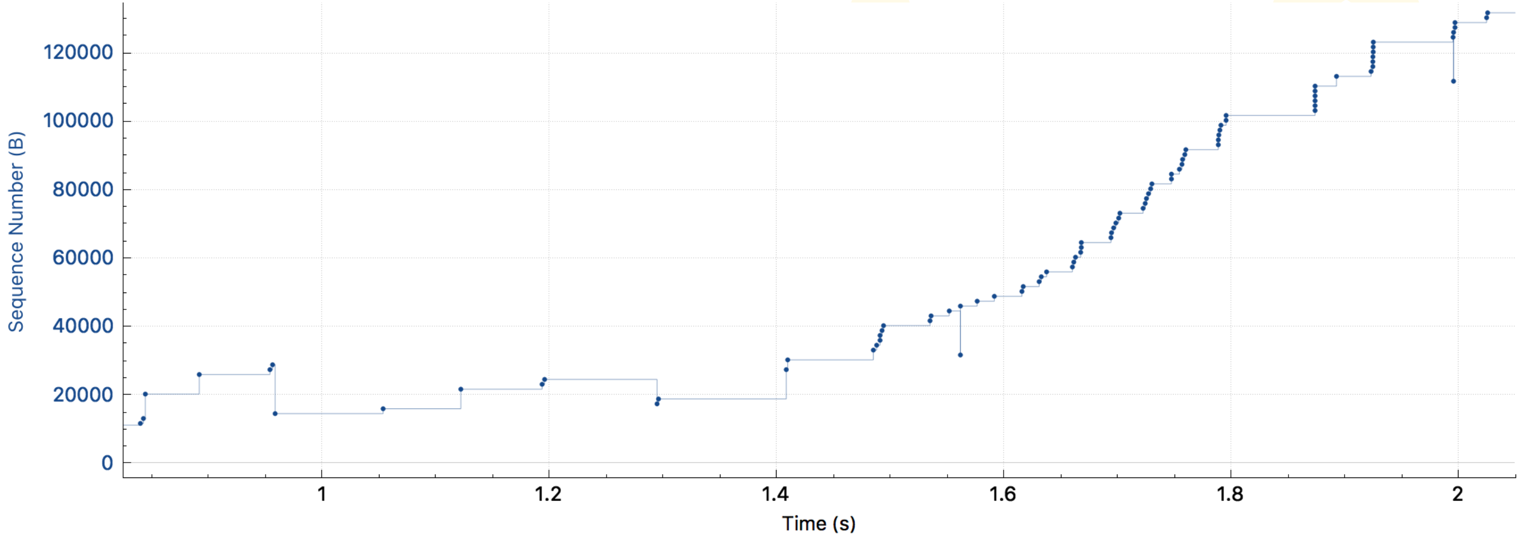
- 110 KB picture of cheese taking 1.85s to load
- Chrome indicates “Content Download” takes 1.3s



eltaTimeDisp	Time	Source	Destination	Length	Content Type	Info	APDU Rsp Time
0.022105	0.178047	cheese.com	Client	344	Handshake,Change…	New Session Ticket, Change Cipher Spec, Encry…	
0.000088	0.178135	Client	cheese.com	86		56994 → 443 [ACK] Seq=292 Ack=3160 Win=130784…	
0.321291	0.499426	Client	cheese.com	674	Application Data	Application Data	0.054128000
0.048832	0.548258	cheese.com	Client	1514		443 → 56994 [ACK] Seq=3160 Ack=880 Win=30848 …	
0.002764	0.551022	cheese.com	Client	1514		443 → 56994 [ACK] Seq=4588 Ack=880 Win=30848 …	
0.000047	0.551069	Client	cheese.com	86		56994 → 443 [ACK] Seq=880 Ack=6016 Win=129632…	
0.000220	0.551289	cheese.com	Client	1514		443 → 56994 [ACK] Seq=6016 Ack=880 Win=30848 …	
0.000034	0.551323	Client	cheese.com	86		56994 → 443 [ACK] Seq=880 Ack=7444 Win=131072…	
0.000923	0.552246	cheese.com	Client	1514		443 → 56994 [ACK] Seq=7444 Ack=880 Win=30848 …	
0.001308	0.553554	cheese.com	Client	1048	Application Data	Application Data	
0.000036	0.553590	Client	cheese.com	86		56994 → 443 [ACK] Seq=880 Ack=9834 Win=130080…	
0.211115	0.764705	Client	cheese.com	653	Application Data	Application Data	0.040519000
0.040164	0.804869	cheese.com	Client	1514		443 → 56994 [ACK] Seq=9834 Ack=1447 Win=32000…	
0.000355	0.805224	cheese.com	Client	588	Application Data	Application Data	
0.000034	0.805258	Client	cheese.com	86		56994 → 443 [ACK] Seq=1447 Ack=11764 Win=1305…	
0.002584	0.807842	Client	cheese.com	707	Application Data	Application Data	
0.124828	0.932670	Client	cheese.com	707		[TCP Retransmission] 56994 → 443 [PSH, ACK] S…	1.323781000
0.013219	0.945889	cheese.com	Client	1514		443 → 56994 [ACK] Seq=11764 Ack=2068 Win=3328…	
0.002637	0.948526	cheese.com	Client	1514		443 → 56994 [ACK] Seq=13192 Ack=2068 Win=3328…	
0.000064	0.948590	Client	cheese.com	86		56994 → 443 [ACK] Seq=2068 Ack=14620 Win=1296…	
0.001626	0.950216	cheese.com	Client	1514		[TCP Previous segment not captured] 443 → 569…	
0.000057	0.950273	Client	cheese.com	98		[TCP Window Update] 56994 → 443 [ACK] Seq=206…	
0.047421	0.997694	cheese.com	Client	1514		[TCP Previous segment not captured] 443 → 569…	
0.000039	0.997733	Client	cheese.com	106		[TCP Dup ACK 666#1] 56994 → 443 [ACK] Seq=206…	
0.062161	1.059894	cheese.com	Client	98		[TCP Dup ACK 664#1] 443 → 56994 [ACK] Seq=274…	
0.002319	1.062213	cheese.com	Client	1514		[TCP Previous segment not captured] , Ignored…	
0.000048	1.062261	Client	cheese.com	114		[TCP Dup ACK 666#2] 56994 → 443 [ACK] Seq=206…	
0.002211	1.064472	cheese.com	Client	1514		[TCP Fast Retransmission] 443 → 56994 [ACK] S…	
0.000050	1.064522	Client	cheese.com	114		56994 → 443 [ACK] Seq=2068 Ack=16048 Win=1296…	
0.094985	1.159507	cheese.com	Client	1514		[TCP Retransmission] 443 → 56994 [ACK] Seq=16…	
0.000037	1.159544	Client	cheese.com	114		56994 → 443 [ACK] Seq=2068 Ack=17476 Win=1296…	

Example: Slow Image Load

- Steven’s Graph shows it’s a mess due to loss of multiple packets
- Long TTFB was probably due to lost request from client
- Network issue!

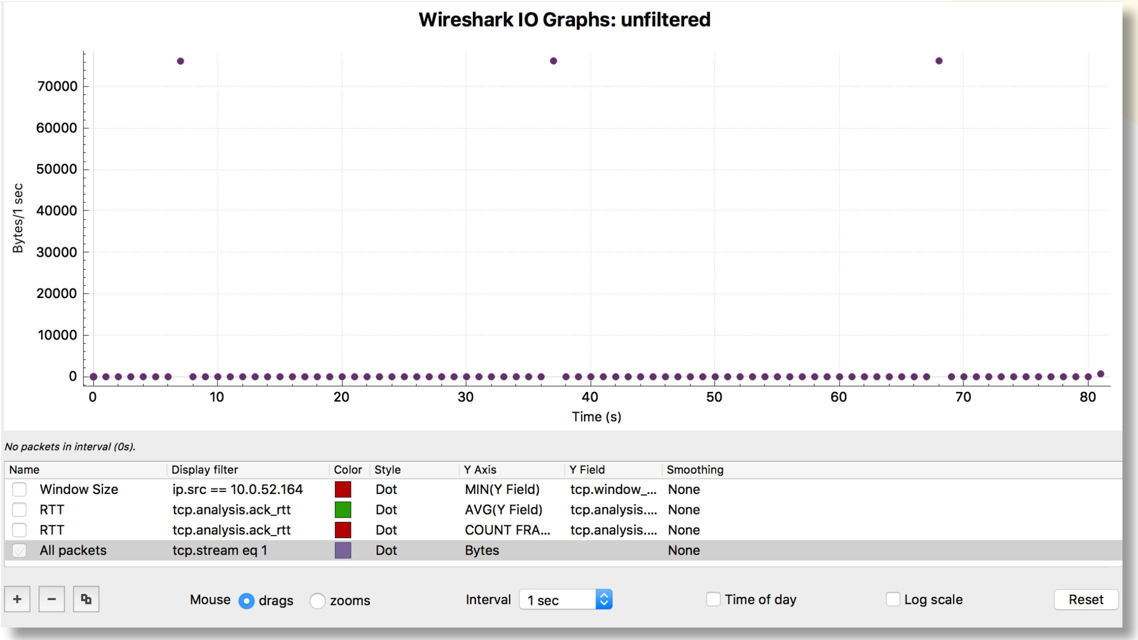


IO Graphs

- Get average/total stats of your choice for trace/client/server/connection
- Look for patterns
- Share data with others

IO Graph Example

- At what data rate is this connection sending data?

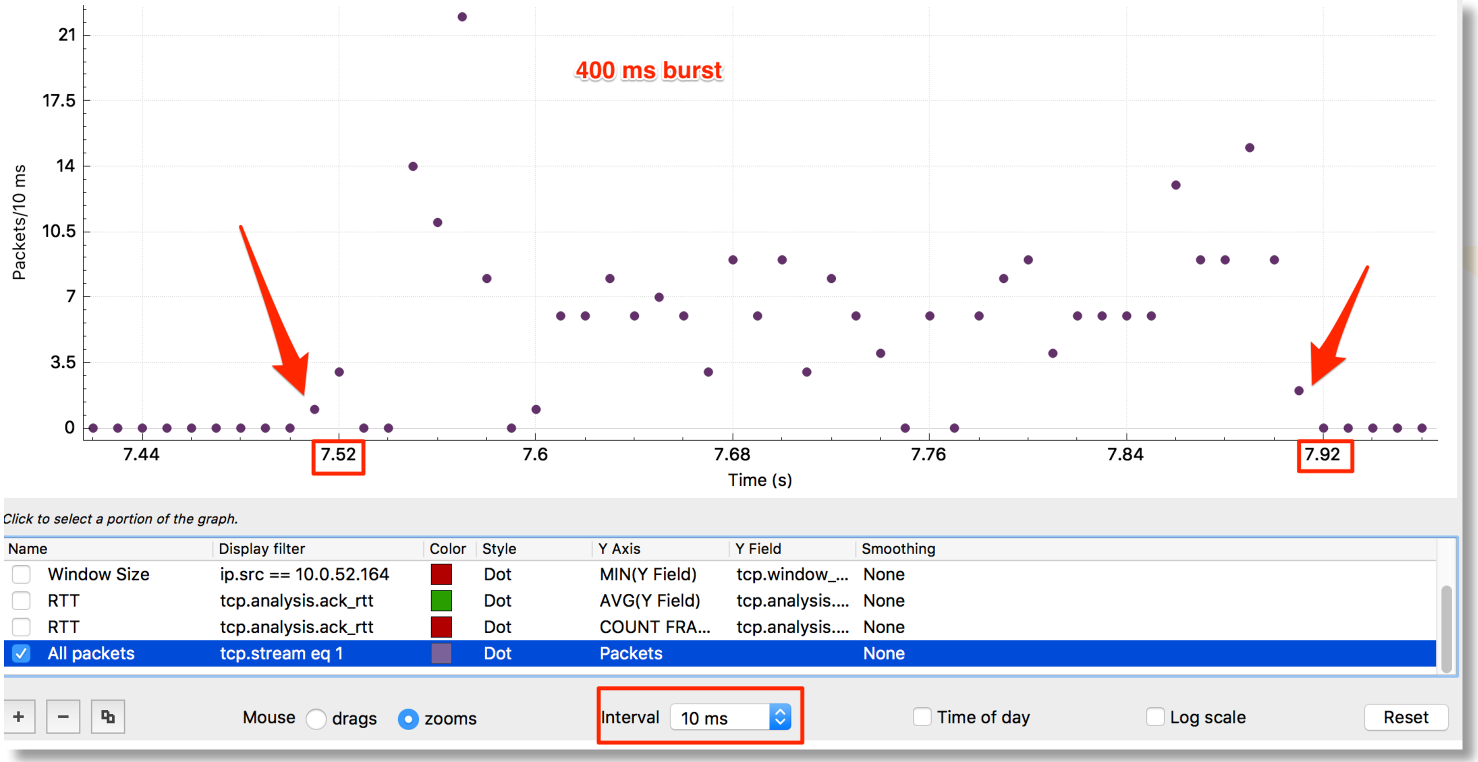


IO Graph Example

Zoom in on a burst and change interval to 10 ms

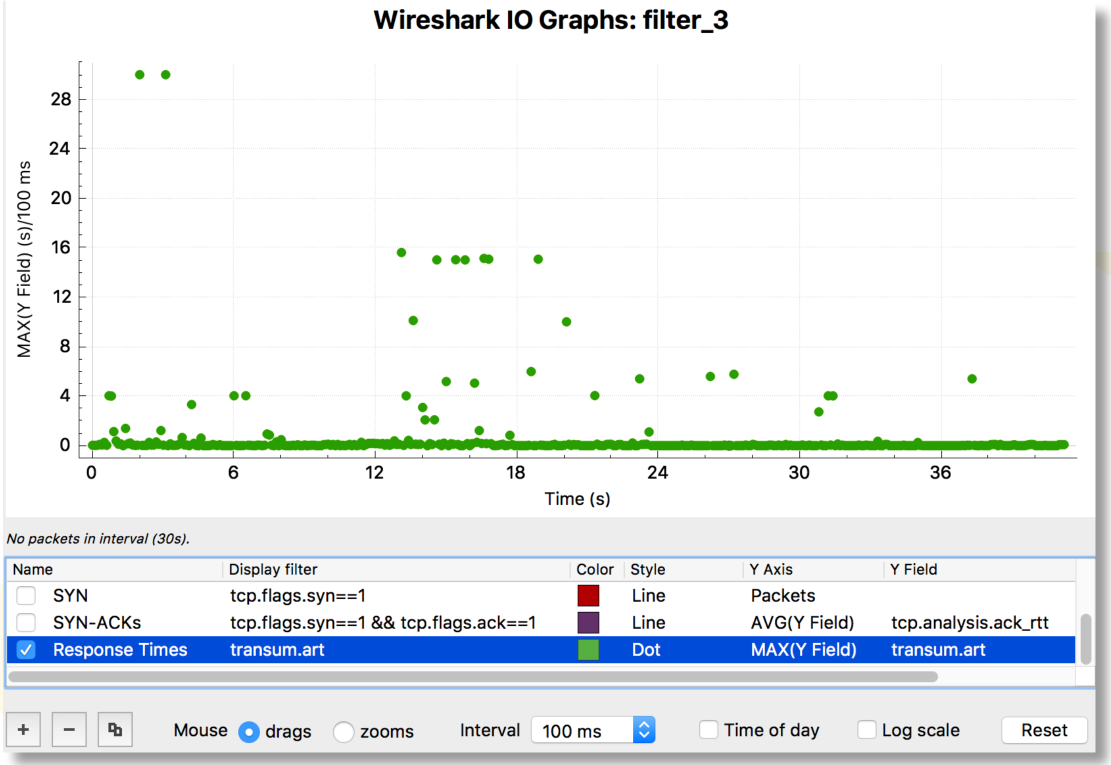
(78000 bytes * 8 bits/byte) / 400 ms

=1.56 Mbps

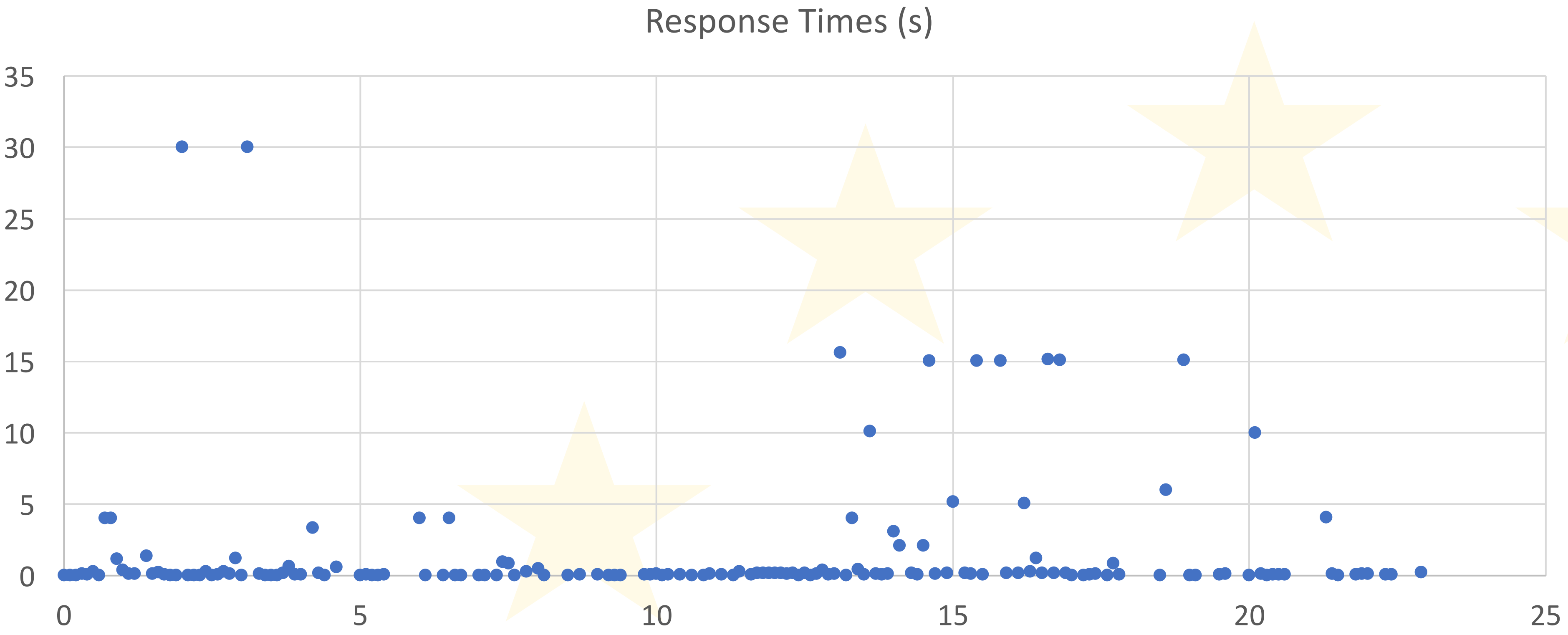


Suspicious Patterns

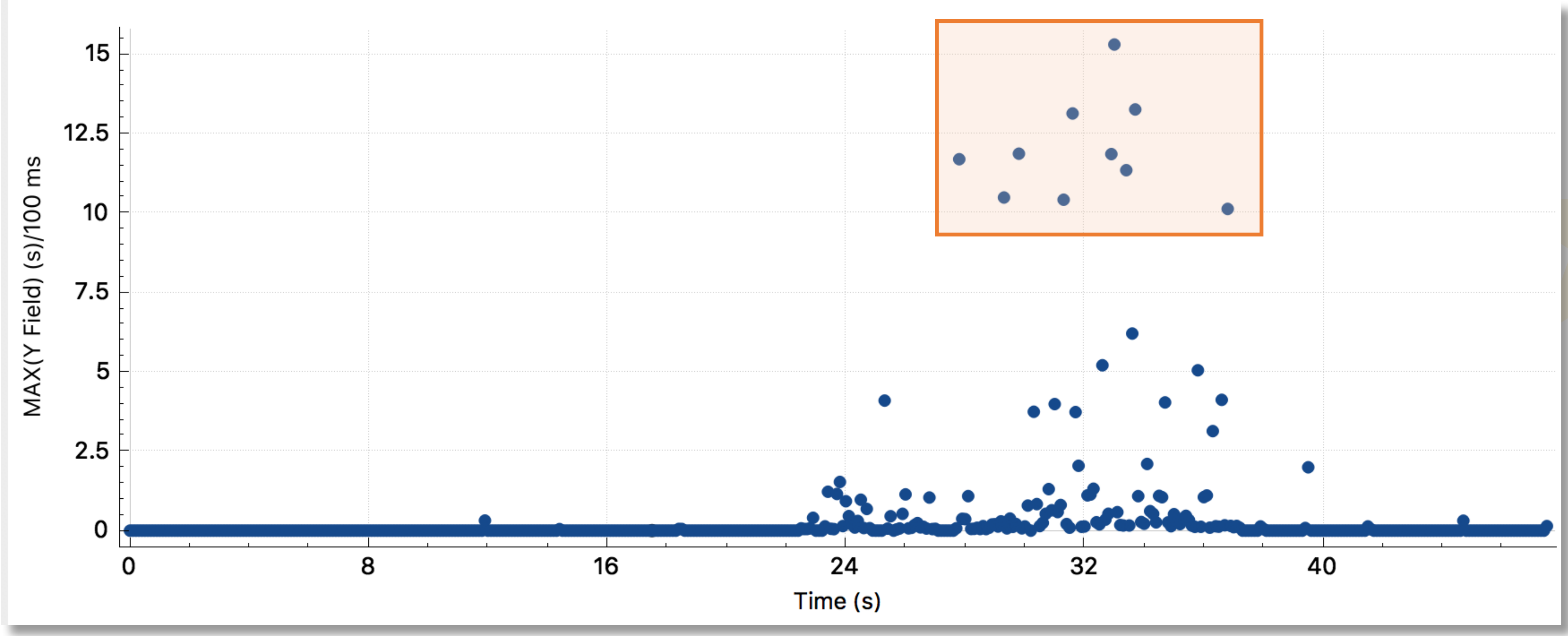
- Use Wireshark IO Graphs (Style = Dot)
- Example: Max ADPU Response Time for each 100ms interval
- Look for powers of 2 and multiples of 5
- Look for patterns



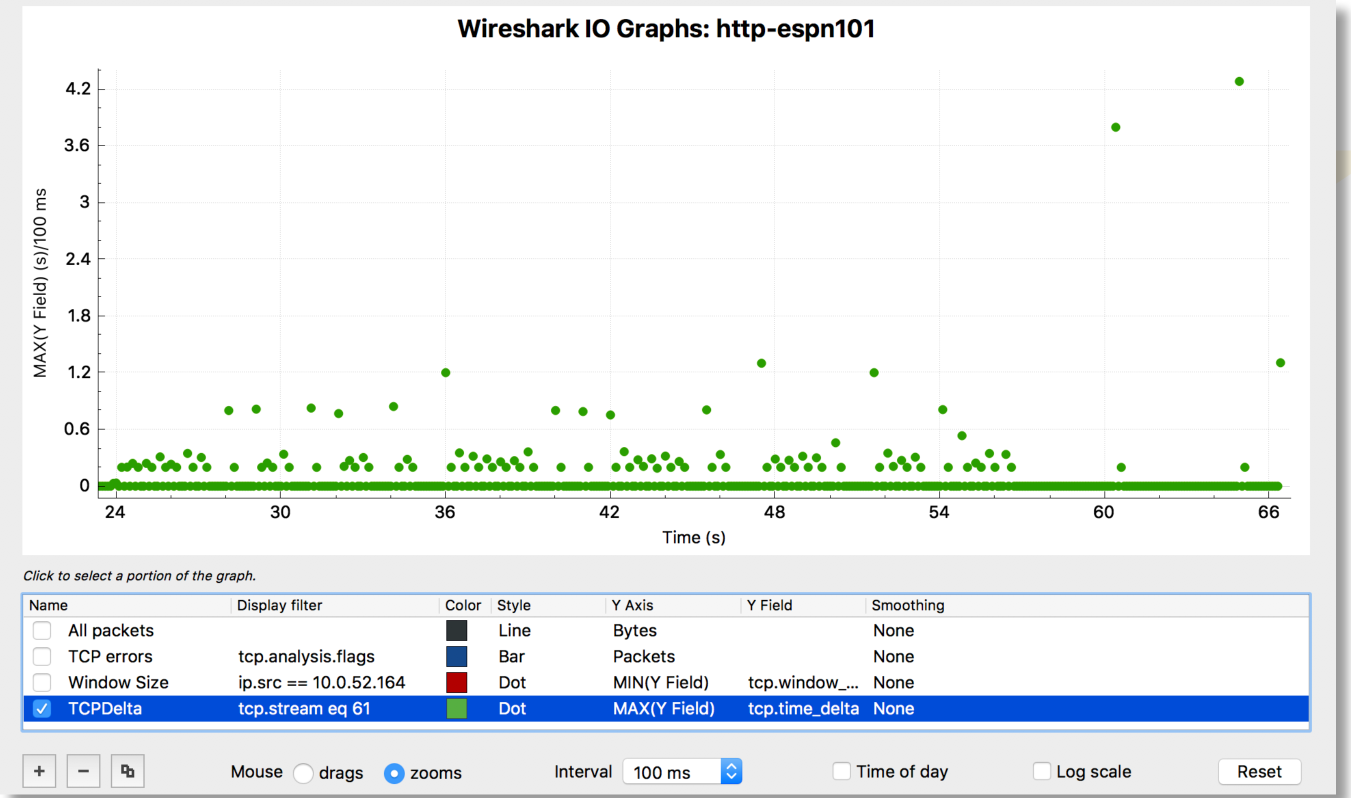
Suspicious Patterns



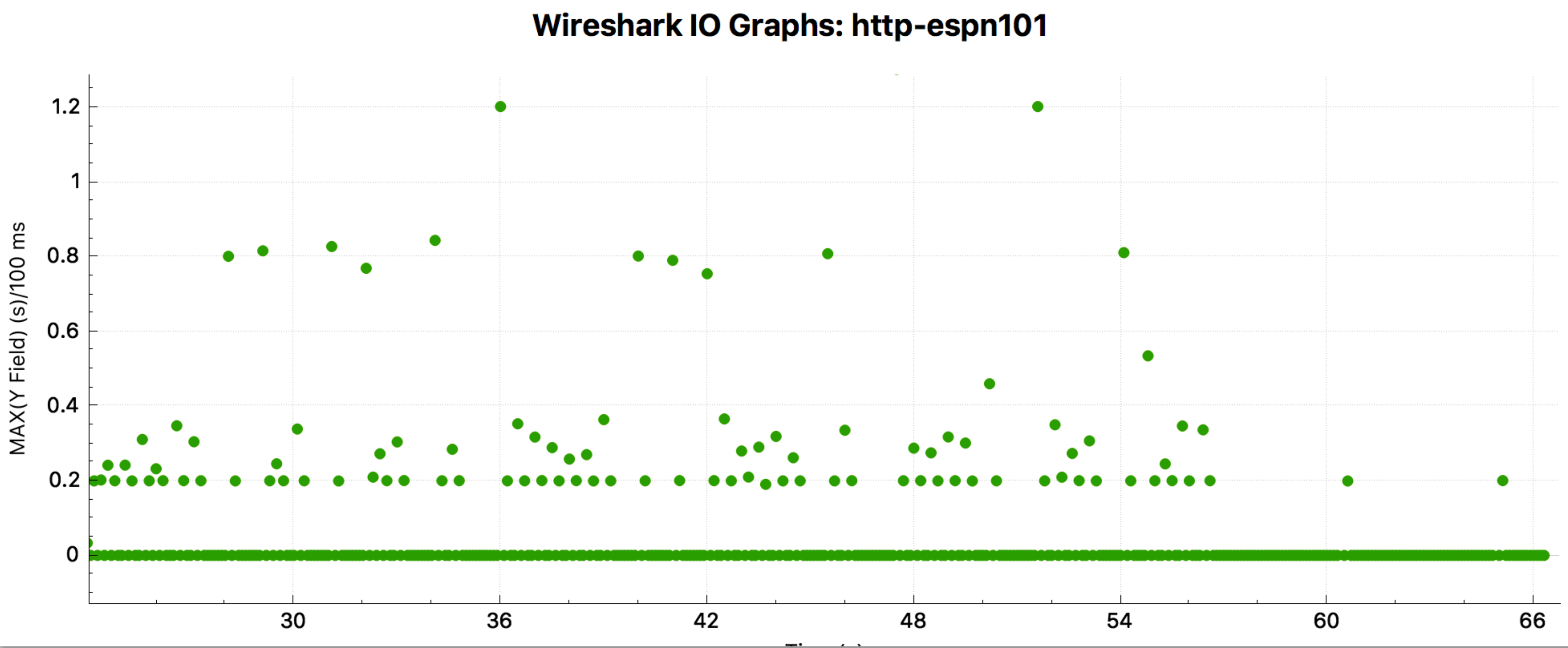
Suspicious Patterns



Suspicious Patterns



Suspicious Patterns (zoomed in)



Suspicious Patterns

- Is a pattern happening every X seconds?
- Is there a floor or ceiling on response times?
- Are many responses times just over X seconds?

- Timeout?
- Something else happening periodically (possibly that you can't see)?
- Do some googling, look in doc, capture closer to the thing responding
- Use other tools to look for similar patterns

Visualization/Reporting Techniques

eltaTimeDisp	Time	Source	Destination	Length	Content Type	Info	APDU Rsp Time
0.022105	0.178047	cheese.com	Client	344	Handshake,Change…	New Session Ticket, Change Cipher Spec, Encry…	
0.000088	0.178135	Client	cheese.com	86		56994 → 443 [ACK] Seq=292 Ack=3160 Win=130784…	
0.321291	0.499426	Client	cheese.com	674	Application Data	Application Data	0.054128000
0.048832	0.548258	cheese.com	Client	1514		443 → 56994 [ACK] Seq=3160 Ack=880 Win=30848 …	
0.002764	0.551022	cheese.com	Client	1514		443 → 56994 [ACK] Seq=4588 Ack=880 Win=30848 …	
0.000047	0.551069	Client	cheese.com	86		56994 → 443 [ACK] Seq=880 Ack=6016 Win=129632…	
0.000220	0.551289	cheese.com	Client	1514		443 → 56994 [ACK] Seq=6016 Ack=880 Win=30848 …	
0.000034	0.551323	Client	cheese.com	86		443 [ACK] Seq=880 Ack=7444 Win=131072…	
0.000923	0.552246	cheese.com	Client	1514		5994 [ACK] Seq=7444 Ack=880 Win=30848 …	
0.001308	0.553554	cheese.com	Client	1048		tion Data	
0.000036	0.553590	Client	cheese.com	86		443 [ACK] Seq=880 Ack=9834 Win=130080…	
0.211115	0.764705	Client	cheese.com	653		tion Data	0.040519000
0.040164	0.804869	cheese.com	Client	1514		5994 [ACK] Seq=9834 Ack=1447 Win=32000…	
0.000355	0.805224	cheese.com	Client	588		tion Data	
0.000034	0.805258	Client	cheese.com	86		443 [ACK] Seq=1447 Ack=11764 Win=1305…	
0.002584	0.807842	Client	cheese.com	707		tion Data	
0.124828	0.932670	Client	cheese.com	707		ransmission] 56994 → 443 [PSH, ACK] S…	1.323781000
0.013219	0.945889	cheese.com	Client	1514		5994 [ACK] Seq=11764 Ack=2068 Win=3328…	
0.002637	0.948526	cheese.com	Client	1514		5994 [ACK] Seq=13192 Ack=2068 Win=3328…	
0.000064	0.948590	Client	cheese.com	86		443 [ACK] Seq=2068 Ack=14620 Win=1296…	
0.001626	0.950216	cheese.com	Client	1514		evious segment not captured] 443 → 569…	
0.000057	0.950273	Client	cheese.com	98		ndow Update] 56994 → 443 [ACK] Seq=206…	
0.047421	0.997694	cheese.com	Client	1514		[TCP Previous segment not captured] 443 → 569…	
0.000039	0.997733	Client	cheese.com	106		[TCP Dup ACK 666#1] 56994 → 443 [ACK] Seq=206…	
0.062161	1.059894	cheese.com	Client	98		[TCP Dup ACK 664#1] 443 → 56994 [ACK] Seq=274…	
0.002319	1.062213	cheese.com	Client	1514		[TCP Previous segment not captured] , Ignored…	
0.000048	1.062261	Client	cheese.com	114		[TCP Dup ACK 666#2] 56994 → 443 [ACK] Seq=206…	
0.002211	1.064472	cheese.com	Client	1514		[TCP Fast Retransmission] 443 → 56994 [ACK] S…	
0.000050	1.064522	Client	cheese.com	114		56994 → 443 [ACK] Seq=2068 Ack=16048 Win=1296…	
0.094985	1.159507	cheese.com	Client	1514		[TCP Retransmission] 443 → 56994 [ACK] Seq=16…	
0.000037	1.159544	Client	cheese.com	114		56994 → 443 [ACK] Seq=2068 Ack=17476 Win=1296…	
0.068369	1.227913	cheese.com	Client	1514		[TCP Retransmission] 443 → 56994 [ACK] Seq=21…	
0.000278	1.228191	Client	cheese.com	114		[TCP Window Update] 56994 → 443 [ACK] Seq=206…	
0.071242	1.299433	cheese.com	Client	1514		[TCP Retransmission] 443 → 56994 [ACK] Seq=23…	

Visualization & Creating Reports

- Create tables to summarize statistics or compare transactions
- Give context

Worst Performing Microsoft Servers

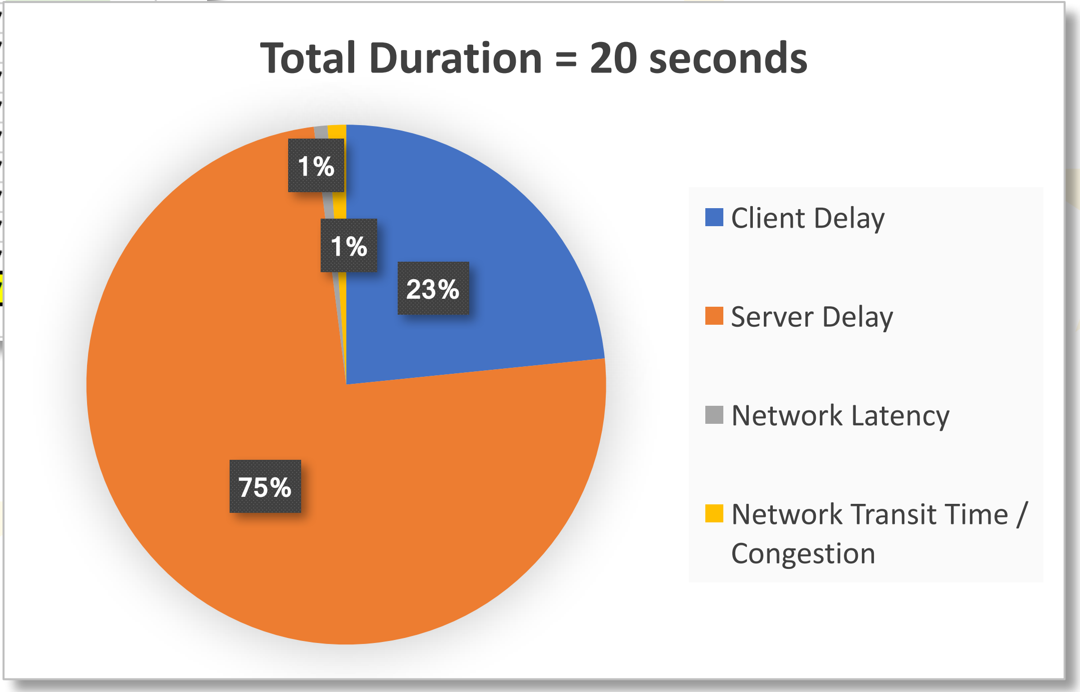
Information	Throughput (Inbound and Outbound) [kbits/sec]	Throughput (Inbound) [kbits/sec]	Throughput (Outbound) [kbits/sec]	Connections (TCP Servers) [#]	Connections Failed (TCP Servers) [#]	Server Response Time (Servers) [msec]
Microsoft Cloud	24264.253	6146.303	18117.950	443558.000	296845.000	144.672
IP addresses	1.801	0.945	0.857	1004.000	20129.000	73.846
	1.763	0.924	0.839	958.000	19957.000	49.632
	1.695	0.887	0.808	907.000	19346.000	51.373
	1.697	0.888	0.808	909.000	19345.000	55.135
	1.695	0.888	0.806	904.000	19314.000	51.411
	1.678	0.879	0.799	882.000	19280.000	29.832
	1.699	0.889	0.809	923.000	19249.000	76.384
	1.689	0.884	0.804	914.000	19156.000	32.223
	1.672	0.876	0.796	887.000	19115.000	28.843
	1.678	0.880	0.798	904.000	19085.000	29.703

Connection success rate: 5%

Home Page	
Number of Objects on Page	76
Number of Servers Contacted	18
Total Size of Content	1.39 MB
Number of Packets Sent	2077

Use Excel For Pie Charts and Graphs

J	K	L	M	N	O	P	Q	R	S	T	
leFactor	SEQ	Len	NXTSEQ	ACK	APDUtime	Service Time	Rsp Spread	Latency	iRTT	RTT	Info
	0	0		0	0.019752	0.019752	0	0.01677			5511
32	1	192	193	1	0.037561	0.018604	0.018957	0.01677			
32	193	126	319	6586	0.019188	0.019188	0	0.01677			
32	319	362	681	6828	0.155389	0.155282	0.000107	0.01677			
32	681	454	1135	17116	0.01982	0.01982	0	0.01677			
32	1135	449	1584	17839	0.042088	0.02155	0.020538	0.01677			
32	1584	409	1993	82217	0.023314	0.023049	0.000265	0.01677			
32	1993	411	2404	95810	0.023031	0.02195	0.001081	0.01677			
32	2404	458	2862	139396	0.06324	0.01677	0	0.01677			
32	2862	1358	4220	140619	14.888013	14.865046	0.022967	0.01677			
				SUMS	15.291396	15.181011	0.063915	0.1677			
					Min RTT	0.01677					



Common Problem #1: Large Client/Server Delay(s)

- Symptoms: Big gap(s) in application layer communication between client and server, but continued TCP communication (ACKs, Keepalives, etc.)
- Goal: Show decisively that delay was not on network, but on server side or client side
- Search for largest transum.art, tcp.time_delta, http.time, etc.
- Filter on that stream and sort by time
- Server delay:
  - Investigate to see what came after delay. Is this resource intensive? (aspx, requires backend db query…)
  - Validate with server side capture
- Client delay:
  - Look at resource bottlenecks on client (processes, CPU, mem, disk, network, number of open connections)
  - See if other comms to/from client may have caused slowdown

Common Problem #2: Huge/complex/chatty communications

- Examples:
  - Enormous web pages: count objects on page, connections, servers
  - Client establishing connections with large number of servers: count TCP connections, servers
  - Sending too much data given what application is trying to accomplish: Understand transaction. Look at # of bytes, packets, objects, connections, servers…
  - Thousands of application turns (chatty application) Set display filter to transum.art, count # of packets displayed at bottom = rough count of # of application turns. Also look for many small packets and packet patterns reminiscent of DB queries.
- Goal: illustrate how much data is being sent and that normal/typical conditions may result in slow response times
- Display list of content, connections, do simple math to show impact (RTT X application turns)

Common Problem #3: Timeouts and Failures

- DNS Failures/Errors
  - Look for: transum.status == "Response missing" && dns
  - Look for slow DNS.time
- Connecting to wrong server
- Failed connections
  - RST (may be fine), SYN retransmissions
- Server prematurely terminates connection
  - RST followed by SYN as client tries to reopen connection
- Something times out in backend but transaction still completes
  - Use I/O graphs
  - Look for long but consistent response times and delta times, suspicious numbers like 200ms, 1s, 2s, 4s, 5s, 8s, 10s, 15s, etc.

Thank you!

- Check out Wireshark Retrospective page for slides and more info/resources!