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

Hello IT
Have you tried turning it off and on again?
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
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

How to rule the world... by looking at packets!
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:

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Blocked</td>
<td>Time spent in a queue waiting for a network connection.</td>
</tr>
<tr>
<td></td>
<td>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 <code>network.http.max-persistent-connections-per-server</code> preference. If all connections are in use, the browser can't download more resources until a connection is released.</td>
</tr>
<tr>
<td>DNS resolution</td>
<td>Time taken to resolve a host name.</td>
</tr>
<tr>
<td>Connecting</td>
<td>Time taken to create a TCP connection.</td>
</tr>
<tr>
<td>Sending</td>
<td>Time taken to send the HTTP request to the server.</td>
</tr>
<tr>
<td>Waiting</td>
<td>Waiting for a response from the server.</td>
</tr>
<tr>
<td>Receiving</td>
<td>Time taken to read the entire response from the server (or cache).</td>
</tr>
</tbody>
</table>
• 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)

### Wireshark - Enabled Protocols

<table>
<thead>
<tr>
<th>Protocol</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>TNEF</td>
<td>Transport-Neutral Encapsulation Format</td>
</tr>
<tr>
<td>TNS</td>
<td>Transparent Network Substrate Protocol</td>
</tr>
<tr>
<td>Token-Ring</td>
<td>Token-Ring</td>
</tr>
<tr>
<td>TCP</td>
<td>Aiteon - Transparent Proxy Cache Protocol</td>
</tr>
<tr>
<td>TPKT</td>
<td>TPKT - ISO on TCP - RFC1006</td>
</tr>
<tr>
<td>TPNCP</td>
<td>AudioCodes TPNCP (TrunkPack Network Control Protocol)</td>
</tr>
<tr>
<td>TR-MAC</td>
<td>Token-Ring Media Access Control</td>
</tr>
<tr>
<td>TRANSMEM</td>
<td>TRANSMEM</td>
</tr>
<tr>
<td>TRILL</td>
<td>TRILL</td>
</tr>
<tr>
<td>TRILLVR</td>
<td>Microsoft Distributed Link Tracking Server Service</td>
</tr>
<tr>
<td>TSP</td>
<td>Time Synchronization Protocol</td>
</tr>
<tr>
<td>TTE</td>
<td>TTE</td>
</tr>
<tr>
<td>TTEET</td>
<td>TTE Ethernet</td>
</tr>
<tr>
<td>TTEPCF</td>
<td>TTE Ethernet Protocol Control Frame</td>
</tr>
<tr>
<td>TTP</td>
<td>Tiny Transport Protocol</td>
</tr>
<tr>
<td>TurboCell</td>
<td>TurboCell</td>
</tr>
<tr>
<td>TurboCellAggregateData</td>
<td>TurboCell Aggregate Data</td>
</tr>
<tr>
<td>TURNCHANNEL</td>
<td>TURN Channel</td>
</tr>
<tr>
<td>TURNchannel_stun</td>
<td>TURN Channel over STUN</td>
</tr>
<tr>
<td>TUXEDO</td>
<td>BEA Tuxedo</td>
</tr>
<tr>
<td>tuxedo_tcp</td>
<td>Tuxedo over TCP</td>
</tr>
</tbody>
</table>

Disabling a protocol prevents higher layer protocols from being displayed.
How to rule the world... by looking at packets!

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

TRANSUM Stats in decodes


Secure Sockets Layer

TRANSUM RTE Data

- [RTE Status: OK]
- [Req First Seg: 7200]
- [Req Last Seg: 7200]
- [Rsp First Seg: 7202]
- [Rsp Last Seg: 8582]
- [APDU Rsp Time: 4.032047000 seconds]
- [Service Time: 0.028464000 seconds]
- [Req Spread: 0.000000000 seconds]
- [Rsp Spread: 4.003583000 seconds]
- [Trace clip filter: tcp.stream==176 && frame.number>=7200 && frame.number<=8582 && tcp.len>0]
- [Calculation: Generic TCP]
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`
2. DNS Analysis

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

<table>
<thead>
<tr>
<th>No.</th>
<th>Time</th>
<th>Source</th>
<th>Destination</th>
<th>Info</th>
<th>Info</th>
</tr>
</thead>
<tbody>
<tr>
<td>104</td>
<td>43.536737</td>
<td>SkyRouter.H...</td>
<td>Lornas-MBP</td>
<td>Standard query response 0xb47 A cs.ns1p.net CNAME lb.ns1p.net A 178.62.45.99</td>
<td></td>
</tr>
<tr>
<td>104</td>
<td>43.537096</td>
<td>Lornas-MBP</td>
<td>SkyRouter.Home</td>
<td>Standard query 0x38b3 AAAA lb.ns1p.net</td>
<td></td>
</tr>
<tr>
<td>105</td>
<td>44.537710</td>
<td>Lornas-MBP</td>
<td>SkyRouter.Home</td>
<td>Standard query 0x38b3 AAAA lb.ns1p.net</td>
<td></td>
</tr>
</tbody>
</table>

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

<table>
<thead>
<tr>
<th>No.</th>
<th>Time</th>
<th>Source</th>
<th>Destination</th>
<th>Info</th>
<th>Info</th>
</tr>
</thead>
<tbody>
<tr>
<td>104</td>
<td>43.536737</td>
<td>SkyRouter.H...</td>
<td>Lornas-MBP</td>
<td>Standard query response 0xb47 A cs.ns1p.net CNAME lb.ns1p.net A 178.62.45.99</td>
<td></td>
</tr>
<tr>
<td>104</td>
<td>43.537096</td>
<td>Lornas-MBP</td>
<td>SkyRouter.Home</td>
<td>Standard query 0x38b3 AAAA lb.ns1p.net</td>
<td></td>
</tr>
<tr>
<td>105</td>
<td>44.537710</td>
<td>Lornas-MBP</td>
<td>SkyRouter.Home</td>
<td>Standard query 0x38b3 AAAA lb.ns1p.net</td>
<td></td>
</tr>
<tr>
<td>106</td>
<td>45.603846</td>
<td>lb.ns1p.net</td>
<td>lb.ns1p.net</td>
<td>53828 - 443 [SYN] Seq=0 Win=65535 Len=0 MSS=1460 WS=32 TSecr=...</td>
<td></td>
</tr>
<tr>
<td>106</td>
<td>45.603981</td>
<td>lb.ns1p.net</td>
<td>lb.ns1p.net</td>
<td>53829 - 443 [SYN] Seq=0 Win=65535 Len=0 MSS=1460 WS=32 TSecr=...</td>
<td></td>
</tr>
<tr>
<td>106</td>
<td>45.623579</td>
<td>lb.ns1p.net</td>
<td>Lornas-MBP</td>
<td>443 - 53828 [SYN, ACK] Seq=0 Ack=1 Win=28960 Len=0 MSS=1460 SACK_PERM=1 TSecr=...</td>
<td></td>
</tr>
<tr>
<td>106</td>
<td>45.623646</td>
<td>lb.ns1p.net</td>
<td>Lornas-MBP</td>
<td>53828 - 443 [ACK] Seq=1 Ack=1 Win=131744 Len=0 TSecr=718539618 TSecr=37925430</td>
<td></td>
</tr>
</tbody>
</table>

- 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:

\[ \text{tcp.seq} == 1 \land \text{tcp.ack} == 1 \land \text{tcp.window\_size\_scalefactor} == -1 \land \text{tcp.time\_delta} == 0 \]

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

\[ !\text{tcp.window\_size\_scalefactor} == -1 \]

<table>
<thead>
<tr>
<th>No.</th>
<th>DeltaTimeStream</th>
<th>Time</th>
<th>Source</th>
<th>Destination</th>
<th>Protocol</th>
<th>Stream index</th>
<th>Info</th>
</tr>
</thead>
<tbody>
<tr>
<td>11..</td>
<td>0.0000000000</td>
<td>1.758055</td>
<td>2a02:c7d:5bfa:...</td>
<td>lhr25s13-in-x03.1e100...</td>
<td>TCP</td>
<td>21</td>
<td>57652 → 443 [ACK]</td>
</tr>
<tr>
<td>93..</td>
<td>0.0000000000</td>
<td>12.887295</td>
<td>2a02:c7d:5bfa:...</td>
<td>lhr25s13-in-x03.1e100...</td>
<td>TCP</td>
<td>116</td>
<td>57653 → 443 [ACK]</td>
</tr>
<tr>
<td>93..</td>
<td>0.0000000000</td>
<td>14.190690</td>
<td>162.125.18.133</td>
<td>Lornas-MBP</td>
<td>TLSv...</td>
<td>117</td>
<td>Application Data</td>
</tr>
<tr>
<td>10..</td>
<td>0.0000000000</td>
<td>39.162551</td>
<td>2a02:c7d:5bfa:...</td>
<td>lhr25s13-in-x0d.1e100...</td>
<td>TCP</td>
<td>145</td>
<td>57644 → 443 [ACK]</td>
</tr>
<tr>
<td>10..</td>
<td>0.0000000000</td>
<td>39.162552</td>
<td>2a02:c7d:5bfa:...</td>
<td>safebrowsing.googleap...</td>
<td>TCP</td>
<td>146</td>
<td>57643 → 443 [ACK]</td>
</tr>
<tr>
<td>10..</td>
<td>0.0000000000</td>
<td>39.162552</td>
<td>2a02:c7d:5bfa:...</td>
<td>lhr25s13-in-x03.1e100...</td>
<td>TCP</td>
<td>147</td>
<td>57642 → 443 [ACK]</td>
</tr>
<tr>
<td>11..</td>
<td>0.0000000000</td>
<td>70.757708</td>
<td>162.125.18.133</td>
<td>Lornas-MBP</td>
<td>TLSv...</td>
<td>154</td>
<td>Application Data</td>
</tr>
</tbody>
</table>
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:
  \[ \text{frame.time_epoch} < \text{*time*} \land \land \text{tcp.flags.syn} == 1 \]
- Completely unrelated connections? Remove with: \( \text{!tcp.stream eq 0} \)
- Connections related to transaction? Filter out all packets from end of handshake to *time*:
  \( \text{!(tcp.stream eq 0 \land tcp.flags.syn == 0 \land frame.time_epoch < *time*)} \)
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?

<table>
<thead>
<tr>
<th>Address A</th>
<th>Address B</th>
<th>Packets</th>
<th>Bytes</th>
<th>Packets A → B</th>
<th>Bytes A → B</th>
<th>Packets B → A</th>
<th>Bytes B → A</th>
</tr>
</thead>
<tbody>
<tr>
<td>8.41.222.241</td>
<td>Lornas-MBP</td>
<td>36</td>
<td>13 k</td>
<td>13</td>
<td>6617</td>
<td>23</td>
<td>6445</td>
</tr>
<tr>
<td>ec2-18-194-106-16.eu-central-1.compute.amazonaws.com</td>
<td>Lornas-MBP</td>
<td>37</td>
<td>11 k</td>
<td>15</td>
<td>8566</td>
<td>22</td>
<td>3335</td>
</tr>
<tr>
<td>a23-2-12-111.deploy.static.akamaitechnologies.com</td>
<td>Lornas-MBP</td>
<td>99</td>
<td>21 k</td>
<td>43</td>
<td>10 k</td>
<td>56</td>
<td>10 k</td>
</tr>
<tr>
<td>23.111.9.35</td>
<td>Lornas-MBP</td>
<td>158</td>
<td>110 k</td>
<td>84</td>
<td>104 k</td>
<td>74</td>
<td>5744</td>
</tr>
<tr>
<td>a23-212-232-51.deploy.static.akamaitechnologies.com</td>
<td>Lornas-MBP</td>
<td>236</td>
<td>175 k</td>
<td>125</td>
<td>153 k</td>
<td>111</td>
<td>21 k</td>
</tr>
<tr>
<td>ec2-34-204-227-165.compute-1.amazonaws.com</td>
<td>Lornas-MBP</td>
<td>45</td>
<td>10 k</td>
<td>23</td>
<td>8039</td>
<td>22</td>
<td>2847</td>
</tr>
<tr>
<td>ec2-34-205-22-5.compute-1.amazonaws.com</td>
<td>Lornas-MBP</td>
<td>31</td>
<td>10 k</td>
<td>12</td>
<td>6516</td>
<td>19</td>
<td>4325</td>
</tr>
<tr>
<td>ec2-34-205-136-198.compute-1.amazonaws.com</td>
<td>Lornas-MBP</td>
<td>33</td>
<td>14 k</td>
<td>13</td>
<td>8038</td>
<td>20</td>
<td>6279</td>
</tr>
<tr>
<td>ec2-34-206-248-96.compute-1.amazonaws.com</td>
<td>Lornas-MBP</td>
<td>20</td>
<td>9284</td>
<td>9</td>
<td>6470</td>
<td>11</td>
<td>2814</td>
</tr>
<tr>
<td>ec2-34-226-60-227.compute-1.amazonaws.com</td>
<td>Lornas-MBP</td>
<td>45</td>
<td>28 k</td>
<td>19</td>
<td>7025</td>
<td>19</td>
<td>4957</td>
</tr>
<tr>
<td>ec2-34-226-228-77.compute-1.amazonaws.com</td>
<td>Lornas-MBP</td>
<td>31</td>
<td>11 k</td>
<td>12</td>
<td>6445</td>
<td>23</td>
<td>6445</td>
</tr>
</tbody>
</table>
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 \times 3 \times \# \text{ of SSL/TLS connections} / \# \text{ Concurrent Connections} = \text{estimate of the overhead of these handshakes on total response time}
8. Look for SYN Retransmissions

- \(\text{tcp.flags.syn}==1 \&\& \text{tcp.flags.ack}==0 \&\& \text{tcp.analysis.retransmission}\)

- 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.
8. Look for SYN Retransmissions

- Eventually, time between SYN retransmissions increases exponentially
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
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
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!

<table>
<thead>
<tr>
<th>Seq</th>
<th>Time</th>
<th>Source</th>
<th>Destination</th>
<th>Protocol</th>
<th>TLSv1.2</th>
<th>Encrypted Alert</th>
</tr>
</thead>
<tbody>
<tr>
<td>8201</td>
<td>0.000004000</td>
<td>38.684453</td>
<td>s.ns1p.net</td>
<td>TCP</td>
<td></td>
<td></td>
</tr>
<tr>
<td>8202</td>
<td>0.000085000</td>
<td>38.684538</td>
<td>Lornas-MBP</td>
<td>TCP</td>
<td>53830 → 443</td>
<td>[ACK] Seq=755 Ack=36...</td>
</tr>
<tr>
<td>8203</td>
<td>0.000001000</td>
<td>38.684539</td>
<td>Lornas-MBP</td>
<td>TCP</td>
<td>53830 → 443</td>
<td>[ACK] Seq=755 Ack=36...</td>
</tr>
<tr>
<td>8204</td>
<td>0.000389000</td>
<td>38.684928</td>
<td>Lornas-MBP</td>
<td>TCP</td>
<td>53830 → 443</td>
<td>[FIN, ACK] Seq=755 Ack=36...</td>
</tr>
<tr>
<td>8205</td>
<td>0.087488000</td>
<td>38.772416</td>
<td>s.ns1p.net</td>
<td>TCP</td>
<td>443 → 53830</td>
<td>[ACK] Seq=3691 Ack=7...</td>
</tr>
</tbody>
</table>
9. TRANSUM Response Time Analysis

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

- 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

#sf17eu • Estoril, Portugal
Example: Slow Server

![Network Monitor Screenshot]

- **Request**: GET
- **Domain**: adsymptotic
- **Cause**: img
- **Trans:** gif
- **Size**: 49 B, 49 B
- **Response Time**: 34.00 s, 15.09 s

Related Log Entry:
```
5436 8.00000000 14.401882 Lornas-MBP fra-lb10.eu.adsym... 78 159 53759 -> 443 [SYN] Seq=0 Win=65_ 0.037980000 0.037980000
```
Example 2: Slow Server
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`)
Example: Slow Image Load

- 110 KB picture of cheese taking 1.85s to load
- Chrome indicates “Content Download” takes 1.3s
<table>
<thead>
<tr>
<th>DataTimeDisp</th>
<th>Time</th>
<th>Source</th>
<th>Destination</th>
<th>Length</th>
<th>Content Type</th>
<th>Info</th>
<th>APDU Rsp Time</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.022105</td>
<td>0.178047</td>
<td>cheese.com</td>
<td>Client</td>
<td>344</td>
<td>Handshake,Change...</td>
<td>New Session Ticket, Change Cipher Spec, Encry...</td>
<td>0.054128000</td>
</tr>
<tr>
<td>0.000088</td>
<td>0.178135</td>
<td>cheese.com</td>
<td>Client</td>
<td>674</td>
<td>Application Data</td>
<td>Application Data</td>
<td></td>
</tr>
<tr>
<td>0.048832</td>
<td>0.548258</td>
<td>cheese.com</td>
<td>Client</td>
<td>1514</td>
<td>Application Data</td>
<td>Application Data</td>
<td>0.040519000</td>
</tr>
<tr>
<td>0.000047</td>
<td>0.551069</td>
<td>cheese.com</td>
<td>Client</td>
<td>86</td>
<td>Application Data</td>
<td>Application Data</td>
<td></td>
</tr>
<tr>
<td>0.000220</td>
<td>0.551289</td>
<td>cheese.com</td>
<td>Client</td>
<td>86</td>
<td>Application Data</td>
<td>Application Data</td>
<td></td>
</tr>
<tr>
<td>0.000034</td>
<td>0.551323</td>
<td>cheese.com</td>
<td>Client</td>
<td>86</td>
<td>Application Data</td>
<td>Application Data</td>
<td></td>
</tr>
<tr>
<td>0.013808</td>
<td>0.553554</td>
<td>cheese.com</td>
<td>Client</td>
<td>1048</td>
<td>Application Data</td>
<td>Application Data</td>
<td></td>
</tr>
<tr>
<td>0.000036</td>
<td>0.553590</td>
<td>cheese.com</td>
<td>Client</td>
<td>86</td>
<td>Application Data</td>
<td>Application Data</td>
<td></td>
</tr>
<tr>
<td>0.211115</td>
<td>0.764705</td>
<td>cheese.com</td>
<td>Client</td>
<td>653</td>
<td>Application Data</td>
<td>Application Data</td>
<td></td>
</tr>
<tr>
<td>0.040164</td>
<td>0.804869</td>
<td>cheese.com</td>
<td>Client</td>
<td>1514</td>
<td>Application Data</td>
<td>Application Data</td>
<td></td>
</tr>
<tr>
<td>0.000355</td>
<td>0.805224</td>
<td>cheese.com</td>
<td>Client</td>
<td>588</td>
<td>Application Data</td>
<td>Application Data</td>
<td></td>
</tr>
<tr>
<td>0.000034</td>
<td>0.805285</td>
<td>cheese.com</td>
<td>Client</td>
<td>86</td>
<td>Application Data</td>
<td>Application Data</td>
<td></td>
</tr>
<tr>
<td>0.124828</td>
<td>0.932670</td>
<td>cheese.com</td>
<td>Client</td>
<td>707</td>
<td>Application Data</td>
<td>Application Data</td>
<td>1.323781000</td>
</tr>
<tr>
<td>0.013219</td>
<td>0.945889</td>
<td>cheese.com</td>
<td>Client</td>
<td>1514</td>
<td>Application Data</td>
<td>Application Data</td>
<td></td>
</tr>
<tr>
<td>0.026371</td>
<td>0.948526</td>
<td>cheese.com</td>
<td>Client</td>
<td>1514</td>
<td>Application Data</td>
<td>Application Data</td>
<td></td>
</tr>
<tr>
<td>0.000064</td>
<td>0.948590</td>
<td>cheese.com</td>
<td>Client</td>
<td>86</td>
<td>Application Data</td>
<td>Application Data</td>
<td></td>
</tr>
<tr>
<td>0.001626</td>
<td>0.950216</td>
<td>cheese.com</td>
<td>Client</td>
<td>1514</td>
<td>Application Data</td>
<td>Application Data</td>
<td></td>
</tr>
<tr>
<td>0.000057</td>
<td>0.950273</td>
<td>cheese.com</td>
<td>Client</td>
<td>98</td>
<td>Application Data</td>
<td>Application Data</td>
<td></td>
</tr>
<tr>
<td>0.047421</td>
<td>0.997694</td>
<td>cheese.com</td>
<td>Client</td>
<td>1514</td>
<td>Application Data</td>
<td>Application Data</td>
<td></td>
</tr>
<tr>
<td>0.000039</td>
<td>0.997733</td>
<td>cheese.com</td>
<td>Client</td>
<td>106</td>
<td>Application Data</td>
<td>Application Data</td>
<td></td>
</tr>
<tr>
<td>0.062161</td>
<td>1.059894</td>
<td>cheese.com</td>
<td>Client</td>
<td>98</td>
<td>Application Data</td>
<td>Application Data</td>
<td></td>
</tr>
<tr>
<td>0.002319</td>
<td>1.062213</td>
<td>cheese.com</td>
<td>Client</td>
<td>1514</td>
<td>Application Data</td>
<td>Application Data</td>
<td></td>
</tr>
<tr>
<td>0.000048</td>
<td>1.062261</td>
<td>cheese.com</td>
<td>Client</td>
<td>114</td>
<td>Application Data</td>
<td>Application Data</td>
<td></td>
</tr>
<tr>
<td>0.022111</td>
<td>1.064472</td>
<td>cheese.com</td>
<td>Client</td>
<td>1514</td>
<td>Application Data</td>
<td>Application Data</td>
<td></td>
</tr>
<tr>
<td>0.000050</td>
<td>1.064522</td>
<td>cheese.com</td>
<td>Client</td>
<td>1514</td>
<td>Application Data</td>
<td>Application Data</td>
<td></td>
</tr>
<tr>
<td>0.094985</td>
<td>1.159507</td>
<td>cheese.com</td>
<td>Client</td>
<td>1514</td>
<td>Application Data</td>
<td>Application Data</td>
<td></td>
</tr>
<tr>
<td>0.000037</td>
<td>1.159544</td>
<td>cheese.com</td>
<td>Client</td>
<td>114</td>
<td>Application Data</td>
<td>Application Data</td>
<td></td>
</tr>
<tr>
<td>0.000050</td>
<td>1.201703</td>
<td>cheese.com</td>
<td>Client</td>
<td>114</td>
<td>Application Data</td>
<td>Application Data</td>
<td></td>
</tr>
<tr>
<td>0.013808</td>
<td>1.201703</td>
<td>cheese.com</td>
<td>Client</td>
<td>1048</td>
<td>Application Data</td>
<td>Application Data</td>
<td></td>
</tr>
</tbody>
</table>
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
• At what data rate is this connection sending data?
Estoril, Portugal

How to rule the world... by looking at packets!

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

Response Times (s)

#sf17eu • Estoril, Portugal

How to rule the world... by looking at packets!

51
Suspicious Patterns

MAX(Y Field) (s)/100 ms

Time (s)

How to rule the world... by looking at packets!
Suspicious Patterns

How to rule the world... by looking at packets!

Wireshark IO Graphs: http-espn101

Time (s)
0
24
30
36
42
48
54
60
66
MAX(X) Field (s)/100 ms
0
0.6
1.2
1.8
2.4
3.0
3.6
4.2
Click to select a portion of the graph.

<table>
<thead>
<tr>
<th>Name</th>
<th>Display filter</th>
<th>Color</th>
<th>Style</th>
<th>Y Axis</th>
<th>Y Field</th>
<th>Smoothing</th>
</tr>
</thead>
<tbody>
<tr>
<td>All packets</td>
<td>tcp.analysis.flags</td>
<td>Line</td>
<td>Bytes</td>
<td>None</td>
<td>tcp.window...</td>
<td>None</td>
</tr>
<tr>
<td>TCP errors</td>
<td>tcp.stream eq 61</td>
<td>Bar</td>
<td>Packets</td>
<td>None</td>
<td>tcp.time_delta</td>
<td>None</td>
</tr>
<tr>
<td>Window Size</td>
<td>ip.src == 10.0.52.164</td>
<td>Dot</td>
<td>MIN(Y Field)</td>
<td>tcp.window...</td>
<td>None</td>
<td></td>
</tr>
</tbody>
</table>

Mouse drags
Interval 100 ms

#sf17eu • Estoril, Portugal
How to rule the world... by looking at packets!
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

<table>
<thead>
<tr>
<th>Time</th>
<th>Source</th>
<th>Destination</th>
<th>Length</th>
<th>Content Type</th>
<th>Info</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.021805</td>
<td>Client cheese.com</td>
<td>cheese.com</td>
<td>86</td>
<td>Handshake,Change Cipher Spec, Encry...</td>
<td>New Session Ticket, Change Cipher Spec, Encry...</td>
</tr>
<tr>
<td>0.000000</td>
<td>Client cheese.com</td>
<td>cheese.com</td>
<td>674</td>
<td>Application Data</td>
<td>Application Data</td>
</tr>
<tr>
<td>0.000000</td>
<td>cheese.com</td>
<td>Client cheese.com</td>
<td>1514</td>
<td>443</td>
<td>443 - 56994 [ACK] Seq=3169 Ack=880 Win=30848</td>
</tr>
<tr>
<td>0.000000</td>
<td>cheese.com</td>
<td>Client cheese.com</td>
<td>1514</td>
<td>443</td>
<td>443 - 56994 [ACK] Seq=4588 Ack=880 Win=30848</td>
</tr>
<tr>
<td>0.000000</td>
<td>cheese.com</td>
<td>Client cheese.com</td>
<td>1514</td>
<td>443</td>
<td>443 - 56994 [ACK] Seq=808 Ack=6816 Win=129632</td>
</tr>
<tr>
<td>0.000000</td>
<td>cheese.com</td>
<td>Client cheese.com</td>
<td>1514</td>
<td>443</td>
<td>443 - 56994 [ACK] Seq=7444 Ack=880 Win=30848</td>
</tr>
<tr>
<td>0.000000</td>
<td>cheese.com</td>
<td>Client cheese.com</td>
<td>1514</td>
<td>443</td>
<td>443 - 56994 [ACK] Seq=800 Ack=7444 Win=131072</td>
</tr>
<tr>
<td>0.000000</td>
<td>cheese.com</td>
<td>Client cheese.com</td>
<td>1514</td>
<td>443</td>
<td>443 - 56994 [ACK] Seq=7444 Ack=880 Win=30848</td>
</tr>
<tr>
<td>0.000000</td>
<td>Client cheese.com</td>
<td>cheese.com</td>
<td>1048</td>
<td>443</td>
<td>443 - 56994 [ACK] Seq=808 Ack=9834 Win=130680</td>
</tr>
<tr>
<td>0.000000</td>
<td>cheese.com</td>
<td>Client cheese.com</td>
<td>1514</td>
<td>443</td>
<td>443 - 56994 [ACK] Seq=9834 Ack=1447 Win=32008</td>
</tr>
<tr>
<td>0.000000</td>
<td>cheese.com</td>
<td>Client cheese.com</td>
<td>1514</td>
<td>443</td>
<td>443 - 56994 [ACK] Seq=1447 Ack=11764 Win=1305</td>
</tr>
<tr>
<td>0.000000</td>
<td>cheese.com</td>
<td>Client cheese.com</td>
<td>1514</td>
<td>443</td>
<td>443 - 56994 [ACK] Seq=1447 Ack=11764 Win=1305</td>
</tr>
</tbody>
</table>

How to rule the world... by looking at packets!
Visualization & Creating Reports

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

Worst Performing Microsoft Servers

<table>
<thead>
<tr>
<th>Information</th>
<th>Number (Inbound) [kb/s]</th>
<th>Number (Outbound) [kb/s]</th>
<th>Connections (TCP Servers) [#]</th>
<th>Connections Failed (TCP Servers) [#]</th>
<th>Server Response Time (Servers) [msec]</th>
</tr>
</thead>
<tbody>
<tr>
<td>Microsoft Cloud</td>
<td>24264.253</td>
<td>6146.303</td>
<td>18117.950</td>
<td>443586.000</td>
<td>296846.000</td>
</tr>
<tr>
<td></td>
<td>1.801</td>
<td>0.845</td>
<td>0.857</td>
<td>1004.000</td>
<td>20129.000</td>
</tr>
<tr>
<td></td>
<td>1.763</td>
<td>0.924</td>
<td>0.839</td>
<td>968.000</td>
<td>19667.000</td>
</tr>
<tr>
<td></td>
<td>1.695</td>
<td>0.887</td>
<td>0.808</td>
<td>907.000</td>
<td>19346.000</td>
</tr>
<tr>
<td></td>
<td>1.697</td>
<td>0.888</td>
<td>0.808</td>
<td>909.000</td>
<td>19345.000</td>
</tr>
<tr>
<td></td>
<td>1.695</td>
<td>0.888</td>
<td>0.806</td>
<td>904.000</td>
<td>19314.000</td>
</tr>
<tr>
<td></td>
<td>1.678</td>
<td>0.879</td>
<td>0.799</td>
<td>882.000</td>
<td>19280.000</td>
</tr>
<tr>
<td></td>
<td>1.699</td>
<td>0.889</td>
<td>0.809</td>
<td>923.000</td>
<td>19249.000</td>
</tr>
<tr>
<td></td>
<td>1.689</td>
<td>0.884</td>
<td>0.804</td>
<td>914.000</td>
<td>19156.000</td>
</tr>
<tr>
<td></td>
<td>1.672</td>
<td>0.876</td>
<td>0.796</td>
<td>887.000</td>
<td>19115.000</td>
</tr>
<tr>
<td></td>
<td>1.678</td>
<td>0.880</td>
<td>0.798</td>
<td>904.000</td>
<td>19085.000</td>
</tr>
</tbody>
</table>

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

Looking at packets!
### Use Excel For Pie Charts and Graphs

#### Total Duration = 20 seconds

<table>
<thead>
<tr>
<th></th>
<th>Client Delay</th>
<th>Server Delay</th>
<th>Network Latency</th>
<th>Network Transit Time / Congestion</th>
</tr>
</thead>
<tbody>
<tr>
<td>SUMS</td>
<td>15.291396</td>
<td>15.181011</td>
<td>0.063915</td>
<td>0.1677</td>
</tr>
<tr>
<td>Min RTT</td>
<td>0.01677</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

#### Example Data

<table>
<thead>
<tr>
<th>J</th>
<th>K</th>
<th>L</th>
<th>M</th>
<th>N</th>
<th>O</th>
<th>P</th>
<th>Q</th>
<th>R</th>
<th>S</th>
<th>T</th>
</tr>
</thead>
<tbody>
<tr>
<td>Seq</td>
<td>Len</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>8586</td>
<td>0.019188</td>
<td>0.019188</td>
<td>0</td>
<td>0.01677</td>
</tr>
<tr>
<td>32</td>
<td>193</td>
<td>126</td>
<td>319</td>
<td>6586</td>
<td>0.019752</td>
<td>0.019752</td>
<td>0</td>
<td>0.01677</td>
<td>5511</td>
<td></td>
</tr>
<tr>
<td>32</td>
<td>319</td>
<td>362</td>
<td>681</td>
<td>6828</td>
<td>0.155389</td>
<td>0.155282</td>
<td>0.000107</td>
<td>0.01677</td>
<td></td>
<td></td>
</tr>
<tr>
<td>32</td>
<td>681</td>
<td>454</td>
<td>1135</td>
<td>17116</td>
<td>0.01982</td>
<td>0.01982</td>
<td>0</td>
<td>0.01677</td>
<td></td>
<td></td>
</tr>
<tr>
<td>32</td>
<td>1135</td>
<td>449</td>
<td>1584</td>
<td>17939</td>
<td>0.042088</td>
<td>0.02155</td>
<td>0.020538</td>
<td>0.01677</td>
<td></td>
<td></td>
</tr>
<tr>
<td>32</td>
<td>1584</td>
<td>409</td>
<td>1993</td>
<td>82217</td>
<td>0.023314</td>
<td>0.023049</td>
<td>0.000265</td>
<td>0.01677</td>
<td></td>
<td></td>
</tr>
<tr>
<td>32</td>
<td>1993</td>
<td>411</td>
<td>2404</td>
<td>95810</td>
<td>0.023031</td>
<td>0.02195</td>
<td>0.001081</td>
<td>0.01677</td>
<td></td>
<td></td>
</tr>
<tr>
<td>32</td>
<td>2404</td>
<td>458</td>
<td>2862</td>
<td>139396</td>
<td>0.06324</td>
<td>0.01677</td>
<td>0</td>
<td>0.01677</td>
<td></td>
<td></td>
</tr>
<tr>
<td>32</td>
<td>2862</td>
<td>1358</td>
<td>4220</td>
<td>140619</td>
<td>14.888013</td>
<td>14.856046</td>
<td>0.022967</td>
<td>0.01677</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
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!