#sf21veu

# When It's NOT a "Network Problem": Identifying Higher-Layer Issues in Packet Data

**Wes Morgan**
HCL Software

# Wes Morgan

#sf21veu

Full-stack troubleshooter

Longtime UNIX/Linux sysadmin

Old-school open source contributor

(nethack, gnuplot, identd, etc.)

Dates back to punched cards and paper tape

Passionate about a cappella music

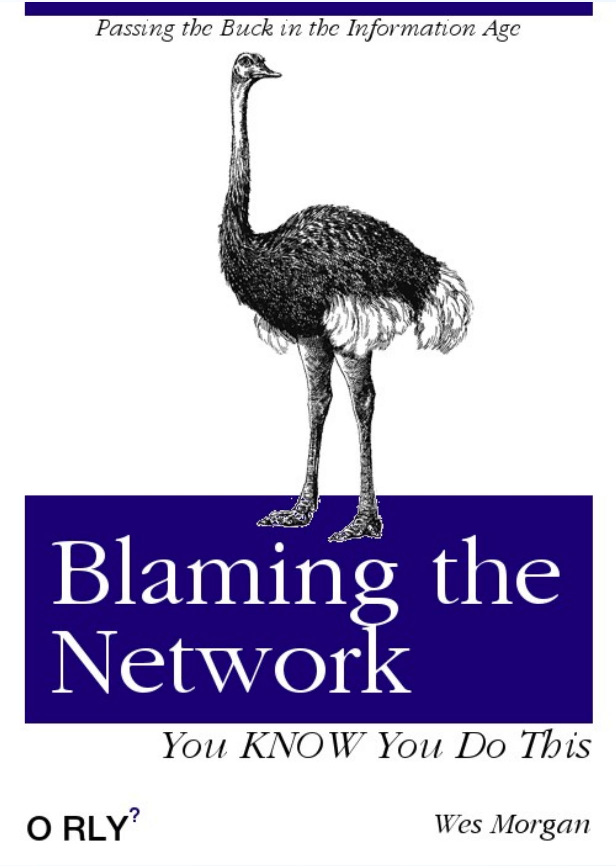
Emulation fan (simh, Hercules, MAME)

First build: Ethereal 0.4, Red Hat 5.2, ThinkPad 560Z

Twitter: @wesmorgan1

## Why Are We Here?

- “It could be the network” is all too common
- It’s only correct (in my experience) about 10-12% of the time
- However, we often see clues to OS and/or application issues in packet data



#sf21veu

# Before you start

- Understand the environment
  - Physical hardware or VM?
  - Storage local or remote?
  - Intermediate devices (e.g. firewall, proxy, etc.)?
- Rule out network-layer concerns
  - TCP retransmissions
  - Congestion/loss issues
- Understand the application in question and the protocols it uses
- Understand the problem
  - Constant or intermittent?
  - Easily reproduced?

# Case study: Simple timing analysis

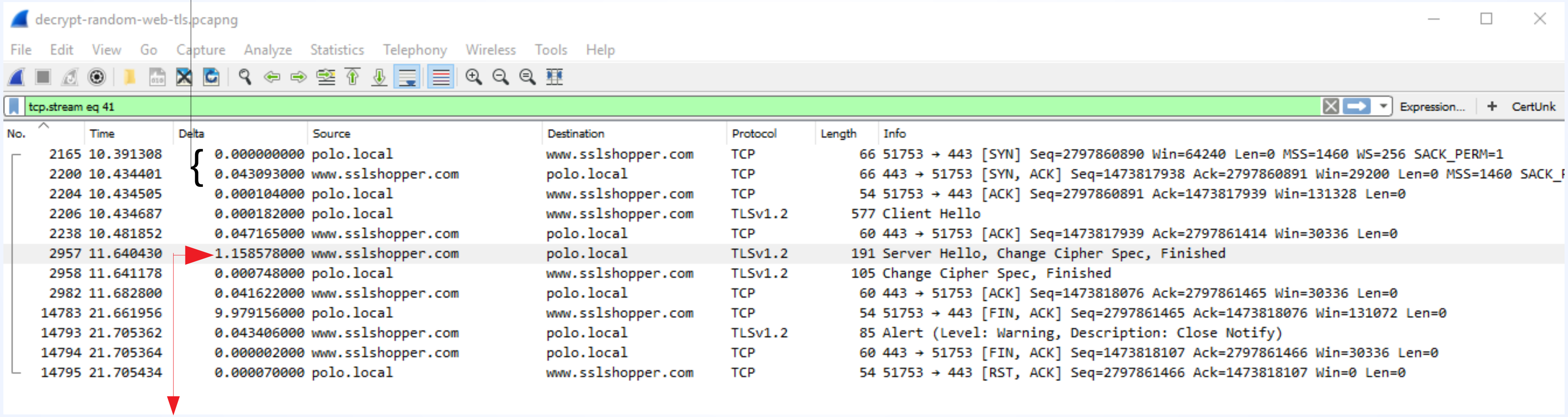
#sf21veu

- **tcp.time_delta** is most useful
  - Consider adding it as a column
- Identify request/response patterns
  - Correlate with application/OS logs when possible
- Consider the nature of the request
  - It may lead you elsewhere

# Simple timing analysis – Example 1

#sf21veu

TCP handshake and other acknowledgments suggest network latency of ~43 ms

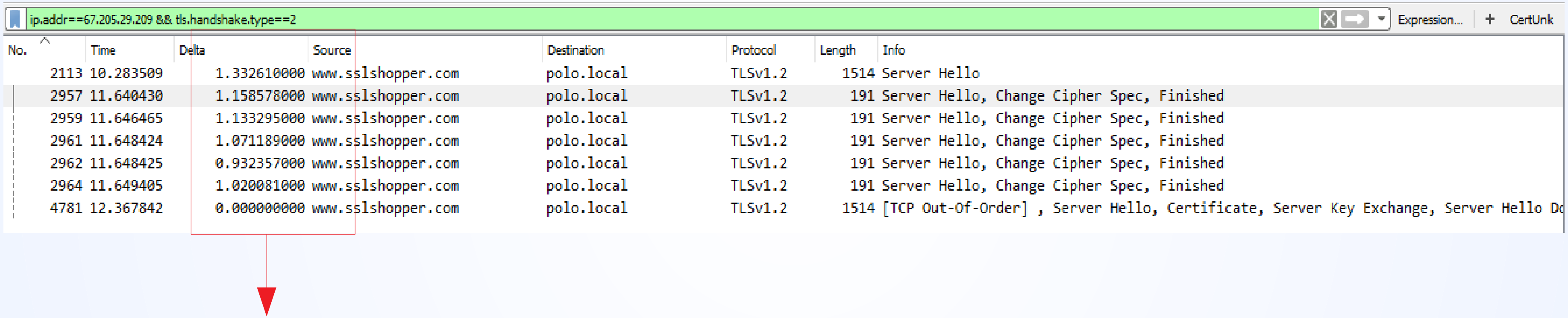


tcp.stream eq 41

```
No.      Time       Delta        Source              Destination          Protocol   Length Info
 2165 10.391308   0.000000000 polo.local          www.sslshopper.com   TCP         66 51753 → 443 [SYN] Seq=2797860890 Win=64240 Len=0 MSS=1460 WS=256 SACK_PERM=1
 2200 10.434401   0.043093000 www.sslshopper.com  polo.local           TCP         66 443 → 51753 [SYN, ACK] Seq=1473817938 Ack=2797860891 Win=29200 Len=0 MSS=1460 SACK_
 2204 10.434505   0.000104000 polo.local          www.sslshopper.com   TCP         54 51753 → 443 [ACK] Seq=2797860891 Ack=1473817939 Win=131328 Len=0
 2206 10.434687   0.000182000 polo.local          www.sslshopper.com   TLSv1.2    577 Client Hello
 2238 10.481852   0.047165000 www.sslshopper.com  polo.local           TCP         60 443 → 51753 [ACK] Seq=1473817939 Ack=2797861414 Win=30336 Len=0
 2957 11.640430   1.158578000 www.sslshopper.com  polo.local           TLSv1.2    191 Server Hello, Change Cipher Spec, Finished
 2958 11.641178   0.000748000 polo.local          www.sslshopper.com   TLSv1.2    105 Change Cipher Spec, Finished
 2982 11.682800   0.041622000 www.sslshopper.com  polo.local           TCP         60 443 → 51753 [ACK] Seq=1473818076 Ack=2797861465 Win=30336 Len=0
14783 21.661956   9.979156000 polo.local          www.sslshopper.com   TCP         54 51753 → 443 [FIN, ACK] Seq=2797861465 Ack=1473818076 Win=131072 Len=0
14793 21.705362   0.043406000 www.sslshopper.com  polo.local           TLSv1.2     85 Alert (Level: Warning, Description: Close Notify)
14794 21.705364   0.000002000 www.sslshopper.com  polo.local           TCP         60 443 → 51753 [FIN, ACK] Seq=1473818107 Ack=2797861466 Win=30336 Len=0
14795 21.705434   0.000070000 polo.local          www.sslshopper.com   TCP         54 51753 → 443 [RST, ACK] Seq=2797861466 Ack=1473818107 Win=0 Len=0
```

1158ms – 43ms = 1115ms to process a Client Hello? Let’s check that out...

# Simple timing analysis – Example 1



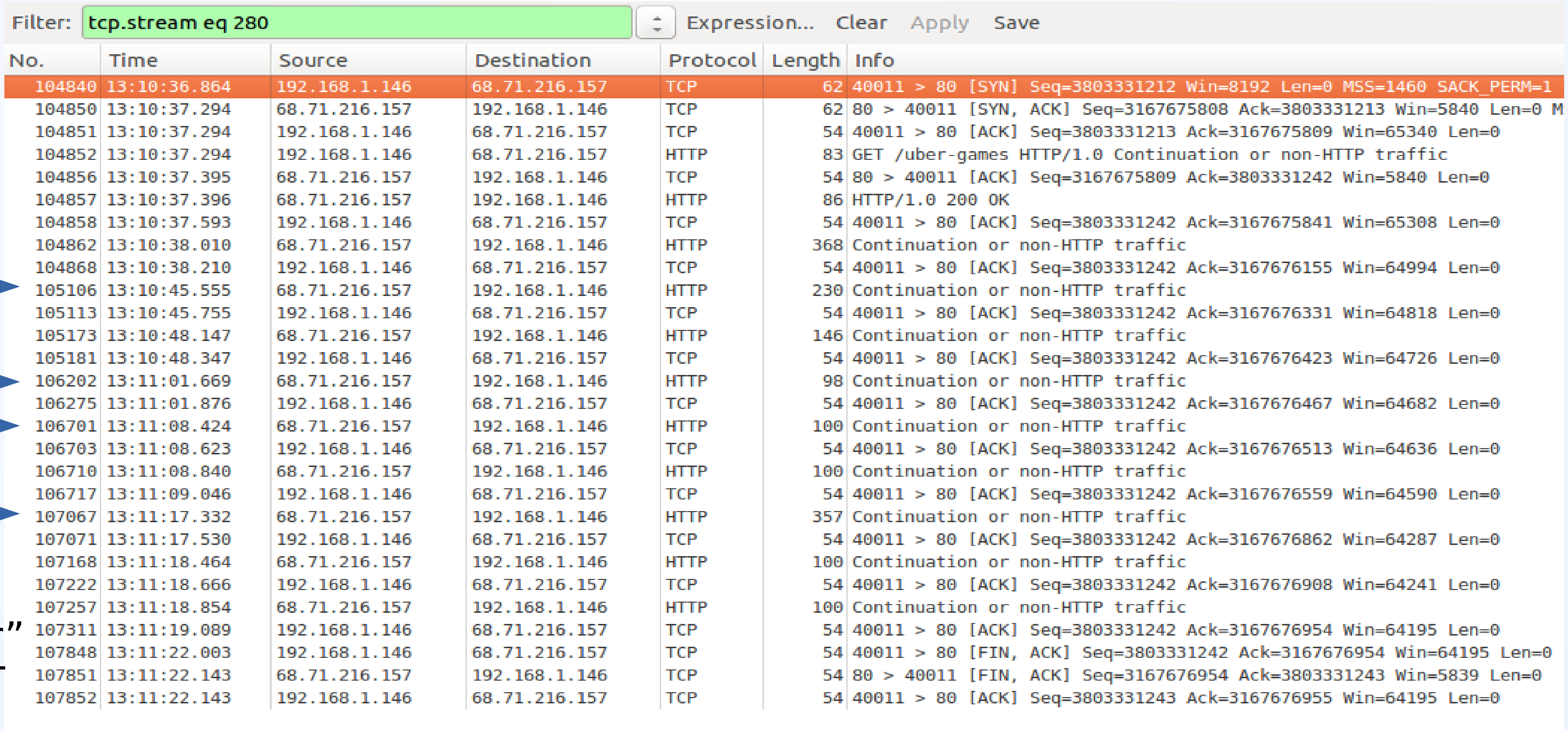
```
ip.addr==67.205.29.209 && tls.handshake.type==2
```

```
No.    Time       Delta        Source             Destination  Protocol  Length Info
  2113 10.283509  1.332610000 www.sslshopper.com  polo.local   TLSv1.2     1514 Server Hello
  2957 11.640430  1.158578000 www.sslshopper.com  polo.local   TLSv1.2      191 Server Hello, Change Cipher Spec, Finished
  2959 11.646465  1.133295000 www.sslshopper.com  polo.local   TLSv1.2      191 Server Hello, Change Cipher Spec, Finished
  2961 11.648424  1.071189000 www.sslshopper.com  polo.local   TLSv1.2      191 Server Hello, Change Cipher Spec, Finished
  2962 11.648425  0.932357000 www.sslshopper.com  polo.local   TLSv1.2      191 Server Hello, Change Cipher Spec, Finished
  2964 11.649405  1.020081000 www.sslshopper.com  polo.local   TLSv1.2      191 Server Hello, Change Cipher Spec, Finished
  4781 12.367842  0.000000000 www.sslshopper.com  polo.local   TLSv1.2     1514 [TCP Out-Of-Order] , Server Hello, Certificate, Server Key Exchange, Server Hello Do
```

Consistent 1000+ ms to process Client Hellos?
Might want to investigate that...

# Simple timing analysis – Example 2

#sf21veu



Filter: tcp.stream eq 280

```
No.     Time          Source          Destination     Protocol Length Info
 104840 13:10:36.864 192.168.1.146   68.71.216.157   TCP          62 40011 > 80 [SYN] Seq=3803331212 Win=8192 Len=0 MSS=1460 SACK_PERM=1
 104850 13:10:37.294 68.71.216.157   192.168.1.146   TCP          62 80 > 40011 [SYN, ACK] Seq=3167675808 Ack=3803331213 Win=5840 Len=0 M
 104851 13:10:37.294 192.168.1.146   68.71.216.157   TCP          54 40011 > 80 [ACK] Seq=3803331213 Ack=3167675809 Win=65340 Len=0
 104852 13:10:37.294 192.168.1.146   68.71.216.157   HTTP         83 GET /uber-games HTTP/1.0 Continuation or non-HTTP traffic
 104856 13:10:37.395 68.71.216.157   192.168.1.146   TCP          54 80 > 40011 [ACK] Seq=3167675809 Ack=3803331242 Win=5840 Len=0
 104857 13:10:37.396 68.71.216.157   192.168.1.146   HTTP         86 HTTP/1.0 200 OK
 104858 13:10:37.593 192.168.1.146   68.71.216.157   TCP          54 40011 > 80 [ACK] Seq=3803331242 Ack=3167675841 Win=65308 Len=0
 104862 13:10:38.010 68.71.216.157   192.168.1.146   HTTP        368 Continuation or non-HTTP traffic
 104868 13:10:38.210 192.168.1.146   68.71.216.157   TCP          54 40011 > 80 [ACK] Seq=3803331242 Ack=3167676155 Win=64994 Len=0
 105106 13:10:45.555 68.71.216.157   192.168.1.146   HTTP        230 Continuation or non-HTTP traffic
 105113 13:10:45.755 192.168.1.146   68.71.216.157   TCP          54 40011 > 80 [ACK] Seq=3803331242 Ack=3167676331 Win=64818 Len=0
 105173 13:10:48.147 68.71.216.157   192.168.1.146   HTTP        146 Continuation or non-HTTP traffic
 105181 13:10:48.347 192.168.1.146   68.71.216.157   TCP          54 40011 > 80 [ACK] Seq=3803331242 Ack=3167676423 Win=64726 Len=0
 106202 13:11:01.669 68.71.216.157   192.168.1.146   HTTP         98 Continuation or non-HTTP traffic
 106275 13:11:01.876 192.168.1.146   68.71.216.157   TCP          54 40011 > 80 [ACK] Seq=3803331242 Ack=3167676467 Win=64682 Len=0
 106701 13:11:08.424 68.71.216.157   192.168.1.146   HTTP        100 Continuation or non-HTTP traffic
 106703 13:11:08.623 192.168.1.146   68.71.216.157   TCP          54 40011 > 80 [ACK] Seq=3803331242 Ack=3167676513 Win=64636 Len=0
 106710 13:11:08.840 68.71.216.157   192.168.1.146   HTTP        100 Continuation or non-HTTP traffic
 106717 13:11:09.046 192.168.1.146   68.71.216.157   TCP          54 40011 > 80 [ACK] Seq=3803331242 Ack=3167676559 Win=64590 Len=0
 107067 13:11:17.332 68.71.216.157   192.168.1.146   HTTP        357 Continuation or non-HTTP traffic
 107071 13:11:17.530 192.168.1.146   68.71.216.157   TCP          54 40011 > 80 [ACK] Seq=3803331242 Ack=3167676862 Win=64287 Len=0
 107168 13:11:18.464 68.71.216.157   192.168.1.146   HTTP        100 Continuation or non-HTTP traffic
 107222 13:11:18.666 192.168.1.146   68.71.216.157   TCP          54 40011 > 80 [ACK] Seq=3803331242 Ack=3167676908 Win=64241 Len=0
 107257 13:11:18.854 68.71.216.157   192.168.1.146   HTTP        100 Continuation or non-HTTP traffic
 107311 13:11:19.089 192.168.1.146   68.71.216.157   TCP          54 40011 > 80 [ACK] Seq=3803331242 Ack=3167676954 Win=64195 Len=0
 107848 13:11:22.003 192.168.1.146   68.71.216.157   TCP          54 40011 > 80 [FIN, ACK] Seq=3803331242 Ack=3167676954 Win=64195 Len=0
 107851 13:11:22.143 68.71.216.157   192.168.1.146   TCP          54 80 > 40011 [FIN, ACK] Seq=3167676954 Ack=3803331243 Win=5839 Len=0
 107852 13:11:22.143 192.168.1.146   68.71.216.157   TCP          54 40011 > 80 [ACK] Seq=3803331243 Ack=3167676955 Win=64195 Len=0
```

Look for
“Dead air”

7s

13s

7s

8s

35s “dead air”
in a 46s GET

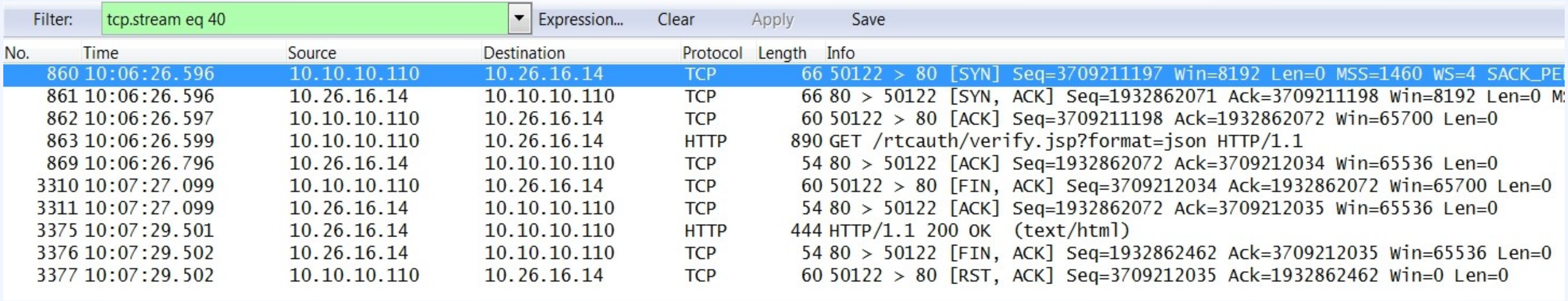
# Case study: “response after client gave up”

#sf21veu

- Marked by traffic after client FIN
  - Often across multiple conversations
  - May be intermittent, based on transaction load
- Examine the request
  - Understand the nature of the request
  - Determine external dependcies (if any)
- Be prepared to be pointed elsewhere for resolution

# Server response after client gave up - example

#sf21veu



Filter: tcp.stream eq 40

```
No.   Time          Source         Destination    Protocol Length Info
  860 10:06:26.596  10.10.10.110   10.26.16.14    TCP        66 50122 > 80 [SYN] Seq=3709211197 Win=8192 Len=0 MSS=1460 WS=4 SACK_PE
  861 10:06:26.596  10.26.16.14    10.10.10.110   TCP        66 80 > 50122 [SYN, ACK] Seq=1932862071 Ack=3709211198 Win=8192 Len=0 M
  862 10:06:26.597  10.10.10.110   10.26.16.14    TCP        60 50122 > 80 [ACK] Seq=3709211198 Ack=1932862072 Win=65700 Len=0
  863 10:06:26.599  10.10.10.110   10.26.16.14    HTTP      890 GET /rtcauth/verify.jsp?format=json HTTP/1.1
  869 10:06:26.796  10.26.16.14    10.10.10.110   TCP        54 80 > 50122 [ACK] Seq=1932862072 Ack=3709212034 Win=65536 Len=0
 3310 10:07:27.099  10.10.10.110   10.26.16.14    TCP        60 50122 > 80 [FIN, ACK] Seq=3709212034 Ack=1932862072 Win=65700 Len=0
 3311 10:07:27.099  10.26.16.14    10.10.10.110   TCP        54 80 > 50122 [ACK] Seq=1932862072 Ack=3709212035 Win=65536 Len=0
 3375 10:07:29.501  10.26.16.14    10.10.10.110   HTTP      444 HTTP/1.1 200 OK  (text/html)
 3376 10:07:29.502  10.26.16.14    10.10.10.110   TCP        54 80 > 50122 [FIN, ACK] Seq=1932862462 Ack=3709212035 Win=65536 Len=0
 3377 10:07:29.502  10.10.10.110   10.26.16.14    TCP        60 50122 > 80 [RST, ACK] Seq=3709212035 Ack=1932862462 Win=0 Len=0
```

- Both sequence and timing are significant
- TCP handshake and client request are as expected and process quickly
- 60 seconds of ‘dead air’  (NEVER TRUST ROUND NUMBERS!)
- Client times out, sends FIN

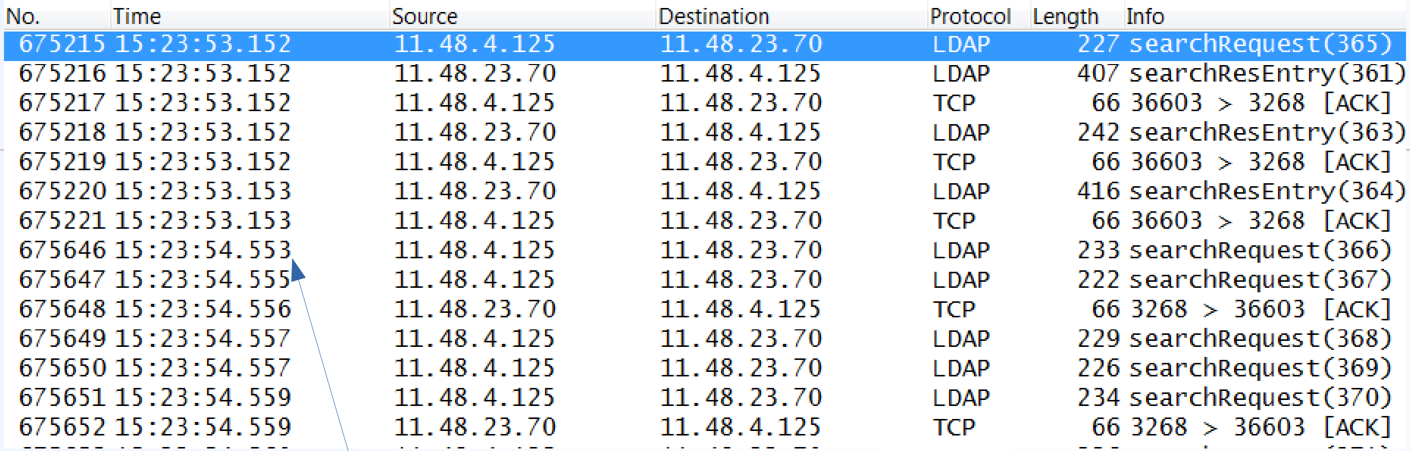
- Server acknowledges FIN *immediately* (i.e. TCP/IP stack is responsive)
- Server delivers response 2 seconds later, but client has already timed out
- HINT: Examine the specific request
- Root cause: Extremely slow response time for backend LDAP server

# Case study: “request backlog”

- Absence of TCP/IP error conditions
  - No packet loss/congestion
- Look for periods of “dead air” in service responses
  - Additional requests may or may not be accepted
- Usually cleared with a flood of responses

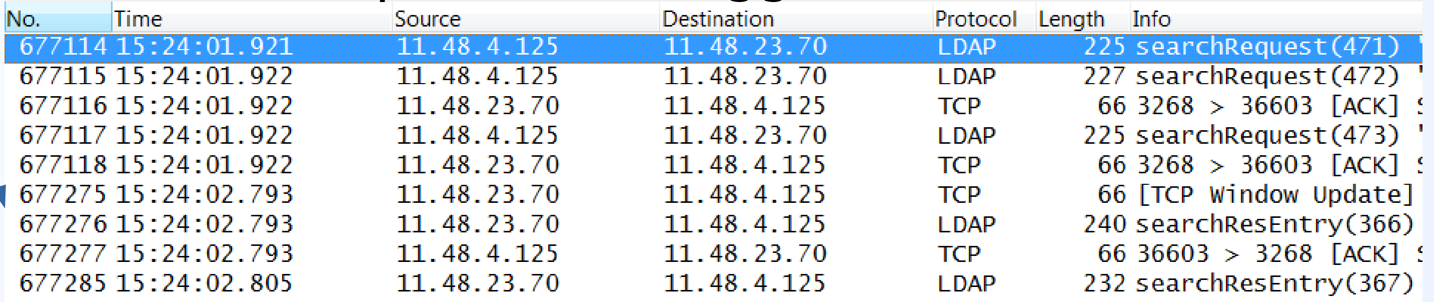
# Request backlog - example

#sf21veu



| No. | Time | Source | Destination | Protocol | Length | Info |
|---|---|---|---|---|---|---|
| 675215 | 15:23:53.152 | 11.48.4.125 | 11.48.23.70 | LDAP | 227 | searchRequest(365) |
| 675216 | 15:23:53.152 | 11.48.23.70 | 11.48.4.125 | LDAP | 407 | searchResEntry(361) |
| 675217 | 15:23:53.152 | 11.48.4.125 | 11.48.23.70 | TCP | 66 | 36603 > 3268 [ACK] |
| 675218 | 15:23:53.152 | 11.48.23.70 | 11.48.4.125 | LDAP | 242 | searchResEntry(363) |
| 675219 | 15:23:53.152 | 11.48.4.125 | 11.48.23.70 | TCP | 66 | 36603 > 3268 [ACK] |
| 675220 | 15:23:53.153 | 11.48.23.70 | 11.48.4.125 | LDAP | 416 | searchResEntry(364) |
| 675221 | 15:23:53.153 | 11.48.4.125 | 11.48.23.70 | TCP | 66 | 36603 > 3268 [ACK] |
| 675646 | 15:23:54.553 | 11.48.4.125 | 11.48.23.70 | LDAP | 233 | searchRequest(366) |
| 675647 | 15:23:54.555 | 11.48.4.125 | 11.48.23.70 | LDAP | 222 | searchRequest(367) |
| 675648 | 15:23:54.556 | 11.48.23.70 | 11.48.4.125 | TCP | 66 | 3268 > 36603 [ACK] |
| 675649 | 15:23:54.557 | 11.48.4.125 | 11.48.23.70 | LDAP | 229 | searchRequest(368) |
| 675650 | 15:23:54.557 | 11.48.4.125 | 11.48.23.70 | LDAP | 226 | searchRequest(369) |
| 675651 | 15:23:54.559 | 11.48.4.125 | 11.48.23.70 | LDAP | 234 | searchRequest(370) |
| 675652 | 15:23:54.559 | 11.48.23.70 | 11.48.4.125 | TCP | 66 | 3268 > 36603 [ACK] |

- No TCP/IP errors
- 7s without server response
- 110+ requests backlogged!



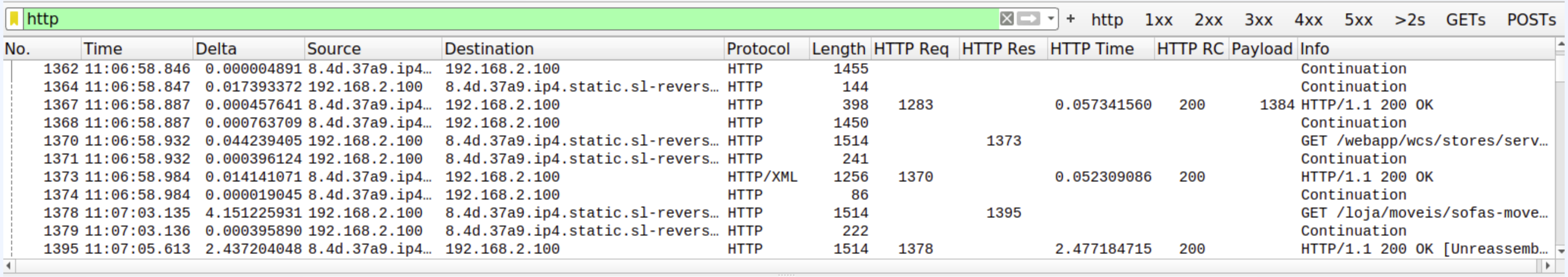
| No. | Time | Source | Destination | Protocol | Length | Info |
|---|---|---|---|---|---|---|
| 677114 | 15:24:01.921 | 11.48.4.125 | 11.48.23.70 | LDAP | 225 | searchRequest(471) |
| 677115 | 15:24:01.922 | 11.48.4.125 | 11.48.23.70 | LDAP | 227 | searchRequest(472) |
| 677116 | 15:24:01.922 | 11.48.23.70 | 11.48.4.125 | TCP | 66 | 3268 > 36603 [ACK] |
| 677117 | 15:24:01.922 | 11.48.4.125 | 11.48.23.70 | LDAP | 225 | searchRequest(473) |
| 677118 | 15:24:01.922 | 11.48.23.70 | 11.48.4.125 | TCP | 66 | 3268 > 36603 [ACK] |
| 677275 | 15:24:02.793 | 11.48.23.70 | 11.48.4.125 | TCP | 66 | [TCP Window Update] |
| 677276 | 15:24:02.793 | 11.48.23.70 | 11.48.4.125 | LDAP | 240 | searchResEntry(366) |
| 677277 | 15:24:02.793 | 11.48.4.125 | 11.48.23.70 | TCP | 66 | 36603 > 3268 [ACK] |
| 677285 | 15:24:02.805 | 11.48.23.70 | 11.48.4.125 | LDAP | 232 | searchResEntry(367) |

# Case study: HTTP Response Time

- **http.time** is your friend!
- Two different measurements, depending on configuration
  - Enable all HTTP reassembly preferences
  - If TCP preference “Allow subdissector to reassemble TCP streams” is OFF, **http.time** will be time to first response packet (the one with the HTTP response code)
  - If it’s ON, **http.time** will be time to LAST packet of response

# HTTP Response Time - Example

#sf21veu

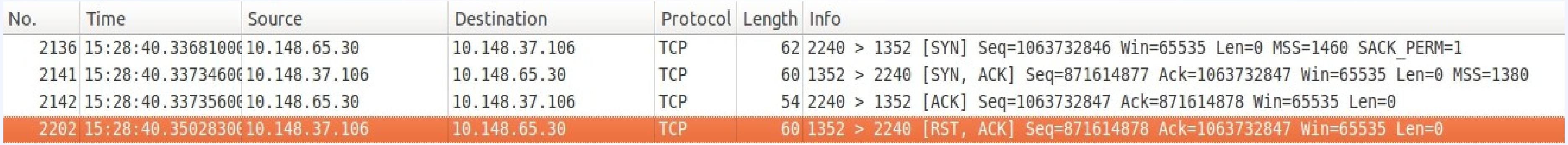


| No. | Time | Delta | Source | Destination | Protocol | Length | HTTP Req | HTTP Res | HTTP Time | HTTP RC | Payload | Info |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 1362 | 11:06:58.846 | 0.000004891 | 8.4d.37a9.ip4… | 192.168.2.100 | HTTP | 1455 | | | | | | Continuation |
| 1364 | 11:06:58.847 | 0.017393372 | 192.168.2.100 | 8.4d.37a9.ip4.static.sl-revers… | HTTP | 144 | | | | | | Continuation |
| 1367 | 11:06:58.887 | 0.000457641 | 8.4d.37a9.ip4… | 192.168.2.100 | HTTP | 398 | 1283 | | 0.057341560 | 200 | 1384 | HTTP/1.1 200 OK |
| 1368 | 11:06:58.887 | 0.000763709 | 8.4d.37a9.ip4… | 192.168.2.100 | HTTP | 1450 | | | | | | Continuation |
| 1370 | 11:06:58.932 | 0.044239405 | 192.168.2.100 | 8.4d.37a9.ip4.static.sl-revers… | HTTP | 1514 | | 1373 | | | | GET /webapp/wcs/stores/serv… |
| 1371 | 11:06:58.932 | 0.000396124 | 192.168.2.100 | 8.4d.37a9.ip4.static.sl-revers… | HTTP | 241 | | | | | | Continuation |
| 1373 | 11:06:58.984 | 0.014141071 | 8.4d.37a9.ip4… | 192.168.2.100 | HTTP/XML | 1256 | 1370 | | 0.052309086 | 200 | | HTTP/1.1 200 OK |
| 1374 | 11:06:58.984 | 0.000019045 | 8.4d.37a9.ip4… | 192.168.2.100 | HTTP | 86 | | | | | | Continuation |
| 1378 | 11:07:03.135 | 4.151225931 | 192.168.2.100 | 8.4d.37a9.ip4.static.sl-revers… | HTTP | 1514 | | 1395 | | | | GET /loja/moveis/sofas-move… |
| 1379 | 11:07:03.136 | 0.000395890 | 192.168.2.100 | 8.4d.37a9.ip4.static.sl-revers… | HTTP | 222 | | | | | | Continuation |
| 1395 | 11:07:05.613 | 2.437204048 | 8.4d.37a9.ip4… | 192.168.2.100 | HTTP | 1514 | 1378 | | 2.477184715 | 200 | | HTTP/1.1 200 OK [Unreassemb… |

- We are NOT reassembling TCP streams, so **http.time** is the time between the request and the first packet of the response
- We see three HTTP transactions – two requests get a response within ~55ms, but one is delayed for almost 2.5s
- Further investigation found consistent delays in a particular type of GET request (pictures)
- Ultimate root cause was poor performance on the server hosting images retrieved via GET

# Case study: TCP handshake - and nothing more

#sf21veu

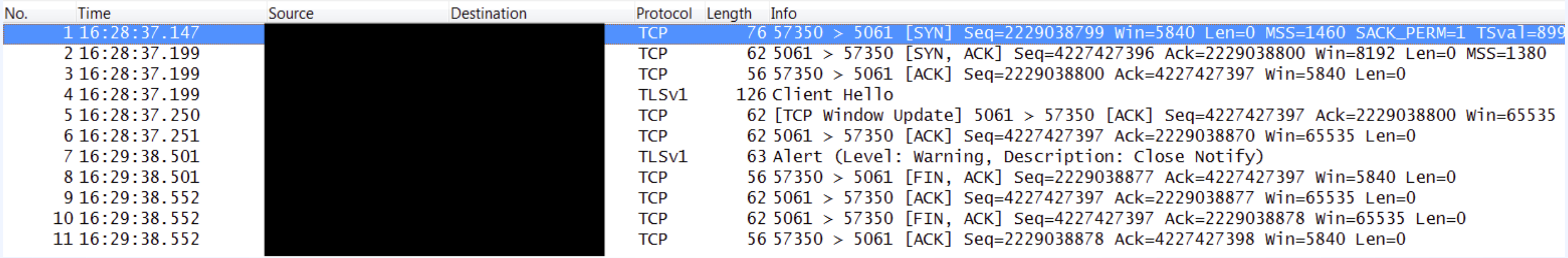


| No. | Time | Source | Destination | Protocol | Length | Info |
|---|---|---|---|---|---|---|
| 2136 | 15:28:40.33681000 | 10.148.65.30 | 10.148.37.106 | TCP | 62 | 2240 > 1352 [SYN] Seq=1063732846 Win=65535 Len=0 MSS=1460 SACK_PERM=1 |
| 2141 | 15:28:40.33734600 | 10.148.37.106 | 10.148.65.30 | TCP | 60 | 1352 > 2240 [SYN, ACK] Seq=871614877 Ack=1063732847 Win=65535 Len=0 MSS=1380 |
| 2142 | 15:28:40.33735600 | 10.148.65.30 | 10.148.37.106 | TCP | 54 | 2240 > 1352 [ACK] Seq=1063732847 Ack=871614878 Win=65535 Len=0 |
| 2202 | 15:28:40.35028300 | 10.148.37.106 | 10.148.65.30 | TCP | 60 | 1352 > 2240 [RST, ACK] Seq=871614878 Ack=1063732847 Win=65535 Len=0 |

- TCP/IP stack handles initial handshake
- When TCP handshake completes, conversation is moved to a *listener backlog queue*
- Each 'listening' application has its own backlog queue
- If the application does not “pick up” the conversation from its backlog queue within a reasonable amount of time, the TCP/IP stack kills it with an RST
- If the backlog queue is full, any additional conversation attempts are rejected with an RST
  - Particularly likely if “can’t connect” only occurs during heavy server load
- Resolution: increase backlog queue size at OS or application layer

# Case study: Check Every Layer!

#sf21veu



| No. | Time | Source | Destination | Protocol | Length | Info |
|---|---|---|---|---|---|---|
| 1 | 16:28:37.147 | [redacted] | [redacted] | TCP | 76 | 57350 > 5061 [SYN] Seq=2229038799 Win=5840 Len=0 MSS=1460 SACK_PERM=1 TSval=899 |
| 2 | 16:28:37.199 | [redacted] | [redacted] | TCP | 62 | 5061 > 57350 [SYN, ACK] Seq=4227427396 Ack=2229038800 Win=8192 Len=0 MSS=1380 |
| 3 | 16:28:37.199 | [redacted] | [redacted] | TCP | 56 | 57350 > 5061 [ACK] Seq=2229038800 Ack=4227427397 Win=5840 Len=0 |
| 4 | 16:28:37.199 | [redacted] | [redacted] | TLSv1 | 126 | Client Hello |
| 5 | 16:28:37.250 | [redacted] | [redacted] | TCP | 62 | [TCP Window Update] 5061 > 57350 [ACK] Seq=4227427397 Ack=2229038800 Win=65535 |
| 6 | 16:28:37.251 | [redacted] | [redacted] | TCP | 62 | 5061 > 57350 [ACK] Seq=4227427397 Ack=2229038870 Win=65535 Len=0 |
| 7 | 16:29:38.501 | [redacted] | [redacted] | TLSv1 | 63 | Alert (Level: Warning, Description: Close Notify) |
| 8 | 16:29:38.501 | [redacted] | [redacted] | TCP | 56 | 57350 > 5061 [FIN, ACK] Seq=2229038877 Ack=4227427397 Win=5840 Len=0 |
| 9 | 16:29:38.552 | [redacted] | [redacted] | TCP | 62 | 5061 > 57350 [ACK] Seq=4227427397 Ack=2229038877 Win=65535 Len=0 |
| 10 | 16:29:38.552 | [redacted] | [redacted] | TCP | 62 | 5061 > 57350 [FIN, ACK] Seq=4227427397 Ack=2229038878 Win=65535 Len=0 |
| 11 | 16:29:38.552 | [redacted] | [redacted] | TCP | 56 | 57350 > 5061 [ACK] Seq=2229038878 Ack=4227427398 Win=5840 Len=0 |

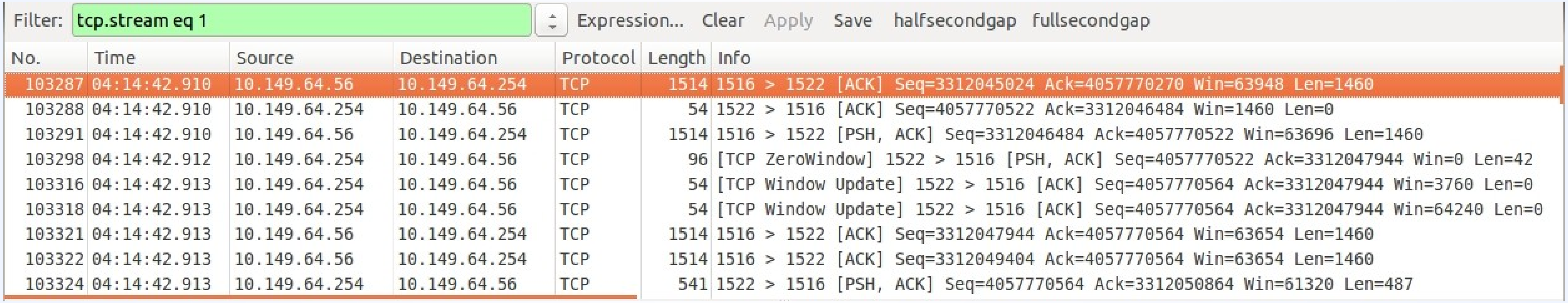
- Issue presented as “VoIP problem – call failed”
- Initial troubleshooting focused on SIP stack
- Key factor – not just SIP, but SIPS (SIP over TLS) was in use
- TLS investigation showed that the TLS handshake failed – should have seen “Server Hello” in response to “Client Hello”, but saw “Close Notify” instead
- Root cause: Error in server TLS configuration

# Case study: TCP Zero Windows

- Indicates that the TCP receive buffer has filled
  - No further data can be sent until space is available
- Critical question – WHY has the receive buffer filled?
- A few short-lived instances (e.g. 10-20ms) may just be a hiccup, but check them anyway
- Lengthy/multiple instances (or simultaneous instances across multiple conversations) almost always indicate resource contention or threading issues at higher layers

# TCP Zero Windows - Example 1

#sf21veu

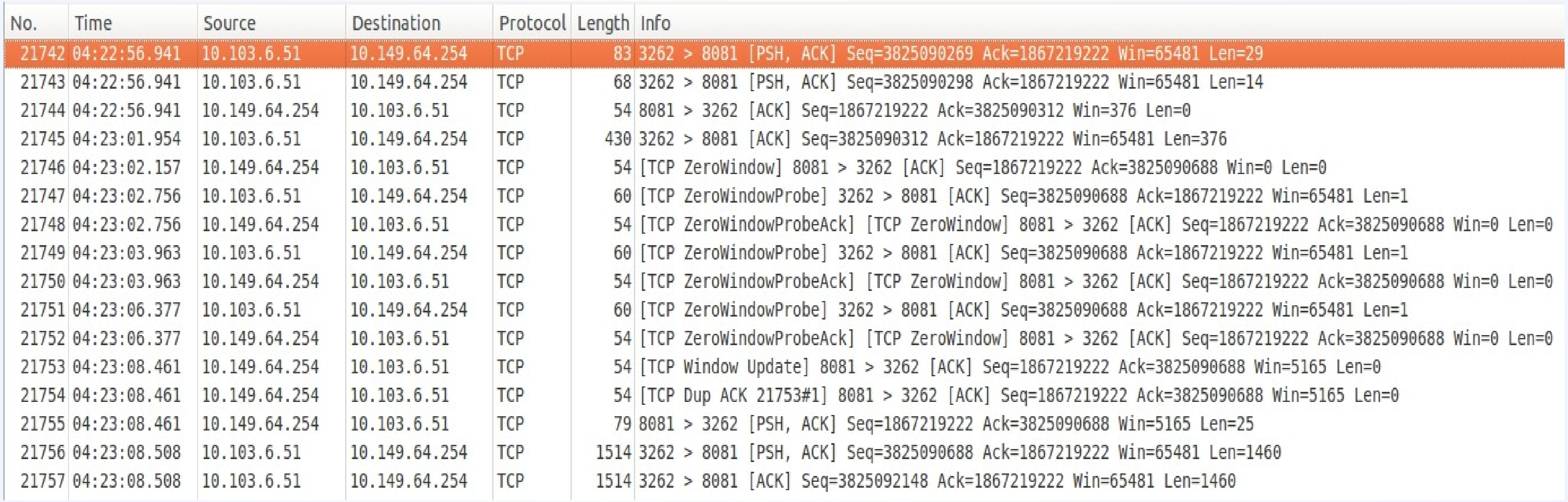


Filter: tcp.stream eq 1

| No. | Time | Source | Destination | Protocol | Length | Info |
|---|---|---|---|---|---|---|
| 103287 | 04:14:42.910 | 10.149.64.56 | 10.149.64.254 | TCP | 1514 | 1516 > 1522 [ACK] Seq=3312045024 Ack=4057770270 Win=63948 Len=1460 |
| 103288 | 04:14:42.910 | 10.149.64.254 | 10.149.64.56 | TCP | 54 | 1522 > 1516 [ACK] Seq=4057770522 Ack=3312046484 Win=1460 Len=0 |
| 103291 | 04:14:42.910 | 10.149.64.56 | 10.149.64.254 | TCP | 1514 | 1516 > 1522 [PSH, ACK] Seq=3312046484 Ack=4057770522 Win=63696 Len=1460 |
| 103298 | 04:14:42.912 | 10.149.64.254 | 10.149.64.56 | TCP | 96 | [TCP ZeroWindow] 1522 > 1516 [PSH, ACK] Seq=4057770522 Ack=3312047944 Win=0 Len=42 |
| 103316 | 04:14:42.913 | 10.149.64.254 | 10.149.64.56 | TCP | 54 | [TCP Window Update] 1522 > 1516 [ACK] Seq=4057770564 Ack=3312047944 Win=3760 Len=0 |
| 103318 | 04:14:42.913 | 10.149.64.254 | 10.149.64.56 | TCP | 54 | [TCP Window Update] 1522 > 1516 [ACK] Seq=4057770564 Ack=3312047944 Win=64240 Len=0 |
| 103321 | 04:14:42.913 | 10.149.64.56 | 10.149.64.254 | TCP | 1514 | 1516 > 1522 [ACK] Seq=3312047944 Ack=4057770564 Win=63654 Len=1460 |
| 103322 | 04:14:42.913 | 10.149.64.56 | 10.149.64.254 | TCP | 1514 | 1516 > 1522 [ACK] Seq=3312049404 Ack=4057770564 Win=63654 Len=1460 |
| 103324 | 04:14:42.913 | 10.149.64.254 | 10.149.64.56 | TCP | 541 | 1522 > 1516 [PSH, ACK] Seq=4057770564 Ack=3312050864 Win=61320 Len=487 |

- Note short duration (1ms) and quick recovery (window went from 0 to 63654 quickly)
- This is definitely a ‘hiccup’
- Usually not a concern unless they happen VERY frequently

# TCP Zero Windows - Example 2

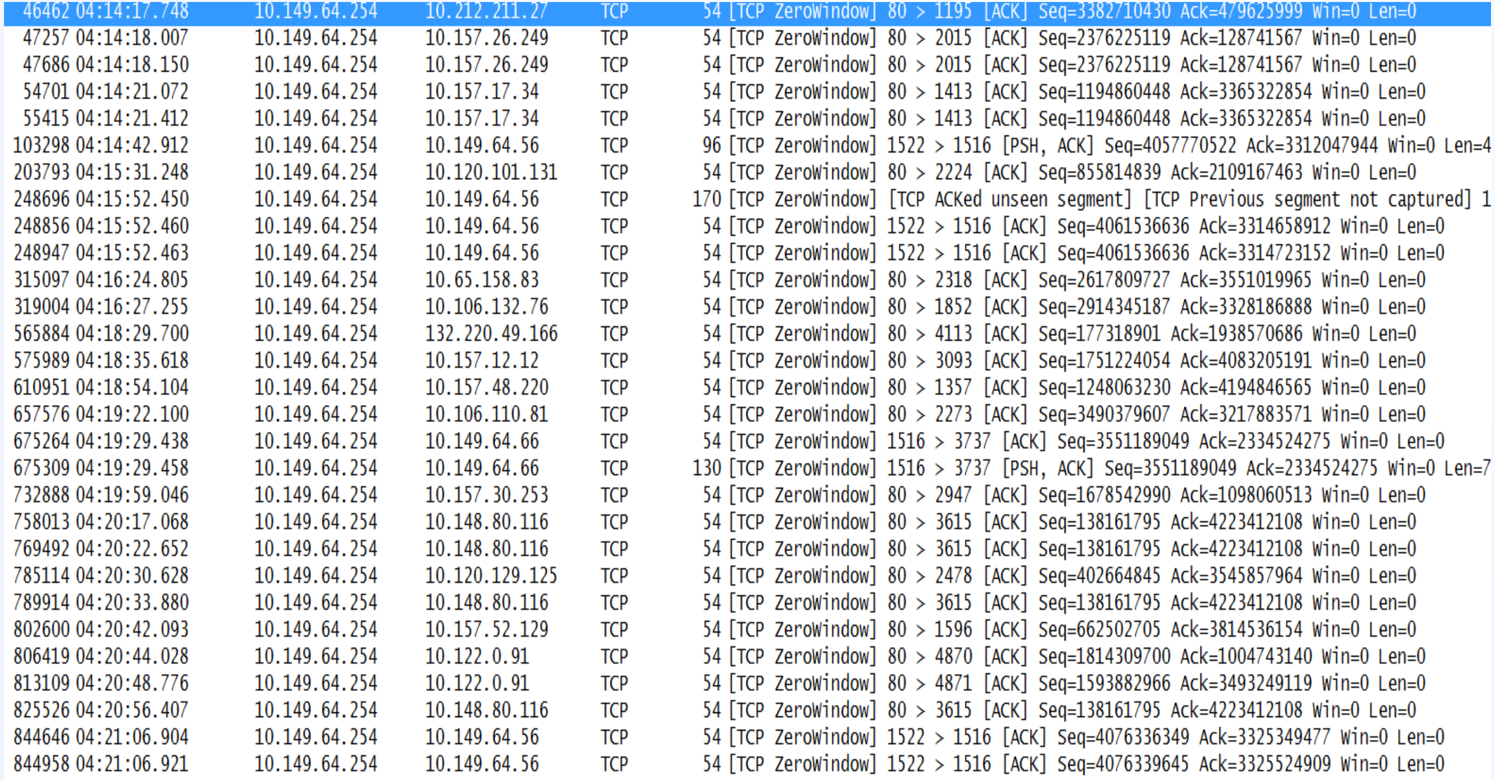


| No. | Time | Source | Destination | Protocol | Length | Info |
|---|---|---|---|---|---|---|
| 21742 | 04:22:56.941 | 10.103.6.51 | 10.149.64.254 | TCP | 83 | 3262 > 8081 [PSH, ACK] Seq=3825090269 Ack=1867219222 Win=65481 Len=29 |
| 21743 | 04:22:56.941 | 10.103.6.51 | 10.149.64.254 | TCP | 68 | 3262 > 8081 [PSH, ACK] Seq=3825090298 Ack=1867219222 Win=65481 Len=14 |
| 21744 | 04:22:56.941 | 10.149.64.254 | 10.103.6.51 | TCP | 54 | 8081 > 3262 [ACK] Seq=1867219222 Ack=3825090312 Win=376 Len=0 |
| 21745 | 04:23:01.954 | 10.103.6.51 | 10.149.64.254 | TCP | 430 | 3262 > 8081 [ACK] Seq=3825090312 Ack=1867219222 Win=65481 Len=376 |
| 21746 | 04:23:02.157 | 10.149.64.254 | 10.103.6.51 | TCP | 54 | [TCP ZeroWindow] 8081 > 3262 [ACK] Seq=1867219222 Ack=3825090688 Win=0 Len=0 |
| 21747 | 04:23:02.756 | 10.103.6.51 | 10.149.64.254 | TCP | 60 | [TCP ZeroWindowProbe] 3262 > 8081 [ACK] Seq=3825090688 Ack=1867219222 Win=65481 Len=1 |
| 21748 | 04:23:02.756 | 10.149.64.254 | 10.103.6.51 | TCP | 54 | [TCP ZeroWindowProbeAck] [TCP ZeroWindow] 8081 > 3262 [ACK] Seq=1867219222 Ack=3825090688 Win=0 Len=0 |
| 21749 | 04:23:03.963 | 10.103.6.51 | 10.149.64.254 | TCP | 60 | [TCP ZeroWindowProbe] 3262 > 8081 [ACK] Seq=3825090688 Ack=1867219222 Win=65481 Len=1 |
| 21750 | 04:23:03.963 | 10.149.64.254 | 10.103.6.51 | TCP | 54 | [TCP ZeroWindowProbeAck] [TCP ZeroWindow] 8081 > 3262 [ACK] Seq=1867219222 Ack=3825090688 Win=0 Len=0 |
| 21751 | 04:23:06.377 | 10.103.6.51 | 10.149.64.254 | TCP | 60 | [TCP ZeroWindowProbe] 3262 > 8081 [ACK] Seq=3825090688 Ack=1867219222 Win=65481 Len=1 |
| 21752 | 04:23:06.377 | 10.149.64.254 | 10.103.6.51 | TCP | 54 | [TCP ZeroWindowProbeAck] [TCP ZeroWindow] 8081 > 3262 [ACK] Seq=1867219222 Ack=3825090688 Win=0 Len=0 |
| 21753 | 04:23:08.461 | 10.149.64.254 | 10.103.6.51 | TCP | 54 | [TCP Window Update] 8081 > 3262 [ACK] Seq=1867219222 Ack=3825090688 Win=5165 Len=0 |
| 21754 | 04:23:08.461 | 10.149.64.254 | 10.103.6.51 | TCP | 54 | [TCP Dup ACK 21753#1] 8081 > 3262 [ACK] Seq=1867219222 Ack=3825090688 Win=5165 Len=0 |
| 21755 | 04:23:08.461 | 10.149.64.254 | 10.103.6.51 | TCP | 79 | 8081 > 3262 [PSH, ACK] Seq=1867219222 Ack=3825090688 Win=5165 Len=25 |
| 21756 | 04:23:08.508 | 10.103.6.51 | 10.149.64.254 | TCP | 1514 | 3262 > 8081 [PSH, ACK] Seq=3825090688 Ack=1867219222 Win=65481 Len=1460 |
| 21757 | 04:23:08.508 | 10.103.6.51 | 10.149.64.254 | TCP | 1514 | 3262 > 8081 [ACK] Seq=3825092148 Ack=1867219222 Win=65481 Len=1460 |

- Note duration (4s), slow recovery (window only went from 0 to 5165), and other side waiting to send data (Zero Window Probes)

# TCP Zero Windows – Example 3

#sf21veu



```
 46462 04:14:17.748   10.149.64.254   10.212.211.27    TCP    54 [TCP ZeroWindow] 80 > 1195 [ACK] Seq=3382710430 Ack=479625999 Win=0 Len=0
 47257 04:14:18.007   10.149.64.254   10.157.26.249    TCP    54 [TCP ZeroWindow] 80 > 2015 [ACK] Seq=2376225119 Ack=128741567 Win=0 Len=0
 47686 04:14:18.150   10.149.64.254   10.157.26.249    TCP    54 [TCP ZeroWindow] 80 > 2015 [ACK] Seq=2376225119 Ack=128741567 Win=0 Len=0
 54701 04:14:21.072   10.149.64.254   10.157.17.34     TCP    54 [TCP ZeroWindow] 80 > 1413 [ACK] Seq=1194860448 Ack=3365322854 Win=0 Len=0
 55415 04:14:21.412   10.149.64.254   10.157.17.34     TCP    54 [TCP ZeroWindow] 80 > 1413 [ACK] Seq=1194860448 Ack=3365322854 Win=0 Len=0
103298 04:14:42.912   10.149.64.254   10.149.64.56     TCP    96 [TCP ZeroWindow] 1522 > 1516 [PSH, ACK] Seq=4057770522 Ack=3312047944 Win=0 Len=4
203793 04:15:31.248   10.149.64.254   10.120.101.131   TCP    54 [TCP ZeroWindow] 80 > 2224 [ACK] Seq=855814839 Ack=2109167463 Win=0 Len=0
248696 04:15:52.450   10.149.64.254   10.149.64.56     TCP   170 [TCP ZeroWindow] [TCP ACKed unseen segment] [TCP Previous segment not captured] 1
248856 04:15:52.460   10.149.64.254   10.149.64.56     TCP    54 [TCP ZeroWindow] 1522 > 1516 [ACK] Seq=4061536636 Ack=3314658912 Win=0 Len=0
248947 04:15:52.463   10.149.64.254   10.149.64.56     TCP    54 [TCP ZeroWindow] 1522 > 1516 [ACK] Seq=4061536636 Ack=3314723152 Win=0 Len=0
315097 04:16:24.805   10.149.64.254   10.65.158.83     TCP    54 [TCP ZeroWindow] 80 > 2318 [ACK] Seq=2617809727 Ack=3551019965 Win=0 Len=0
319004 04:16:27.255   10.149.64.254   10.106.132.76    TCP    54 [TCP ZeroWindow] 80 > 1852 [ACK] Seq=2914345187 Ack=3328186888 Win=0 Len=0
565884 04:18:29.700   10.149.64.254   132.220.49.166   TCP    54 [TCP ZeroWindow] 80 > 4113 [ACK] Seq=177318901 Ack=1938570686 Win=0 Len=0
575989 04:18:35.618   10.149.64.254   10.157.12.12     TCP    54 [TCP ZeroWindow] 80 > 3093 [ACK] Seq=1751224054 Ack=4083205191 Win=0 Len=0
610951 04:18:54.104   10.149.64.254   10.157.48.220    TCP    54 [TCP ZeroWindow] 80 > 1357 [ACK] Seq=1248063230 Ack=4194846565 Win=0 Len=0
657576 04:19:22.100   10.149.64.254   10.106.110.81    TCP    54 [TCP ZeroWindow] 80 > 2273 [ACK] Seq=3490379607 Ack=3217883571 Win=0 Len=0
675264 04:19:29.438   10.149.64.254   10.149.64.66     TCP    54 [TCP ZeroWindow] 1516 > 3737 [ACK] Seq=3551189049 Ack=2334524275 Win=0 Len=0
675309 04:19:29.458   10.149.64.254   10.149.64.66     TCP   130 [TCP ZeroWindow] 1516 > 3737 [PSH, ACK] Seq=3551189049 Ack=2334524275 Win=0 Len=7
732888 04:19:59.046   10.149.64.254   10.157.30.253    TCP    54 [TCP ZeroWindow] 80 > 2947 [ACK] Seq=1678542990 Ack=1098060513 Win=0 Len=0
758013 04:20:17.068   10.149.64.254   10.148.80.116    TCP    54 [TCP ZeroWindow] 80 > 3615 [ACK] Seq=138161795 Ack=4223412108 Win=0 Len=0
769492 04:20:22.652   10.149.64.254   10.148.80.116    TCP    54 [TCP ZeroWindow] 80 > 3615 [ACK] Seq=138161795 Ack=4223412108 Win=0 Len=0
785114 04:20:30.628   10.149.64.254   10.120.129.125   TCP    54 [TCP ZeroWindow] 80 > 2478 [ACK] Seq=402664845 Ack=3545857964 Win=0 Len=0
789914 04:20:33.880   10.149.64.254   10.148.80.116    TCP    54 [TCP ZeroWindow] 80 > 3615 [ACK] Seq=138161795 Ack=4223412108 Win=0 Len=0
802600 04:20:42.093   10.149.64.254   10.157.52.129    TCP    54 [TCP ZeroWindow] 80 > 1596 [ACK] Seq=662502705 Ack=3814536154 Win=0 Len=0
806419 04:20:44.028   10.149.64.254   10.122.0.91      TCP    54 [TCP ZeroWindow] 80 > 4870 [ACK] Seq=1814309700 Ack=1004743140 Win=0 Len=0
813109 04:20:48.776   10.149.64.254   10.122.0.91      TCP    54 [TCP ZeroWindow] 80 > 4871 [ACK] Seq=1593882966 Ack=3493249119 Win=0 Len=0
825526 04:20:56.407   10.149.64.254   10.148.80.116    TCP    54 [TCP ZeroWindow] 80 > 3615 [ACK] Seq=138161795 Ack=4223412108 Win=0 Len=0
844646 04:21:06.904   10.149.64.254   10.149.64.56     TCP    54 [TCP ZeroWindow] 1522 > 1516 [ACK] Seq=4076336349 Ack=3325349477 Win=0 Len=0
844958 04:21:06.921   10.149.64.254   10.149.64.56     TCP    54 [TCP ZeroWindow] 1522 > 1516 [ACK] Seq=4076339645 Ack=3325524909 Win=0 Len=0
```

- Long time span (4:14:17 through 4:21:06)
- Multiple protocols affected
- Multiple conversations affected concurrently
- Definite indicator of systemwide stress

# Thanks for being here!

Questions

and

Answers

Twitter: @wesmorgan1