SharkFest'19 US

TLS1.3, DoH, DNS over TLS, QUIC, IPv6 PDM and more …

Nalini Elkins

Inside Products, Inc.
www.insidethestack.com

Session Description

Many new protocols are being worked on at the IETF. Some are RFCs already. Others will soon gain that status. These protocols include: TLS1.3, DNS over HTTPs, DNS over TLS and QUIC. A fundamental premise that all of these protocols share is that metadata may be misused. So, more and more of the packet is being encrypted. How will this impact diagnostics and troubleshooting? If many of the protocol headers themselves are encrypted, how will we get performance information? One new RFC (RFC8250) for IPv6 Performance Diagnostics and Metrics tries to give us back some of the information we need. This session will discuss these new protocols and show packet flows for each.

About me?

- Product developer (including OEM by IBM, Boole & Babbage)
- Author: RFC8250: IPv6 Performance and Diagnostic Metrics (PDM) and others
- Doing network design / diagnostics for 25+ years
- Member in good standing of TraceRoute fan club (also WireShark!)

Agenda

- Background on “tussle”
- TLS1.3
- DoH
- DNS over TLS
- QUIC (gQuic)(HTTP/3)
- PDM
- Surprise bonus! (Simulated quantum network)

Why?

- Let’s start with something we know.
- TLS1.2

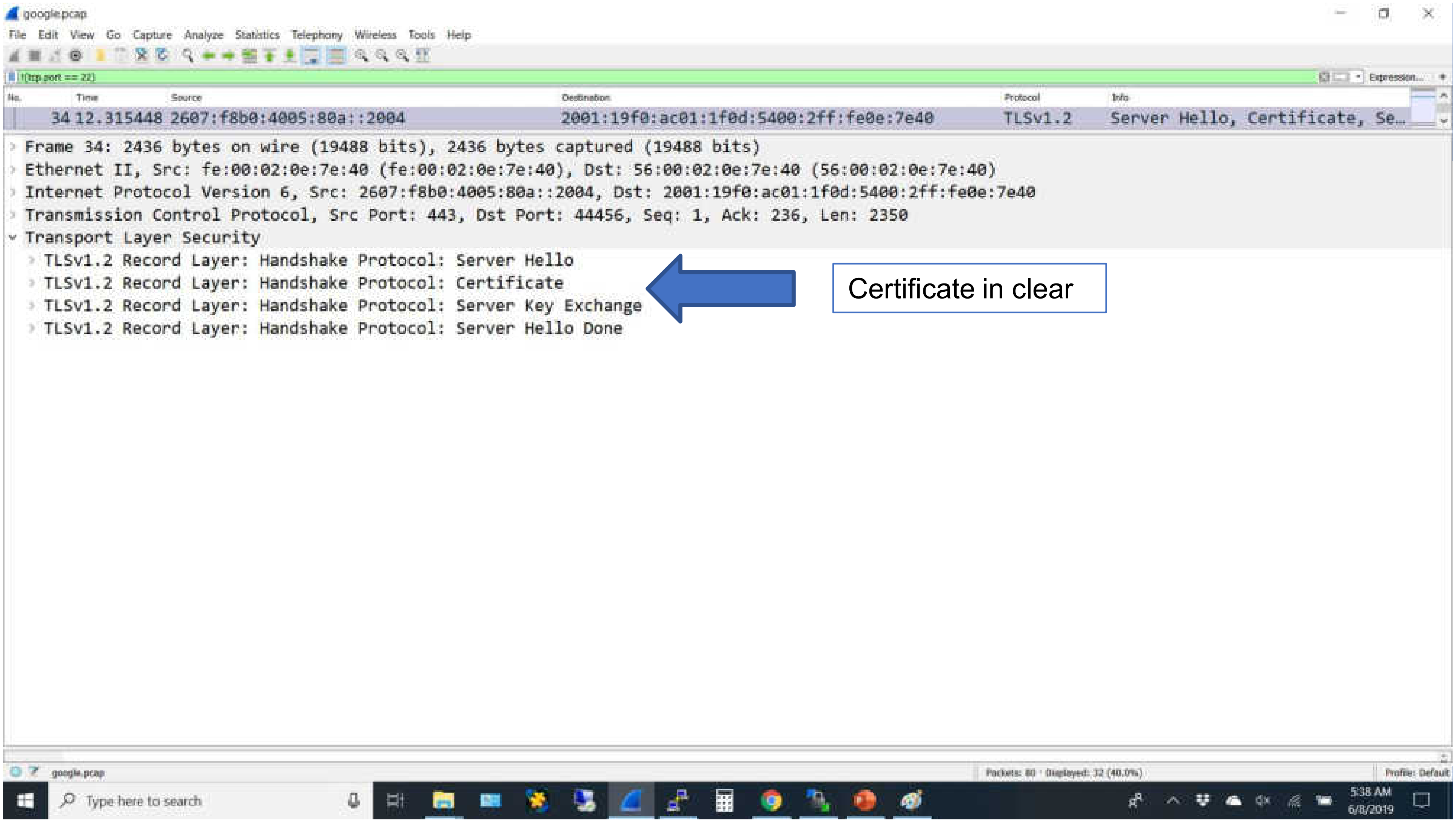
TLS1.2 to Google

google.pcap

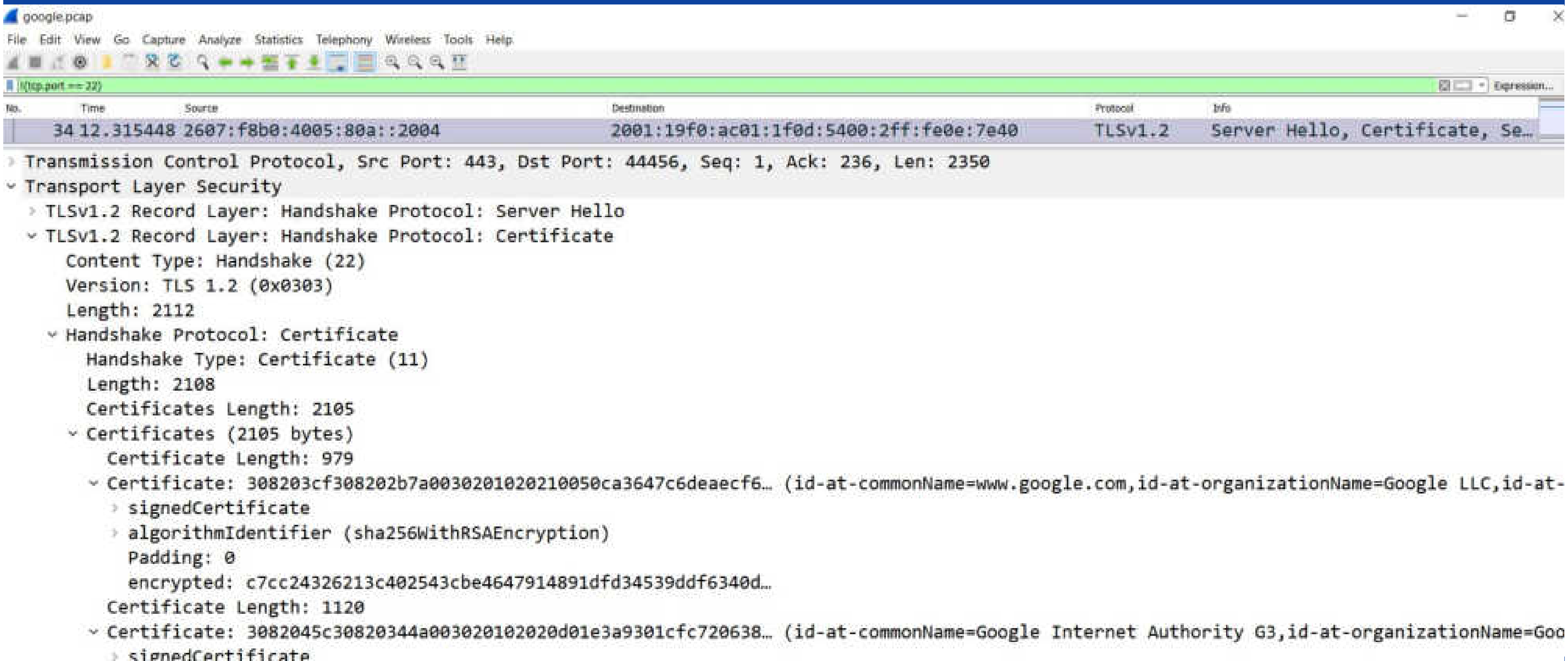
File Edit View Go Capture Analyze Statistics Telephony Wireless Tools Help

!(tcp.port == 22)

No.	Time	Source	Destination	Protocol	Info
29	12.293520	2001:19f0:ac01:1f0d:5400:2ff:fe0e:7e40	2607:f8b0:4005:80a::2004	TCP	44456 → 443 [SYN] Seq=0 Win=6…
30	12.295273	2607:f8b0:4005:80a::2004	2001:19f0:ac01:1f0d:5400:2ff:fe0e:7e40	TCP	443 → 44456 [SYN, ACK] Seq=0 …
31	12.295322	2001:19f0:ac01:1f0d:5400:2ff:fe0e:7e40	2607:f8b0:4005:80a::2004	TCP	44456 → 443 [ACK] Seq=1 Ack=1…
32	12.296353	2001:19f0:ac01:1f0d:5400:2ff:fe0e:7e40	2607:f8b0:4005:80a::2004	TLSv1.2	Client Hello
33	12.297757	2607:f8b0:4005:80a::2004	2001:19f0:ac01:1f0d:5400:2ff:fe0e:7e40	TCP	443 → 44456 [ACK] Seq=1 Ack=2…
34	12.315448	2607:f8b0:4005:80a::2004	2001:19f0:ac01:1f0d:5400:2ff:fe0e:7e40	TLSv1.2	Server Hello, Certificate, Se…
35	12.315478	2001:19f0:ac01:1f0d:5400:2ff:fe0e:7e40	2607:f8b0:4005:80a::2004	TCP	44456 → 443 [ACK] Seq=236 Ack…
36	12.317405	2001:19f0:ac01:1f0d:5400:2ff:fe0e:7e40	2607:f8b0:4005:80a::2004	TLSv1.2	Client Key Exchange, Change C…
37	12.318908	2607:f8b0:4005:80a::2004	2001:19f0:ac01:1f0d:5400:2ff:fe0e:7e40	TLSv1.2	Change Cipher Spec, Encrypted…
38	12.319201	2001:19f0:ac01:1f0d:5400:2ff:fe0e:7e40	2607:f8b0:4005:80a::2004	TLSv1.2	Application Data
39	12.319234	2001:19f0:ac01:1f0d:5400:2ff:fe0e:7e40	2607:f8b0:4005:80a::2004	TLSv1.2	Application Data
40	12.319256	2001:19f0:ac01:1f0d:5400:2ff:fe0e:7e40	2607:f8b0:4005:80a::2004	TLSv1.2	Application Data
41	12.319343	2001:19f0:ac01:1f0d:5400:2ff:fe0e:7e40	2607:f8b0:4005:80a::2004	TLSv1.2	Application Data
42	12.320608	2607:f8b0:4005:80a::2004	2001:19f0:ac01:1f0d:5400:2ff:fe0e:7e40	TLSv1.2	Application Data
43	12.320632	2607:f8b0:4005:80a::2004	2001:19f0:ac01:1f0d:5400:2ff:fe0e:7e40	TLSv1.2	Application Data
44	12.320689	2001:19f0:ac01:1f0d:5400:2ff:fe0e:7e40	2607:f8b0:4005:80a::2004	TLSv1.2	Application Data
45	12.322161	2607:f8b0:4005:80a::2004	2001:19f0:ac01:1f0d:5400:2ff:fe0e:7e40	TCP	443 → 44456 [ACK] Seq=2485 Ac…
47	12.375322	2607:f8b0:4005:80a::2004	2001:19f0:ac01:1f0d:5400:2ff:fe0e:7e40	TLSv1.2	Application Data
48	12.375351	2607:f8b0:4005:80a::2004	2001:19f0:ac01:1f0d:5400:2ff:fe0e:7e40	TLSv1.2	Application Data
49	12.375359	2607:f8b0:4005:80a::2004	2001:19f0:ac01:1f0d:5400:2ff:fe0e:7e40	TLSv1.2	Application Data, Application…
50	12.375413	2607:f8b0:4005:80a::2004	2001:19f0:ac01:1f0d:5400:2ff:fe0e:7e40	TLSv1.2	Application Data, Application…
51	12.375424	2607:f8b0:4005:80a::2004	2001:19f0:ac01:1f0d:5400:2ff:fe0e:7e40	TLSv1.2	Application Data


google.pcap
File Edit View Go Capture Analyze Statistics Telephony Wireless Tools Help
!(tcp.port == 22)
No. Time Source Destination Protocol Info
34 12.315448 2607:f8b0:4005:80a::2004 2001:19f0:ac01:1f0d:5400:2ff:fe0e:7e40 TLSv1.2 Server Hello, Certificate, Se…
Frame 34: 2436 bytes on wire (19488 bits), 2436 bytes captured (19488 bits)
Ethernet II, Src: fe:00:02:0e:7e:40 (fe:00:02:0e:7e:40), Dst: 56:00:02:0e:7e:40 (56:00:02:0e:7e:40)
Internet Protocol Version 6, Src: 2607:f8b0:4005:80a::2004, Dst: 2001:19f0:ac01:1f0d:5400:2ff:fe0e:7e40
Transmission Control Protocol, Src Port: 443, Dst Port: 44456, Seq: 1, Ack: 236, Len: 2350
Transport Layer Security
TLSv1.2 Record Layer: Handshake Protocol: Server Hello
TLSv1.2 Record Layer: Handshake Protocol: Certificate
TLSv1.2 Record Layer: Handshake Protocol: Server Key Exchange
TLSv1.2 Record Layer: Handshake Protocol: Server Hello Done
Certificate in clear
google.pcap
Type here to search
5:38 AM
6/8/2019

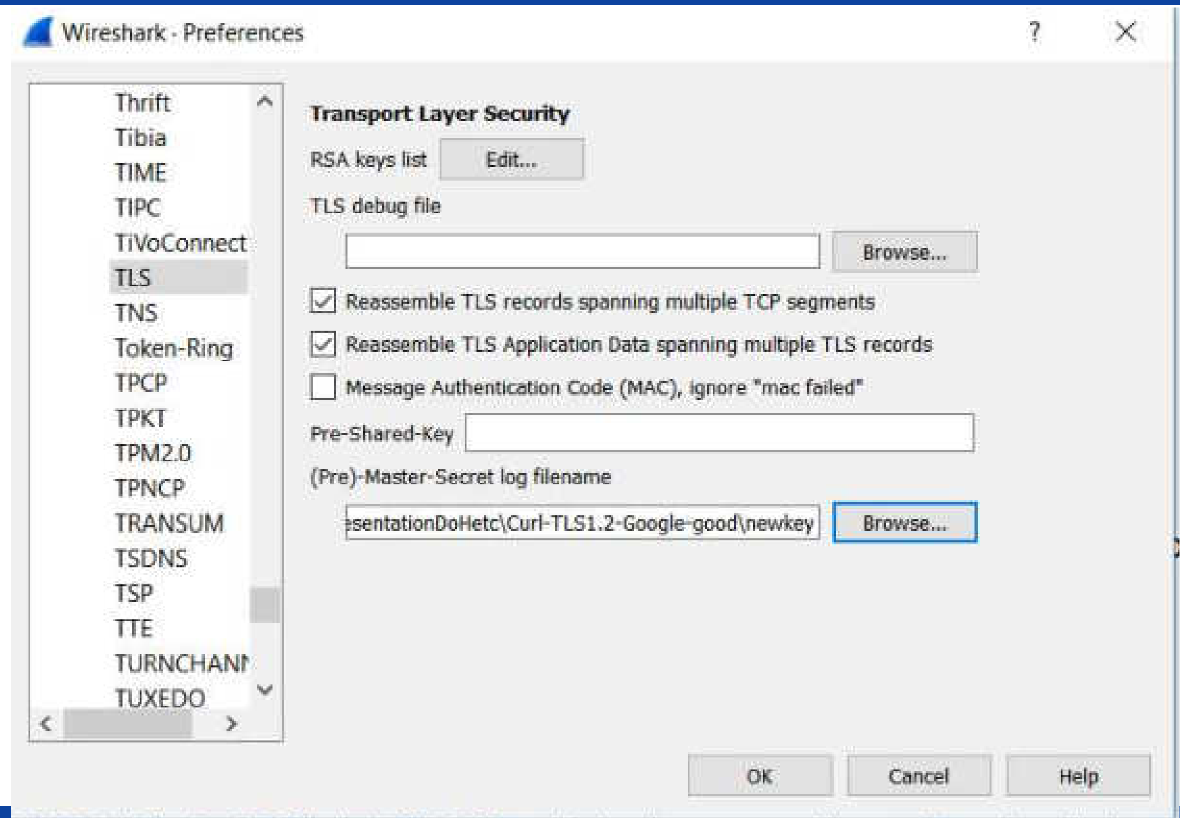
Cert. Encrypted TLS1.2



Let’s Decrypt

Add SSLKEYLOGFILE

CLIENT_RANDOM
03d574c74b3c1a36d37637c6c2779e3e
bd785bb6b5eb76c4546cdfe7e35e2c4c
423e69b3cc63cd433f0dfe0b6df6a4c113
47e5bf3a0783a4e6727a0a26786a53a0
7541b2566c96242486d498b0bfc64c



TLS1.2 Decrypted

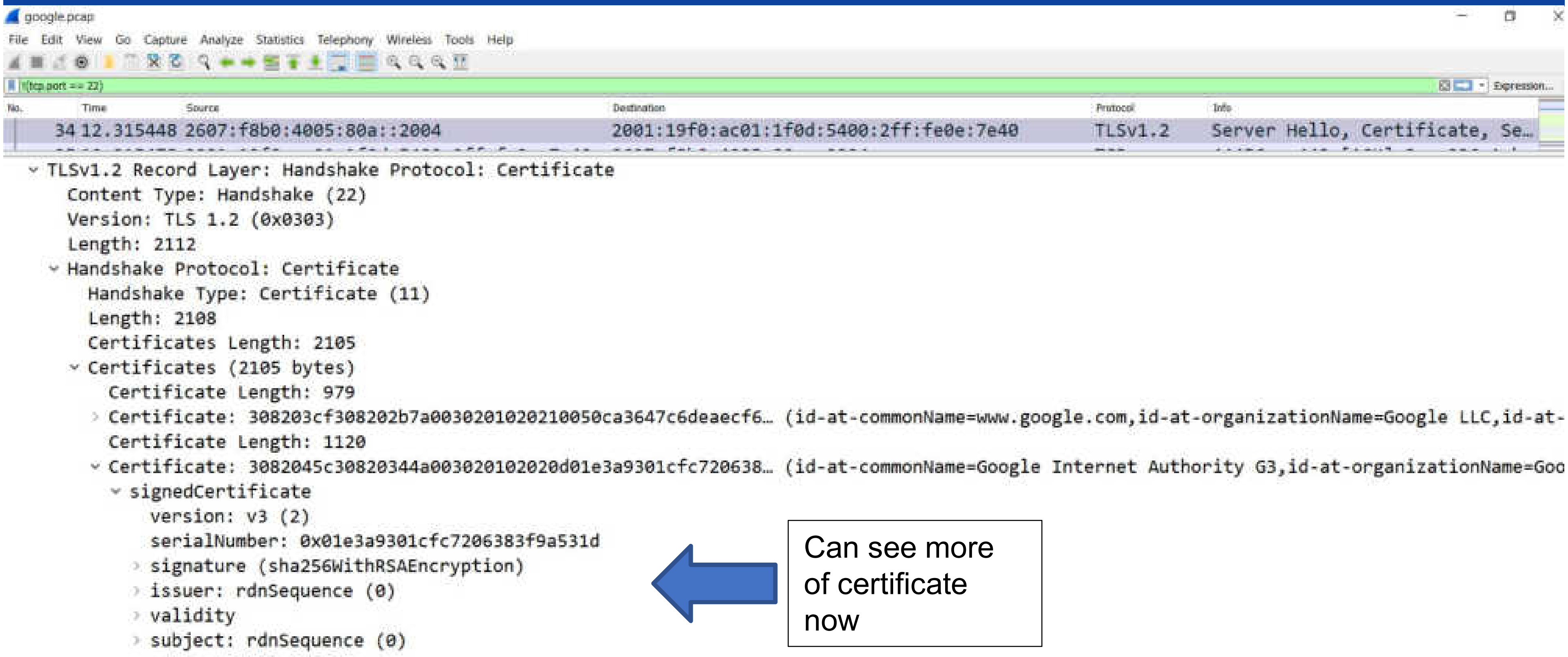
google.pcap

File Edit View Go Capture Analyze Statistics Telephony Wireless Tools Help

!(tcp.port == 22)

No.	Time	Source	Destination	Protocol	Info
32	12.296353	2001:19f0:ac01:1f0d:5400:2ff:fe0e:7e40	2607:f8b0:4005:80a::2004	TLSv1.2	Client Hello
33	12.297757	2607:f8b0:4005:80a::2004	2001:19f0:ac01:1f0d:5400:2ff:fe0e:7e40	TCP	443 → 44456 [ACK] Seq=1 Ack=2…
34	12.315448	2607:f8b0:4005:80a::2004	2001:19f0:ac01:1f0d:5400:2ff:fe0e:7e40	TLSv1.2	Server Hello, Certificate, Se…
35	12.315478	2001:19f0:ac01:1f0d:5400:2ff:fe0e:7e40	2607:f8b0:4005:80a::2004	TCP	44456 → 443 [ACK] Seq=236 Ack…
36	12.317405	2001:19f0:ac01:1f0d:5400:2ff:fe0e:7e40	2607:f8b0:4005:80a::2004	TLSv1.2	Client Key Exchange, Change C…
37	12.318908	2607:f8b0:4005:80a::2004	2001:19f0:ac01:1f0d:5400:2ff:fe0e:7e40	TLSv1.2	Change Cipher Spec, Finished
38	12.319201	2001:19f0:ac01:1f0d:5400:2ff:fe0e:7e40	2607:f8b0:4005:80a::2004	HTTP2	Magic
39	12.319234	2001:19f0:ac01:1f0d:5400:2ff:fe0e:7e40	2607:f8b0:4005:80a::2004	HTTP2	SETTINGS[0]
40	12.319256	2001:19f0:ac01:1f0d:5400:2ff:fe0e:7e40	2607:f8b0:4005:80a::2004	HTTP2	WINDOW_UPDATE[0]
41	12.319343	2001:19f0:ac01:1f0d:5400:2ff:fe0e:7e40	2607:f8b0:4005:80a::2004	HTTP2	HEADERS[1]: GET /
42	12.320608	2607:f8b0:4005:80a::2004	2001:19f0:ac01:1f0d:5400:2ff:fe0e:7e40	HTTP2	SETTINGS[0], WINDOW_UPDATE[0]
43	12.320632	2607:f8b0:4005:80a::2004	2001:19f0:ac01:1f0d:5400:2ff:fe0e:7e40	HTTP2	SETTINGS[0]
44	12.320689	2001:19f0:ac01:1f0d:5400:2ff:fe0e:7e40	2607:f8b0:4005:80a::2004	HTTP2	SETTINGS[0]
45	12.322161	2607:f8b0:4005:80a::2004	2001:19f0:ac01:1f0d:5400:2ff:fe0e:7e40	TCP	443 → 44456 [ACK] Seq=2485 Ac…
47	12.375322	2607:f8b0:4005:80a::2004	2001:19f0:ac01:1f0d:5400:2ff:fe0e:7e40	HTTP2	HEADERS[1]: 200 OK
48	12.375351	2607:f8b0:4005:80a::2004	2001:19f0:ac01:1f0d:5400:2ff:fe0e:7e40	TLSv1.2	[TLS segment of a reassembled…
49	12.375359	2607:f8b0:4005:80a::2004	2001:19f0:ac01:1f0d:5400:2ff:fe0e:7e40	TLSv1.2	[TLS segment of a reassembled…
50	12.375413	2607:f8b0:4005:80a::2004	2001:19f0:ac01:1f0d:5400:2ff:fe0e:7e40	TLSv1.2	[TLS segment of a reassembled…
51	12.375424	2607:f8b0:4005:80a::2004	2001:19f0:ac01:1f0d:5400:2ff:fe0e:7e40	HTTP2	DATA[1] (text/html)
52	12.375473	2001:19f0:ac01:1f0d:5400:2ff:fe0e:7e40	2607:f8b0:4005:80a::2004	TCP	44456 → 443 [ACK] Seq=539 Ack…
53	12.375591	2607:f8b0:4005:80a::2004	2001:19f0:ac01:1f0d:5400:2ff:fe0e:7e40	HTTP2	PING[0]
57	12.376751	2001:19f0:ac01:1f0d:5400:2ff:fe0e:7e40	2607:f8b0:4005:80a::2004	HTTP2	PING[0]

Decrypted Cert



Tussle

- Privacy of metadata
- Endpoints (applications) vs ISPs
- Enterprise diagnostics (packet decryption)

Gigamon

Solutions Products Support &

SSL Decryption

Key Features

- First in the Industry to Integrate SSL Decryption into a Unified Visibility Fabric Architecture
  - Decrypt traffic from anywhere within the Visibility Fabric and send to any connected tools
  - Flow Mapping directs any user-defined flows, not just those on port 443, for decryption
- Extensible, High-Throughput Solution
  - GigaVUE-HD4/8: 4M sessions, 5 Mpps per second per GigaSMART blade
  - GigaVUE-HC2: 2M sessions, 2.5 Mpps per second per GigaSMART module
  - GigaVUE-HB1: 500k sessions, 0.6 Mpps per second
- SSLv3, TLS 1.0, 1.1 and 1.2 Support
  - Public Key: RSA
  - Symmetric key algorithms: AES, 3DES, DES, RC4, CAMELLIA, SEED, IDEA
  - Hashing algorithms: MD5, SHA1, SHA2
  - Supported applications: HTTPS, FTPS and SMTP, IMAP, POP3 with StartTLS
  - Supported key sizes: 128, 256, 512, 1024, 2048, and 4096
- SSL Decryption Statistics
  - Idle sessions and reusable keys
  - Session-level Stats: packets, discards, errored packets, resumptions
- Secure Storage of Private Keys
  - Encryption with independent password
  - Restricted key access based on role-based access controls

- Sample from Gigamon SSL Decryption feature
- Notice the “RSA”

TLS1.3 to Google

No.	Time	Source	Destination	Protocol	Info
204	19.143872	2001:19f0:ac01:1f0d:5400:2ff:fe0e:7e40	2607:f8b0:4005:80a::2004	TCP	44458 → 443 [SYN] Seq=0 Win=6…
207	19.146111	2607:f8b0:4005:80a::2004	2001:19f0:ac01:1f0d:5400:2ff:fe0e:7e40	TCP	443 → 44458 [SYN, ACK] Seq=0 …
208	19.146154	2001:19f0:ac01:1f0d:5400:2ff:fe0e:7e40	2607:f8b0:4005:80a::2004	TCP	44458 → 443 [ACK] Seq=1 Ack=1…
209	19.147632	2001:19f0:ac01:1f0d:5400:2ff:fe0e:7e40	2607:f8b0:4005:80a::2004	TLSv1.3	Client Hello
210	19.149440	2607:f8b0:4005:80a::2004	2001:19f0:ac01:1f0d:5400:2ff:fe0e:7e40	TCP	443 → 44458 [ACK] Seq=1 Ack=5…
211	19.167348	2607:f8b0:4005:80a::2004	2001:19f0:ac01:1f0d:5400:2ff:fe0e:7e40	TLSv1.3	Server Hello, Change Cipher S…
212	19.167375	2001:19f0:ac01:1f0d:5400:2ff:fe0e:7e40	2607:f8b0:4005:80a::2004	TCP	44458 → 443 [ACK] Seq=518 Ack…
213	19.169274	2001:19f0:ac01:1f0d:5400:2ff:fe0e:7e40	2607:f8b0:4005:80a::2004	TLSv1.3	Change Cipher Spec, Applicati…
214	19.169462	2001:19f0:ac01:1f0d:5400:2ff:fe0e:7e40	2607:f8b0:4005:80a::2004	TLSv1.3	Application Data
215	19.169484	2001:19f0:ac01:1f0d:5400:2ff:fe0e:7e40	2607:f8b0:4005:80a::2004	TLSv1.3	Application Data
216	19.169496	2001:19f0:ac01:1f0d:5400:2ff:fe0e:7e40	2607:f8b0:4005:80a::2004	TLSv1.3	Application Data
217	19.169543	2001:19f0:ac01:1f0d:5400:2ff:fe0e:7e40	2607:f8b0:4005:80a::2004	TLSv1.3	Application Data
218	19.171517	2607:f8b0:4005:80a::2004	2001:19f0:ac01:1f0d:5400:2ff:fe0e:7e40	TLSv1.3	Application Data, Application…
219	19.171703	2001:19f0:ac01:1f0d:5400:2ff:fe0e:7e40	2607:f8b0:4005:80a::2004	TLSv1.3	Application Data
220	19.171926	2607:f8b0:4005:80a::2004	2001:19f0:ac01:1f0d:5400:2ff:fe0e:7e40	TLSv1.3	Application Data
222	19.212795	2001:19f0:ac01:1f0d:5400:2ff:fe0e:7e40	2607:f8b0:4005:80a::2004	TCP	44458 → 443 [ACK] Seq=821 Ack…
223	19.212927	2607:f8b0:4005:80a::2004	2001:19f0:ac01:1f0d:5400:2ff:fe0e:7e40	TCP	443 → 44458 [ACK] Seq=3047 Ac…
224	19.217780	2607:f8b0:4005:80a::2004	2001:19f0:ac01:1f0d:5400:2ff:fe0e:7e40	TLSv1.3	Application Data
225	19.217798	2001:19f0:ac01:1f0d:5400:2ff:fe0e:7e40	2607:f8b0:4005:80a::2004	TCP	44458 → 443 [ACK] Seq=821 Ack…
226	19.217833	2607:f8b0:4005:80a::2004	2001:19f0:ac01:1f0d:5400:2ff:fe0e:7e40	TLSv1.3	Application Data
227	19.217845	2001:19f0:ac01:1f0d:5400:2ff:fe0e:7e40	2607:f8b0:4005:80a::2004	TCP	44458 → 443 [ACK] Seq=821 Ack…
228	19.217849	2607:f8b0:4005:80a::2004	2001:19f0:ac01:1f0d:5400:2ff:fe0e:7e40	TLSv1.3	Application Data, Application…

Differences from TLS1.2

- Notice that handshake is different
- Much more encrypted
- Can only see Client Hello and Server Hello

SSLKEYLOGFILE

Used same environment variable in Linux to capture.

```
SERVER_HANDSHAKE_TRAFFIC_SECRET 49a63b08e0810d4abb2ee926c5a7ba4619c97d31a374f11e8a99b680c70336b8
e6bf7bd3f8f8ce2bb2f5b54989b519e0eb7e536e01164cbf542ea52d9b35fd01a873a68df8cf76b241f0c9f0759ac635

EXPORTER_SECRET 49a63b08e0810d4abb2ee926c5a7ba4619c97d31a374f11e8a99b680c70336b8
e05f874b55ea0c3bab17cf8cad3fc2f63245e577235318d1fb99686a40d29edfe5a657919f7f9e886bdb119a464ad8b7

SERVER_TRAFFIC_SECRET_0 49a63b08e0810d4abb2ee926c5a7ba4619c97d31a374f11e8a99b680c70336b8
949ccb80ec9e9be9559010c00fd895992d988d8a07e2ae29b1925dff6cdb0036490c554792a7992823ff2615abffb0e7

CLIENT_HANDSHAKE_TRAFFIC_SECRET 49a63b08e0810d4abb2ee926c5a7ba4619c97d31a374f11e8a99b680c70336b8
3a5da2e1eeeafa0785c351368d0eceebbe451b39b5036c1de72db34f43f1106f318b12ef665d5462a980cb6349b2183b

CLIENT_TRAFFIC_SECRET_0 49a63b08e0810d4abb2ee926c5a7ba4619c97d31a374f11e8a99b680c70336b8
3c6bcfc90dbcd965e62b8eeafaeb3fe9ec59047f98edd3b745f5dc89b9fe4ab8db73032e66d565137df8592cd8b03eb7
```

TLS1.3 Decrypted

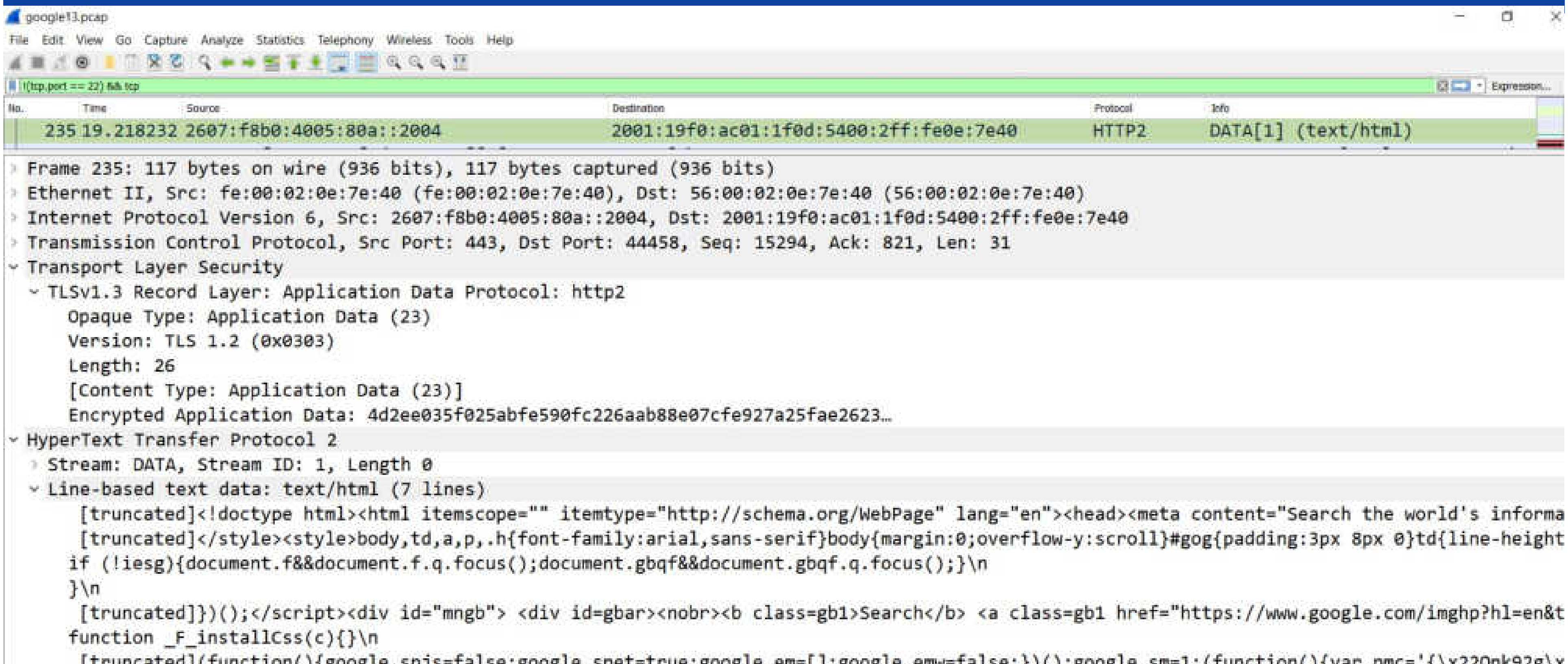
google13.pcap

File Edit View Go Capture Analyze Statistics Telephony Wireless Tools Help

!(tcp.port == 22) && tcp

No.	Time	Source	Destination	Protocol	Info
204	19.143872	2001:19f0:ac01:1f0d:5400:2ff:fe0e:7e40	2607:f8b0:4005:80a::2004	TCP	44458 → 443 [SYN] Seq=0 Win=6…
207	19.146111	2607:f8b0:4005:80a::2004	2001:19f0:ac01:1f0d:5400:2ff:fe0e:7e40	TCP	443 → 44458 [SYN, ACK] Seq=0 …
208	19.146154	2001:19f0:ac01:1f0d:5400:2ff:fe0e:7e40	2607:f8b0:4005:80a::2004	TCP	44458 → 443 [ACK] Seq=1 Ack=1…
209	19.147632	2001:19f0:ac01:1f0d:5400:2ff:fe0e:7e40	2607:f8b0:4005:80a::2004	TLSv1.3	Client Hello
210	19.149440	2607:f8b0:4005:80a::2004	2001:19f0:ac01:1f0d:5400:2ff:fe0e:7e40	TCP	443 → 44458 [ACK] Seq=1 Ack=5…
211	19.167348	2607:f8b0:4005:80a::2004	2001:19f0:ac01:1f0d:5400:2ff:fe0e:7e40	TLSv1.3	Server Hello, Change Cipher S…
212	19.167375	2001:19f0:ac01:1f0d:5400:2ff:fe0e:7e40	2607:f8b0:4005:80a::2004	TCP	44458 → 443 [ACK] Seq=518 Ack…
213	19.169274	2001:19f0:ac01:1f0d:5400:2ff:fe0e:7e40	2607:f8b0:4005:80a::2004	TLSv1.3	Change Cipher Spec, Finished
214	19.169462	2001:19f0:ac01:1f0d:5400:2ff:fe0e:7e40	2607:f8b0:4005:80a::2004	HTTP2	Magic
215	19.169484	2001:19f0:ac01:1f0d:5400:2ff:fe0e:7e40	2607:f8b0:4005:80a::2004	HTTP2	SETTINGS[0]
216	19.169496	2001:19f0:ac01:1f0d:5400:2ff:fe0e:7e40	2607:f8b0:4005:80a::2004	HTTP2	WINDOW_UPDATE[0]
217	19.169543	2001:19f0:ac01:1f0d:5400:2ff:fe0e:7e40	2607:f8b0:4005:80a::2004	HTTP2	HEADERS[1]: GET /
218	19.171517	2607:f8b0:4005:80a::2004	2001:19f0:ac01:1f0d:5400:2ff:fe0e:7e40	HTTP2	SETTINGS[0], WINDOW_UPDATE[0]
219	19.171703	2001:19f0:ac01:1f0d:5400:2ff:fe0e:7e40	2607:f8b0:4005:80a::2004	HTTP2	SETTINGS[0]
220	19.171926	2607:f8b0:4005:80a::2004	2001:19f0:ac01:1f0d:5400:2ff:fe0e:7e40	HTTP2	SETTINGS[0]
222	19.212795	2001:19f0:ac01:1f0d:5400:2ff:fe0e:7e40	2607:f8b0:4005:80a::2004	TCP	44458 → 443 [ACK] Seq=821 Ack…
223	19.212927	2607:f8b0:4005:80a::2004	2001:19f0:ac01:1f0d:5400:2ff:fe0e:7e40	TCP	443 → 44458 [ACK] Seq=3047 Ac…
224	19.217780	2607:f8b0:4005:80a::2004	2001:19f0:ac01:1f0d:5400:2ff:fe0e:7e40	HTTP2	HEADERS[1]: 200 OK
225	19.217798	2001:19f0:ac01:1f0d:5400:2ff:fe0e:7e40	2607:f8b0:4005:80a::2004	TCP	44458 → 443 [ACK] Seq=821 Ack…
226	19.217833	2607:f8b0:4005:80a::2004	2001:19f0:ac01:1f0d:5400:2ff:fe0e:7e40	TLSv1.3	[TLS segment of a reassembled…
227	19.217845	2001:19f0:ac01:1f0d:5400:2ff:fe0e:7e40	2607:f8b0:4005:80a::2004	TCP	44458 → 443 [ACK] Seq=821 Ack…
228	19.217849	2607:f8b0:4005:80a::2004	2001:19f0:ac01:1f0d:5400:2ff:fe0e:7e40	TLSv1.3	[TLS segment of a reassembled…

Packet Data Decrypted



DoH Enterprise Issues

Conversation with Fortune 50 company architect telling him that browsers will have pointer to DoH resolvers.

- You mean that DNS could be resolved outside my enterprise?
- So whoever that is that resolves my DNS sees the pattern and frequency of what sites my company goes to?
- How do I change this?

DoH and House of Lords

Internet Encryption - *Question*

– in the House of Lords at 2:53 pm on 14th May 2019.

Baroness Thornton Shadow Spokesperson (Health) 2:53 pm, 14th May 2019

To ask Her Majesty's Government what assessment they have made of the deployment of the Internet Engineering Task Force's new "DNS over HTTPS" protocol and its implications for the blocking of content by internet service providers and the Internet Watch Foundation; and what steps they intend to take in response.

https://www.theyworkforyou.com/lords/?id=2019-05-14a.1492.3

DOH: How to do

- Curl –doh-url https://1.1.1.1 https://www.google.com

- (1.1.1.1 = cloudflare, can use any public DoH server)

DoH to 1.1.1.1

doh.pcap

File Edit View Go Capture Analyze Statistics Telephony Wireless Tools Help

!(tcp.port == 22) && tcp

No.	Time	Source	Destination	Protocol	Info
68	110.9397…	144.202.109.208	1.1.1.1	TCP	55700 → 443 [SYN] Seq=0 Win=64240 Len=0 MSS=1460 SACK_PERM=1 TSv…
84	110.9402…	1.1.1.1	144.202.109.2…	TCP	443 → 55700 [SYN, ACK] Seq=0 Ack=1 Win=29200 Len=0 MSS=1460 SACK…
85	110.9402…	144.202.109.208	1.1.1.1	TCP	55700 → 443 [ACK] Seq=1 Ack=1 Win=64256 Len=0
92	110.9405…	144.202.109.208	1.1.1.1	TCP	55702 → 443 [SYN] Seq=0 Win=64240 Len=0 MSS=1460 SACK_PERM=1 TSv…
99	110.9410…	1.1.1.1	144.202.109.208	TCP	443 → 55702 [SYN, ACK] Seq=0 Ack=1 Win=29200 Len=0 MSS=1460 SACK…
100	110.9410…	144.202.109.208	1.1.1.1	TCP	55702 → 443 [ACK] Seq=1 Ack=1 Win=64256 Len=0
115	110.9419…	144.202.109.208	1.1.1.1	TLSv1.3	Client Hello
123	110.9424…	1.1.1.1	144.202.109.208	TCP	443 → 55700 [ACK] Seq=1 Ack=518 Win=30720 Len=0
124	110.9425…	144.202.109.208	1.1.1.1	TLSv1.3	Client Hello
125	110.9430…	1.1.1.1	144.202.109.208	TCP	443 → 55702 [ACK] Seq=1 Ack=518 Win=30720 Len=0
126	110.9435…	1.1.1.1	144.202.109.208	TLSv1.3	Server Hello, Change Cipher Spec
127	110.9435…	144.202.109.208	1.1.1.1	TCP	55700 → 443 [ACK] Seq=518 Ack=1461 Win=64128 Len=0
128	110.9435…	1.1.1.1	144.202.109.208	TLSv1.3	Application Data
129	110.9435…	144.202.109.208	1.1.1.1	TCP	55700 → 443 [ACK] Seq=518 Ack=2746 Win=63360 Len=0
130	110.9458…	1.1.1.1	144.202.109.208	TLSv1.3	Server Hello, Change Cipher Spec
131	110.9458…	144.202.109.208	1.1.1.1	TCP	55702 → 443 [ACK] Seq=518 Ack=1461 Win=64128 Len=0
132	110.9458…	1.1.1.1	144.202.109.208	TLSv1.3	Application Data
133	110.9458…	144.202.109.208	1.1.1.1	TCP	55702 → 443 [ACK] Seq=518 Ack=2745 Win=63360 Len=0
134	110.9474…	144.202.109.208	1.1.1.1	TLSv1.3	Change Cipher Spec, Application Data
135	110.9480…	1.1.1.1	144.202.109.208	TLSv1.3	Application Data, Application Data
136	110.9481…	144.202.109.208	1.1.1.1	TLSv1.3	Application Data
137	110.9481…	144.202.109.208	1.1.1.1	TLSv1.3	Application Data

Notice TLS1.3

Could this be DNS query?

Decrypted

doh.pcap

File Edit View Go Capture Analyze Statistics Telephony Wireless Tools Help

!(tcp.port == 22) && tcp

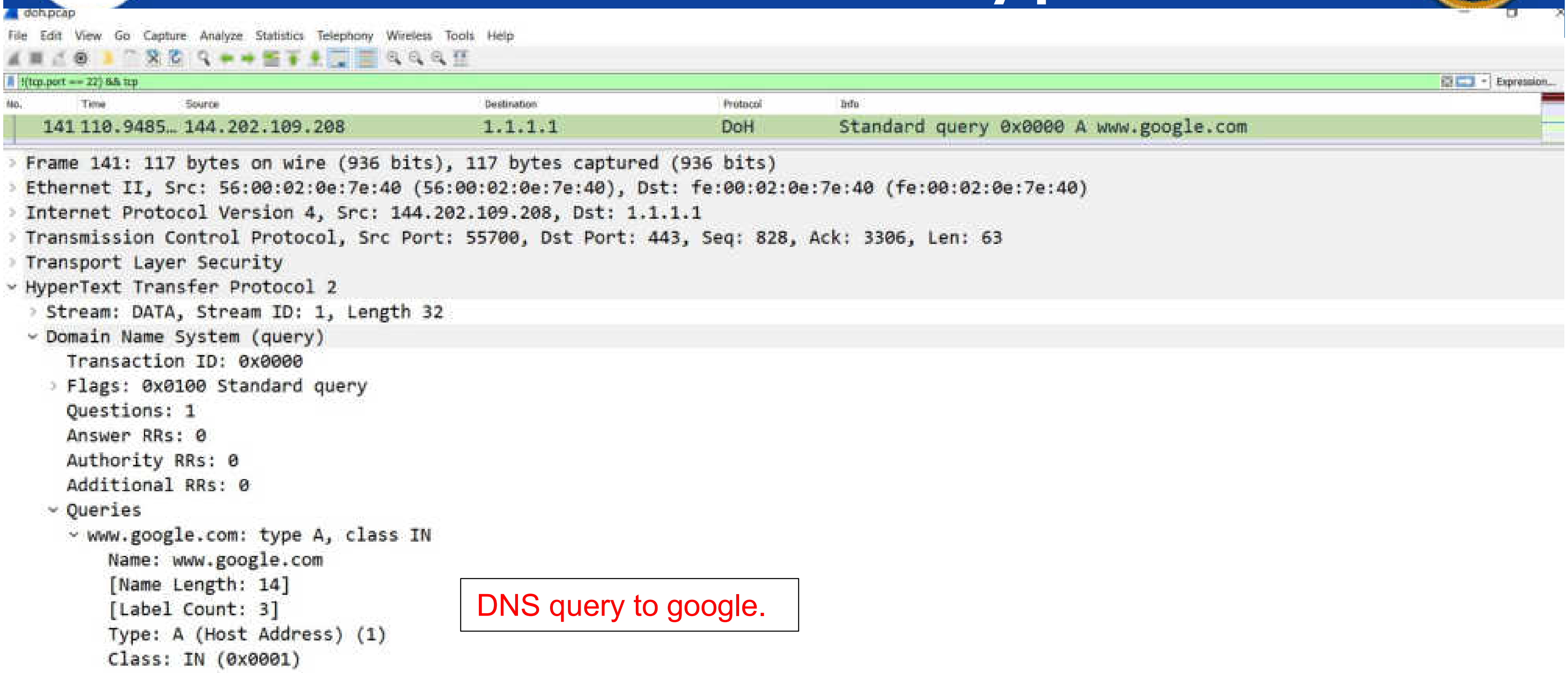
No.	Time	Source	Destination	Protocol	Info
124	110.9425…	144.202.109.208	1.1.1.1	TLSv1.3	Client Hello
125	110.9430…	1.1.1.1	144.202.109.208	TCP	443 → 55702 [ACK] Seq=1 Ack=518 Win=30720 Len=0
126	110.9435…	1.1.1.1	144.202.109.208	TLSv1.3	Server Hello, Change Cipher Spec
127	110.9435…	144.202.109.208	1.1.1.1	TCP	55700 → 443 [ACK] Seq=518 Ack=1461 Win=64128 Len=0
128	110.9435…	1.1.1.1	144.202.109.208	TLSv1.3	Encrypted Extensions, Certificate, Certificate Verify, Finished
129	110.9435…	144.202.109.208	1.1.1.1	TCP	55700 → 443 [ACK] Seq=518 Ack=2746 Win=63360 Len=0
130	110.9458…	1.1.1.1	144.202.109.208	TLSv1.3	Server Hello, Change Cipher Spec
131	110.9458…	144.202.109.208	1.1.1.1	TCP	55702 → 443 [ACK] Seq=518 Ack=1461 Win=64128 Len=0
132	110.9458…	1.1.1.1	144.202.109.208	TLSv1.3	Encrypted Extensions, Certificate, Certificate Verify, Finished
133	110.9458…	144.202.109.208	1.1.1.1	TCP	55702 → 443 [ACK] Seq=518 Ack=2745 Win=63360 Len=0
134	110.9474…	144.202.109.208	1.1.1.1	TLSv1.3	Change Cipher Spec, Finished
135	110.9480…	1.1.1.1	144.202.109.208	HTTP2	SETTINGS[0], WINDOW_UPDATE[0]
136	110.9481…	144.202.109.208	1.1.1.1	HTTP2	Magic
137	110.9481…	144.202.109.208	1.1.1.1	HTTP2	SETTINGS[0]
138	110.9481…	144.202.109.208	1.1.1.1	HTTP2	WINDOW_UPDATE[0]
139	110.9482…	144.202.109.208	1.1.1.1	HTTP2	HEADERS[1]: POST /
140	110.9485…	144.202.109.208	1.1.1.1	HTTP2	SETTINGS[0]
141	110.9485…	144.202.109.208	1.1.1.1	DoH	Standard query 0x0000 A www.google.com
142	110.9486…	1.1.1.1	144.202.109.208	TCP	443 → 55700 [ACK] Seq=3306 Ack=693 Win=30720 Len=0
143	110.9486…	1.1.1.1	144.202.109.208	HTTP2	SETTINGS[0]
144	110.9490…	1.1.1.1	144.202.109.208	TCP	443 → 55700 [ACK] Seq=3337 Ack=828 Win=30720 Len=0
145	110.9507…	144.202.109.208	1.1.1.1	TLSv1.3	Change Cipher Spec, Finished

Port 443 used

Notice HTTP/2 used.

DoH packet

DoH Packet Decrypted

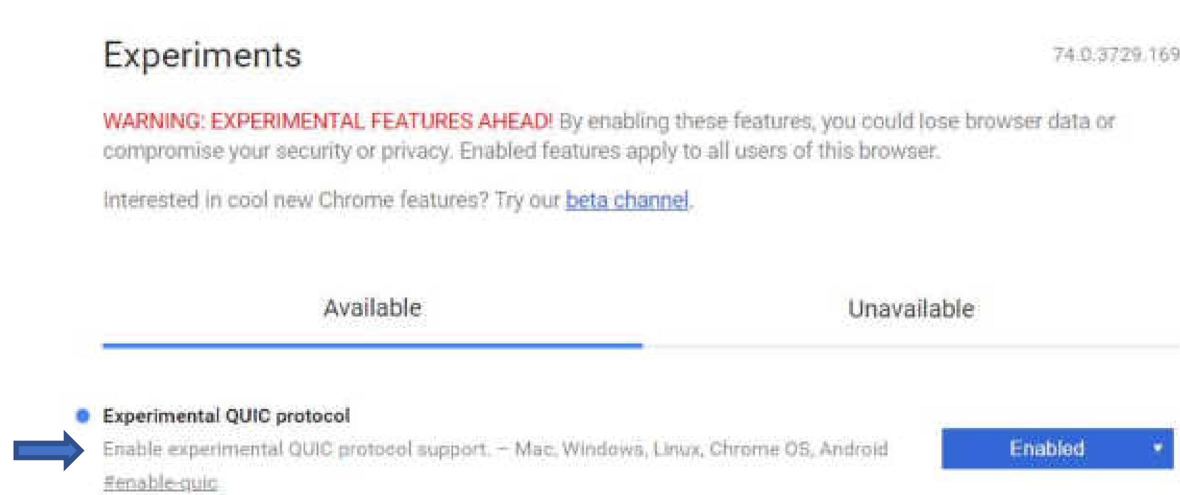


DNS over TLS

- Dnsprivacy.org
- Being displaced by DoH? Probably.

QUIC

- Enable on Windows Chrome
- Lots of work going on.
- Lots of bugs in downloads!



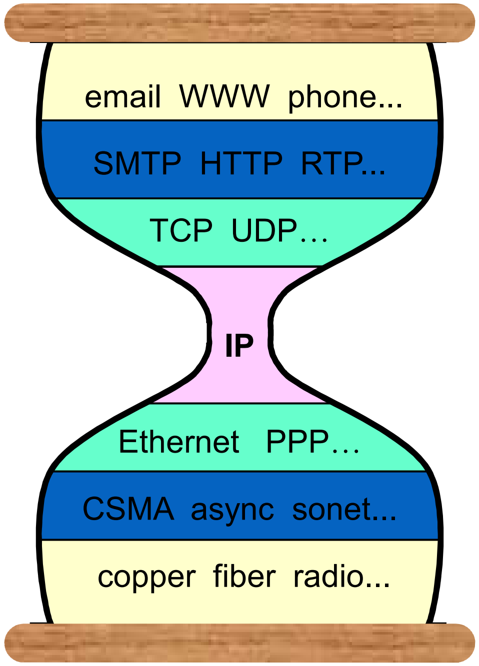
What is it?

- New transport layer (equivalent to TCP and UDP)
- New protocol to replace HTTP
- Originally from Google

The Internet hourglass

- 1998 version:
  - IP on everything:
    - Global addressing
    - Maximize interoperability

Idea: Least common functionalities to maximize the number of usable networks



S. Deering, Watch the Waist of the Protocol Hourglass. Keynote, IEEE ICNP 1998 and IETF 51, London, August 2001

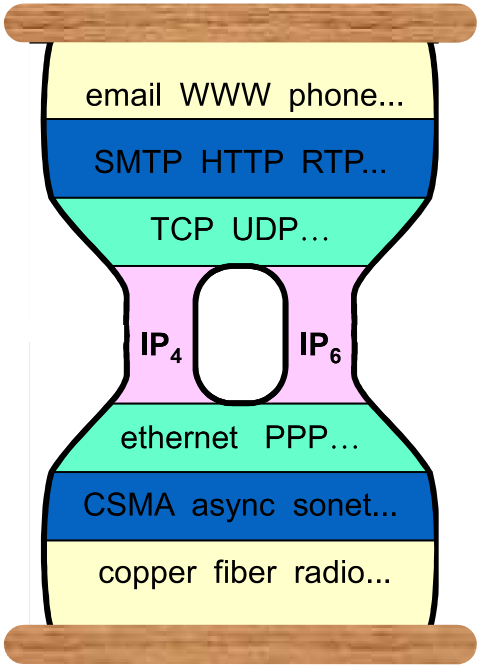
From March 16, 2017, EDCO
QUIC Presentation: Simone Ferlin

The Internet hourglass

- 1998 version:
  - IP on everything:
    - Global addressing
    - Maximise interoperability

It took over 20 years to deploy IPv6

- Lots of innovation in the application layers
  - The Internet grew a lot between these years...
- But only TCP or UDP as transport
  - SCTP (RFC2960, 4960, ...), DCCP (RFC4340) – or anything that is *different* did not get enough traction



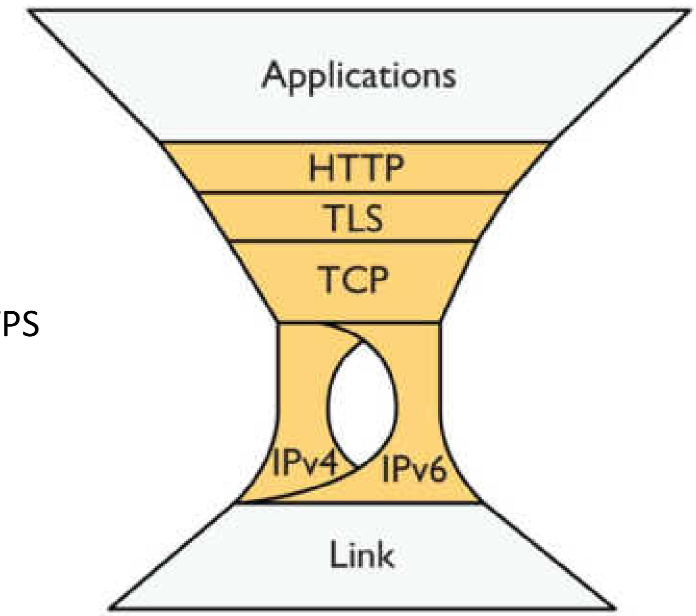
S. Deering, Watch the Waist of the Protocol Hourglass. Keynote, IEEE ICNP 1998 and IETF 51, London, August 2001

From March 16, 2017, EDCO
QUIC Presentation: Simone Ferlin

The Internet hourglass

2017 version:

- Still all over IP, but IPv4 and IPv6
- TCP is drowning out UDP
- HTTP and TLS (HTTPS) are part of the transport
  - More than 50% of the Internet's traffic is already HTTPS

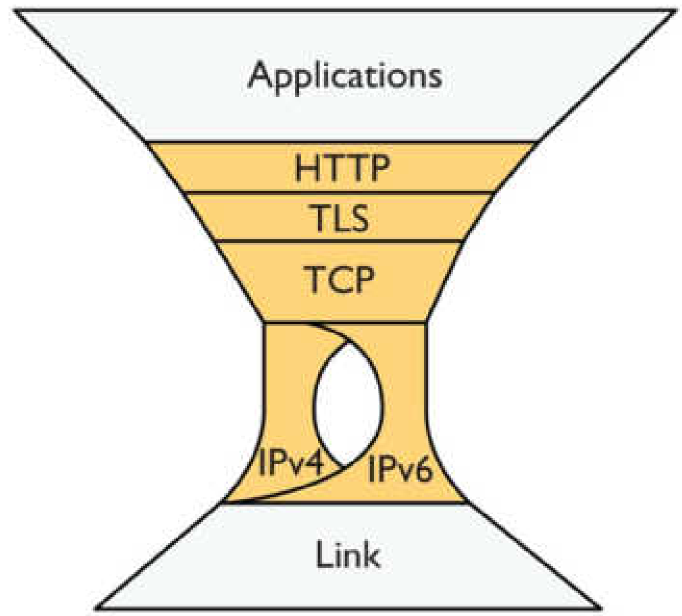


B. Trammell and J. Hildebrand, "Evolving Transport in the Internet", IEEE Internet Computing, vol. 18, no. 5, pp. 60-64, Sept.-Oct. 2014.

From March 16, 2017, EDCO
QUIC Presentation: Simone Ferlin

Why?

Innovation is difficult in some places:



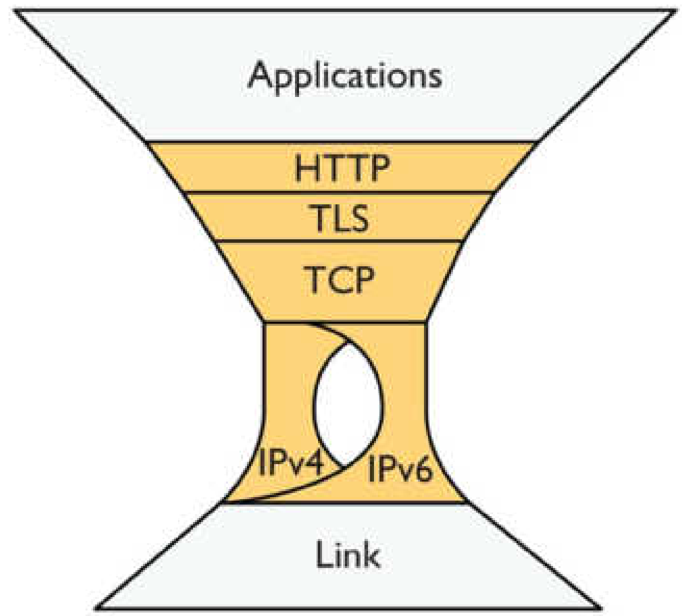
B. Trammell and J. Hildebrand, "Evolving Transport in the Internet", IEEE Internet Computing, vol. 18, no. 5, pp. 60-64, Sept.-Oct. 2014.

From March 16, 2017, EDCO
QUIC Presentation: Simone Ferlin

Why?

Innovation is difficult in some places:

- **Transport:**
  - Application developers resort to known, wide deployed protocols
  - OS (kernel) developers only implement a new protocol, if it gives benefits requested by (many) others.



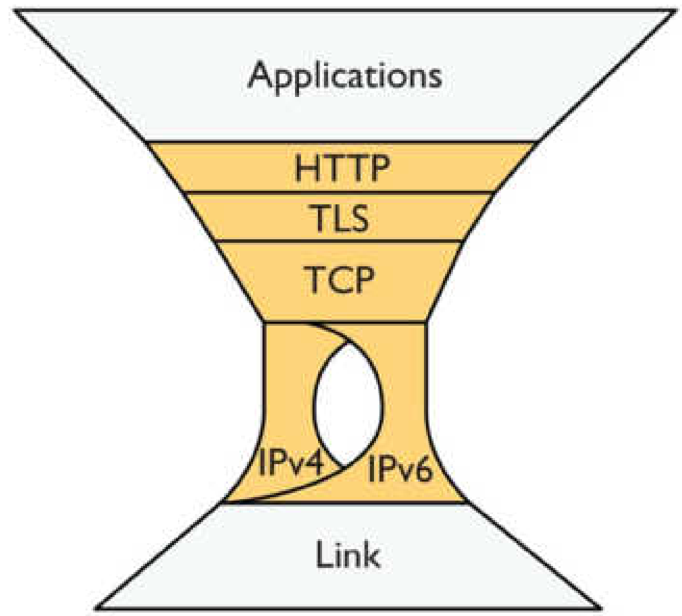
B. Trammell and J. Hildebrand, “Evolving Transport in the Internet”, IEEE Internet Computing, vol. 18, no. 5, pp. 60-64, Sept.-Oct. 2014.

From March 16, 2017, EDCO
QUIC Presentation: Simone Ferlin

Why?

Innovation is difficult in some places:

- **Transport:**
  - Application developers resort to known, wide deployed protocols
  - OS (kernel) developers only implement a new protocol, if it gives benefits requested by (many) others.
- **Network:**
  - The Internet is already too large and involves too many stakeholders on this layer (different goals, budget, etc.)

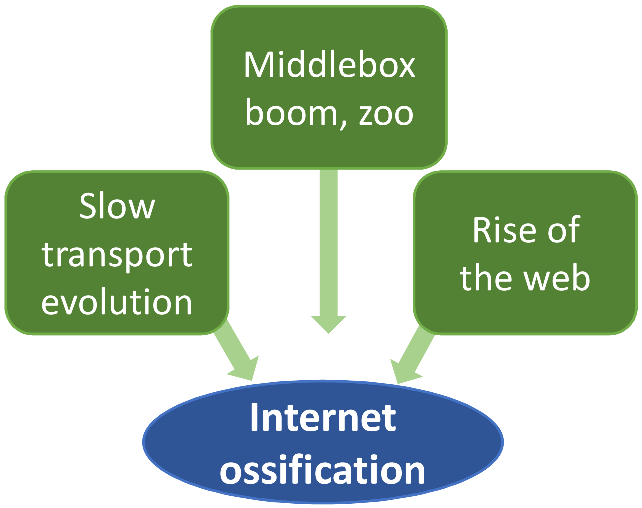


B. Trammell and J. Hildebrand, “Evolving Transport in the Internet”, IEEE Internet Computing, vol. 18, no. 5, pp. 60-64, Sept.-Oct. 2014.

From March 16, 2017, EDCO
QUIC Presentation: Simone Ferlin

What happened?

- **Transport:**
  - TCP evolves **very** slow.
    - MPTCP's, an extension of TCP for multiple paths RFC6824, largest work is dedicated to engineering **- not innovation**.

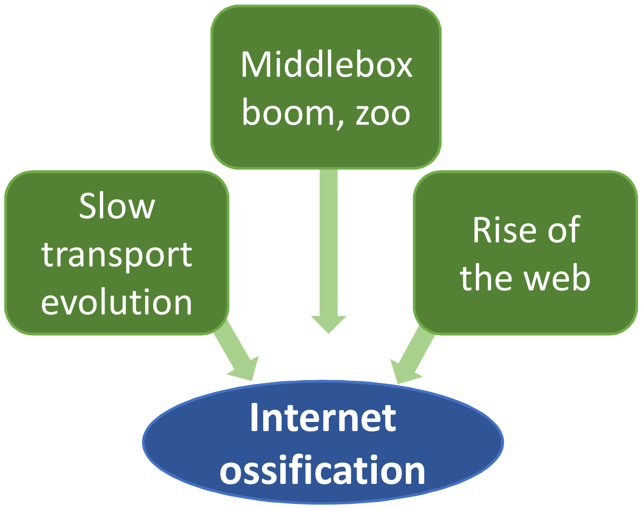


L. Eggert, "QUIC – A New Internet Transport. Guest Talk December, 14th, 2017, RWTH Aachen, Germany.

From March 16, 2017, EDCO
QUIC Presentation: Simone Ferlin

What happened?

- **Transport:**
  - TCP evolves **very** slow.
    - MPTCP's, an extension of TCP for multiple paths RFC6824, largest work is dedicated to engineering **- not innovation**.
- **Network:**
  - Made assumptions about TCP (and other traffic) and baking these inside TCP accelerators, FWs, NAT, etc.
  - Middlebox boom with IPv4 address exhaustion

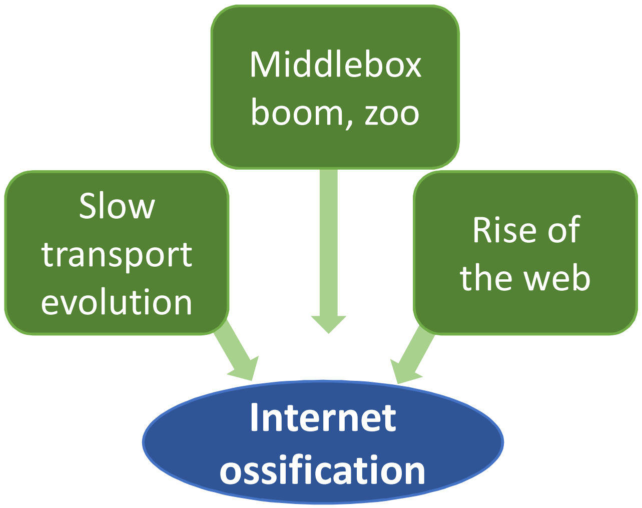


L. Eggert, "QUIC – A New Internet Transport. Guest Talk December, 14th, 2017, RWTH Aachen, Germany.

From March 16, 2017, EDCO
QUIC Presentation: Simone Ferlin

What happened?

- **Transport:**
  - TCP evolves **very** slow.
    - MPTCP's, an extension of TCP for multiple paths RFC6824, largest work is dedicated to engineering **- not innovation**.
- **Network:**
  - Made assumptions about TCP (and other traffic) and baking these inside TCP accelerators, FWs, NAT, etc.
  - Middlebox boom with IPv4 address exhaustion
- The web happened (through these years of fights for changes)
  - Amplified dominance with mobile web and cloud
    - Almost all content is HTTP(S) based



L. Eggert, "QUIC – A New Internet Transport. Guest Talk December, 14th, 2017, RWTH Aachen, Germany.

From March 16, 2017, EDCO QUIC Presentation: Simone Ferlin

Examples of ossification

Original:

snd/rcv from/to anywhere anytime →

Many protocols on top of only IP

E2E addressing

IP options to signal

Network is stateless

Data has meaning to applications only

Now:

Enforced directionality (middleboxes, FWs)

Packets dropped unless TCP or UDP

Network (NATs) rewrites options, e.g. ports

Options not used or dropped, no wide support

Network tracks entire connections, e.g. IDS/IPS

Network rewrite and insert data

From March 16, 2017, EDCO
QUIC Presentation: Simone Ferlin

L. Eggert, "QUIC – A New Internet Transport. Guest Talk
December, 14th, 2017, RWTH Aachen, Germany.

Transport Protocols are not aging well

TCP

- New TCP must look like old TCP
  - Multipath TCP was an engineering challenge
- TCP semantics is already complicated
  - New TCP must look like old TCP
- TCP headers are not encrypted or even authenticated
  - “TCP accelerators”
- TCP options space is limited and crowded
  - TCP header 20B without options, max. of 60B with options, i.e. 40B for option space: window scale (3), timestamp (10), MSS (4), SACK (2) MPTCP needs 12B
- Slow upgrade cycles
  - Old machines with old kernels (high-risk, invasive)

From March 16, 2017, EDCO
QUIC Presentation: Simone Ferlin

End-to-end Principle

"Some of us who have been in the IETF for a long time find that having smart endpoints and a dumb network is the best architecture. This is the end-to-end principle."

Unsustainable

- Others believe that the end-to-end principle leads to an unsustainable trajectory to ever more complex endpoints and network functions.

- Middleboxes serve useful functions (load balancers, firewalls, NAT, etc)

EMAIL to QUIC WG

- However, in those discussions, a related concern was identified; confusion between QUIC-the-transport-protocol, and QUIC-the-HTTP-binding. I and others have seen a number of folks not closely involved in this work conflating the two, even though they're now separate things.
- To address this, I'd like to suggest that -- after coordination with the HTTP WG -- we rename our the[sic] HTTP document to **"HTTP/3",** and using the final ALPN token "h3". Doing so clearly identifies it as another binding of HTTP semantics to the wire protocol -- just as HTTP/2 did -- so people understand its separation from QUIC.
- Oct. 18, 2018: Mark Nottingham: co-chair QUIC WG

GQUIC Traces

Gquic.pcapng

File Edit View Go Capture Analyze Statistics Telephony Wireless Tools Help

(ipv6.addr == 2607:f8b0:4005:805::2004)

No.	Time	Source	Destination	Protocol	Info
99	5.61...	2607:f8b0:4005:805::2004	2601:642:c202:9550:50a3:d610:bc1d:721b	TLSv1.3	Server Hello, Change Cipher Spec
1...	5.61...	2607:f8b0:4005:805::2004	2601:642:c202:9550:50a3:d610:bc1d:721b	TCP	443 → 54362 [ACK] Seq=1221 Ack=518 Win=2841
1...	5.61...	2607:f8b0:4005:805::2004	2601:642:c202:9550:50a3:d610:bc1d:721b	TLSv1.3	Encrypted Extensions, Certificate, Certific
1...	5.61...	2601:642:c202:9550:50a3:d610:bc1d:721b	2607:f8b0:4005:805::2004	TCP	54362 → 443 [ACK] Seq=518 Ack=2441 Win=6656
1...	5.64...	2601:642:c202:9550:50a3:d610:bc1d:721b	2607:f8b0:4005:805::2004	TLSv1.3	Change Cipher Spec, Finished
1...	5.64...	2601:642:c202:9550:50a3:d610:bc1d:721b	2607:f8b0:4005:805::2004	TLSv1.3	Change Cipher Spec, Finished
1...	5.64...	2601:642:c202:9550:50a3:d610:bc1d:721b	2607:f8b0:4005:805::2004	TCP	54362 → 443 [FIN, ACK] Seq=582 Ack=2627 Wir
1...	5.65...	2601:642:c202:9550:50a3:d610:bc1d:721b	2607:f8b0:4005:805::2004	HTTP2	Magic, SETTINGS[0], WINDOW_UPDATE[0]
1...	5.65...	2601:642:c202:9550:50a3:d610:bc1d:721b	2607:f8b0:4005:805::2004	HTTP2	HEADERS[1]: GET /complete/search?client=chr
1...	5.65...	2601:642:c202:9550:50a3:d610:bc1d:721b	2607:f8b0:4005:805::2004	HTTP2	HEADERS[3]: GET /async/ddljson?async=ntp:1
1...	5.66...	2607:f8b0:4005:805::2004	2601:642:c202:9550:50a3:d610:bc1d:721b	TCP	443 → 54361 [ACK] Seq=2626 Ack=668 Win=2841
1...	5.66...	2607:f8b0:4005:805::2004	2601:642:c202:9550:50a3:d610:bc1d:721b	HTTP2	SETTINGS[0], WINDOW_UPDATE[0]
1...	5.66...	2601:642:c202:9550:50a3:d610:bc1d:721b	2607:f8b0:4005:805::2004	HTTP2	SETTINGS[0]
1...	5.66...	2607:f8b0:4005:805::2004	2601:642:c202:9550:50a3:d610:bc1d:721b	HTTP2	SETTINGS[0], WINDOW_UPDATE[0]
1...	5.66...	2601:642:c202:9550:50a3:d610:bc1d:721b	2607:f8b0:4005:805::2004	TCP	54362 → 443 [RST, ACK] Seq=583 Ack=3191 Wir
1...	5.66...	2607:f8b0:4005:805::2004	2601:642:c202:9550:50a3:d610:bc1d:721b	HTTP2	SETTINGS[0]
1...	5.66...	2607:f8b0:4005:805::2004	2601:642:c202:9550:50a3:d610:bc1d:721b	TCP	443 → 54362 [FIN, ACK] Seq=3191 Ack=583 Wir
1...	5.66...	2607:f8b0:4005:805::2004	2601:642:c202:9550:50a3:d610:bc1d:721b	TCP	443 → 54361 [ACK] Seq=3221 Ack=2048 Win=312
2...	5.70...	2601:642:c202:9550:50a3:d610:bc1d:721b	2607:f8b0:4005:805::2004	GQUIC	Client Hello, PKN: 1, CID: 925949590225281
2...	5.70...	2607:f8b0:4005:805::2004	2601:642:c202:9550:50a3:d610:bc1d:721b	HTTP2	HEADERS[3]: 200 OK
2...	5.70...	2601:642:c202:9550:50a3:d610:bc1d:721b	2607:f8b0:4005:805::2004	TCP	54361 → 443 [ACK] Seq=2079 Ack=3644 Win=652
2...	5.70...	2607:f8b0:4005:805::2004	2601:642:c202:9550:50a3:d610:bc1d:721b	HTTP2	DATA[3]
2...	5.70...	2607:f8b0:4005:805::2004	2601:642:c202:9550:50a3:d610:bc1d:721b	HTTP2	DATA[3] (application/json)
2...	5.70...	2607:f8b0:4005:805::2004	2601:642:c202:9550:50a3:d610:bc1d:721b	HTTP2	PING[0]
2...	5.70...	2601:642:c202:9550:50a3:d610:bc1d:721b	2607:f8b0:4005:805::2004	TCP	54361 → 443 [ACK] Seq=2079 Ack=3729 Win=652

GQUIC

File Edit View Go Capture Analyze Statistics Telephony Wireless Tools Help

(ipv6.addr == 2607:f8b0:4005:805::2004)

No.	Time	Source	Destination	Protocol	Info
2…	5.71…	2601:642:c202:9550:50a3:d610:bc1d:721b	2607:f8b0:4005:805::2004	TCP	54361 → 443 [ACK] Seq=2118 Ack=4223 Win=66
2…	5.73…	2601:642:c202:9550:50a3:d610:bc1d:721b	2607:f8b0:4005:805::2004	GQUIC	Client Hello, PKN: 2, CID: 925949590225281
2…	5.73…	2601:642:c202:9550:50a3:d610:bc1d:721b	2607:f8b0:4005:805::2004	HTTP2	HEADERS[5]: GET /async/newtab_ogb?hl=en-US
2…	5.73…	2601:642:c202:9550:50a3:d610:bc1d:721b	2607:f8b0:4005:805::2004	HTTP2	HEADERS[7]: GET /async/newtab_promos
2…	5.74…	2607:f8b0:4005:805::2004	2601:642:c202:9550:50a3:d610:bc1d:721b	GQUIC	Rejection, PKN: 1, CID: 925949590225281496
2…	5.74…	2607:f8b0:4005:805::2004	2601:642:c202:9550:50a3:d610:bc1d:721b	GQUIC	Payload (Encrypted), PKN: 2, CID: 92594959
2…	5.74…	2601:642:c202:9550:50a3:d610:bc1d:721b	2607:f8b0:4005:805::2004	GQUIC	Payload (Encrypted), PKN: 3, CID: 92594959
2…	5.74…	2607:f8b0:4005:805::2004	2601:642:c202:9550:50a3:d610:bc1d:721b	TCP	443 → 54361 [ACK] Seq=4223 Ack=2212 Win=31
2…	5.74…	2601:642:c202:9550:50a3:d610:bc1d:721b	2607:f8b0:4005:805::2004	GQUIC	Client Hello, PKN: 4, CID: 925949590225281
2…	5.76…	2607:f8b0:4005:805::2004	2601:642:c202:9550:50a3:d610:bc1d:721b	GQUIC	Payload (Encrypted), PKN: 3, CID: 92594959
2…	5.78…	2607:f8b0:4005:805::2004	2601:642:c202:9550:50a3:d610:bc1d:721b	GQUIC	Payload (Encrypted), PKN: 4
2…	5.78…	2607:f8b0:4005:805::2004	2601:642:c202:9550:50a3:d610:bc1d:721b	GQUIC	Payload (Encrypted), PKN: 5
2…	5.78…	2601:642:c202:9550:50a3:d610:bc1d:721b	2607:f8b0:4005:805::2004	GQUIC	Payload (Encrypted), PKN: 5, CID: 92594959
2…	5.78…	2601:642:c202:9550:50a3:d610:bc1d:721b	2607:f8b0:4005:805::2004	GQUIC	Payload (Encrypted), PKN: 6, CID: 92594959
2…	5.79…	2607:f8b0:4005:805::2004	2601:642:c202:9550:50a3:d610:bc1d:721b	TCP	443 → 54361 [ACK] Seq=4223 Ack=2290 Win=31
2…	5.79…	2607:f8b0:4005:805::2004	2601:642:c202:9550:50a3:d610:bc1d:721b	HTTP2	HEADERS[7]: 200 OK
2…	5.79…	2607:f8b0:4005:805::2004	2601:642:c202:9550:50a3:d610:bc1d:721b	HTTP2	DATA[7]
2…	5.79…	2607:f8b0:4005:805::2004	2601:642:c202:9550:50a3:d610:bc1d:721b	HTTP2	DATA[7] (application/json)
2…	5.79…	2607:f8b0:4005:805::2004	2601:642:c202:9550:50a3:d610:bc1d:721b	HTTP2	PING[0]
2…	5.79…	2601:642:c202:9550:50a3:d610:bc1d:721b	2607:f8b0:4005:805::2004	TCP	54361 → 443 [ACK] Seq=2290 Ack=4497 Win=66
2…	5.79…	2601:642:c202:9550:50a3:d610:bc1d:721b	2607:f8b0:4005:805::2004	TCP	54361 → 443 [ACK] Seq=2290 Ack=4567 Win=66
2…	5.80…	2601:642:c202:9550:50a3:d610:bc1d:721b	2607:f8b0:4005:805::2004	HTTP2	PING[0]
3…	5.81…	2607:f8b0:4005:805::2004	2601:642:c202:9550:50a3:d610:bc1d:721b	TCP	443 → 54361 [ACK] Seq=4567 Ack=2329 Win=31
3…	5.81…	2607:f8b0:4005:805::2004	2601:642:c202:9550:50a3:d610:bc1d:721b	HTTP2	HEADERS[5]: 200 OK
3…	5.81…	2607:f8b0:4005:805::2004	2601:642:c202:9550:50a3:d610:bc1d:721b	TLSv1.3	[TLS segment of a reassembled PDU]
3…	5.81…	2607:f8b0:4005:805::2004	2601:642:c202:9550:50a3:d610:bc1d:721b	TLSv1.3	[TLS segment of a reassembled PDU]

Notice

- TLS1.3 and GQUIC packets interspersed
- GQUIC packets not decrypted
- TLS1.3 decrypted
- Same two endpoints

IPv6 PDM: RFC8250

[Docs] [txt|pdf] [draft-ietf-ippm...] [Tracker] [Diff1] [Diff2] [IPR]

PROPOSED STANDARD

Internet Engineering Task Force (IETF)
Request for Comments: 8250
Category: Standards Track
ISSN: 2070-1721

N. Elkins
Inside Products
R. Hamilton
Chemical Abstracts Service
M. Ackermann
BCBS Michigan
September 2017

IPv6 Performance and Diagnostic Metrics (PDM) Destination Option

Abstract

To assess performance problems, this document describes optional headers embedded in each packet that provide sequence numbers and timing information as a basis for measurements. Such measurements may be interpreted in real time or after the fact. This document specifies the Performance and Diagnostic Metrics (PDM) Destination Options header. The field limits, calculations, and usage in measurement of PDM are included in this document.

Status of This Memo

This is an Internet Standards Track document.

This document is a product of the Internet Engineering Task Force (IETF). It represents the consensus of the IETF community. It has received public review and has been approved for publication by the Internet Engineering Steering Group (IESG). Further information on

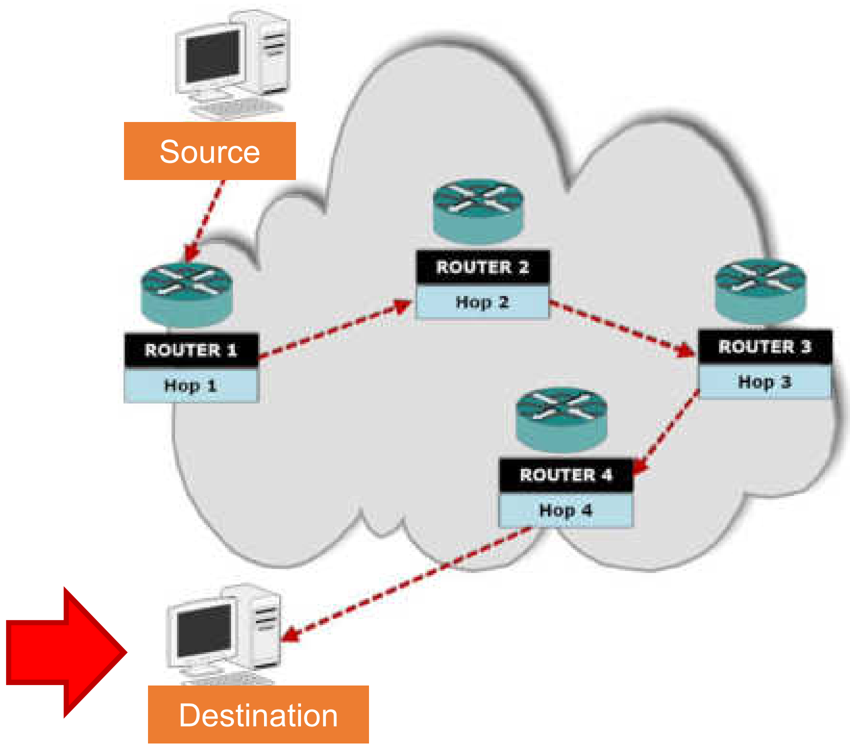
- Standards track RFC
- IETF consensus document
- Implemented in FreeBSD (proprietary)
- Why?
- Presentation from IETF follows

Common IPv6 Extension Headers

Next Header (Hex)	Next Header (Decimal)	Header Name	Description
0	0	Hop-by-Hop Options	For all devices on the path
2B	43	Routing	0 – Source Routing (deprecated) 2 – Mobile IPv6
2C	44	Fragment	Only when packet is fragmented
32	50	Encapsulated Security Payload (ESP)	IPSec encrypted data
33	51	Authentication Header (AH)	IPSec authentication
3C	60	Destination Options	http://www.iana.org/assignments/ipv6-parameters/ipv6-parameters.xml (Mobile IP, etc)

IPv6 Destination Options

- Destination Options: for end host



IPID FIELD IN IPv4 - BACKGROUND

- **IPID: Internet Protocol Identification. Provides a unique identifying number for a given IP Packet within a flow.**
- Sometimes called Datagram number.
- **USAGE/VALUE**
- Enable Fragmentation.
- Packet sequencing at end points (Edge Networks).
- **Diagnostics! Logically associate packets across complex network situations.**
- IPID is frequently used in IPv4 troubleshooting for the purposes of “watermarking” the packets to correlate them in different troubleshooting scenarios. The implementations are such that the IPID is infrequently changed by middle boxes even if the content is.

IPID FIELD IN IPv6 – CURRENT STATE

- **IMPLEMENTED IN FRAGMENT HEADER EXTENSION (TYPE 44).**
- **LOCATION:**
  - 32 bit field at offset 4 in FHE.
- **ISSUES:**
  - Only used if fragmentation required!
- **IPID not always available to facilitate network diagnostics!**

Why We Need It

- Provides recognition of sequencing and duplication of packets
  - TCP SEQ / ACK (retransmissions, duplication: true and false)
  - UDP – no sequence number
  - ICMP – need to see sequence number in embedded packet
  - Across multiple trace points
  - **It's not going to get any easier.**

Added Response Time

- As we progressed, we could see that end-to-end response time as well as breakout of server and network time was missing!
- Also, if we add that, we could get support from IP Performance Metrics (IPPM) Working Group at the IETF

IPPM Considerations for the IPv6 PDM Destination Option

Nalini Elkins – Inside Products, Inc.

IETF91

We propose:

Requirement

- In basic IP transport
- Undisturbed by middle systems

Solution

•**Implementation** of existing extension header: Destination Options Header (DOH)

•Performance and Diagnostic Metrics (PDM) DOH

PDM

- Performance and Diagnostic Metrics Destination Option (PDM) contains the following fields: (by 5-tuple)

- PSNTP : Packet Sequence Number This Packet
- PSNLR : Packet Sequence Number Last Received
- DELTALR : Delta Last Received
- DELTALS : Delta Last Sent
- TIMEBASE : Base timer unit
- SCALEDL : Scale for Delta Last Received
- SCALEDS : Scale for Delta Last Sent

PDM Timing

- No time synchronization needed
- All times are in relation to self

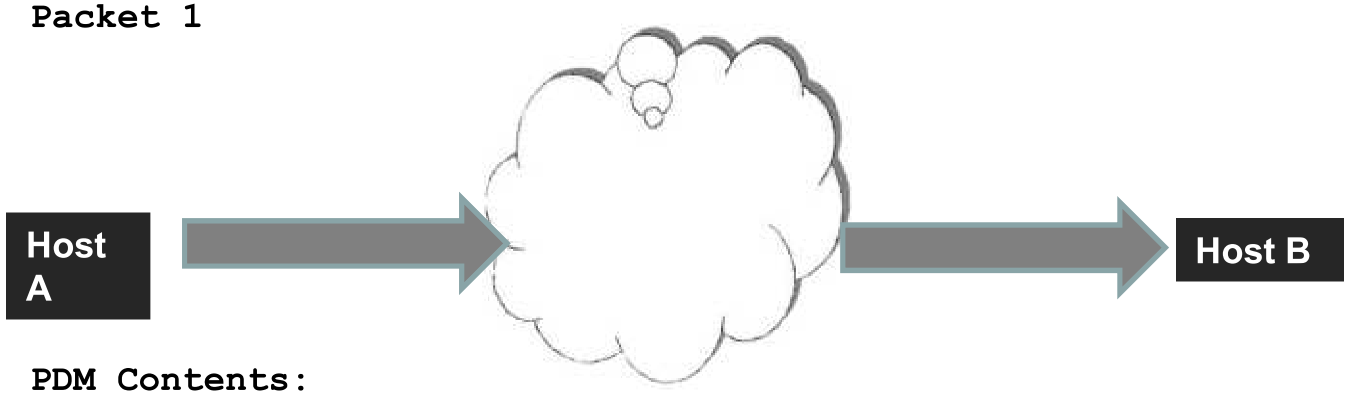
Start Flow

- Packet 1 is sent from Host A to Host B. The time for Host A is 10:00AM.

- The time and packet sequence number are saved by Host A internally. The packet sequence number and delta times are sent in the packet.

Packet 1

Packet 1



PDM Contents:

PSNTP	: Packet Sequence Number This Packet:	25
PSNLR	: Packet Sequence Number Last Received:	-
DELTALR	: Delta Last Received:	-
DELTALS	: Delta Last Sent:	-

Keep in Host A

- Internally, within the sender, Host A, it must keep:

- Packet Seq. Number of last packet sent: 25
- Time the last packet was sent: 10:00:00

Keep in Host B

- Packet 1 is received at Host B. Its time is set to one hour later than Host A. In this case, 11:00AM
- Internally, within the receiver, Host B, it must note:

- Packet Seq. Number of last packet received: 25
- Time the last packet was received : 11:00:03

Server Delay

- Host B processes packet 1 and creates a response (packet 2).
- Packet 2 is sent by Host B to Host A.
- This is the time taken by Host B or Server Delay
- Server Delay = Sending time (packet 2) - receive time (packet 1)

DeltaLR

- We will call the result of this calculation: Delta Last Received

- DELTALR = Sending time (packet 2) - receive time (packet 1)

- Note, both sending time and receive time are saved internally in Host B. They do not travel in the packet. Only the Delta is in the packet.

Host B Stats

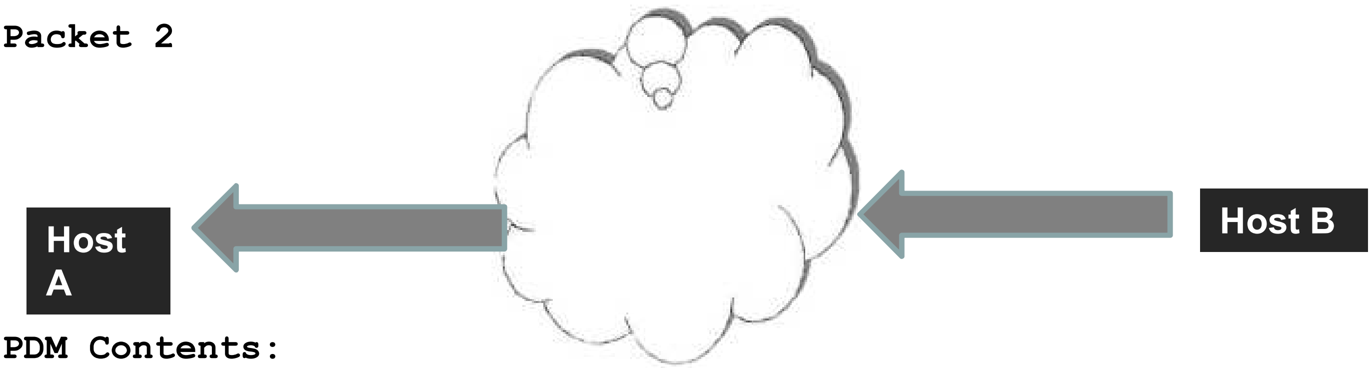
- Within Host B is the following:

- Packet Sequence Number of the last packet received: 25
- Time the last packet was received: 11:00:03
- Packet Sequence Number of this packet: 12
- Time this packet is being sent: 11:00:07

- DELTALR = 4 seconds (11:00:07 - 11:00:03)
- DELTALR is Server Delay.

Packet 2

Packet 2



PDM Contents:

PSNTP	: Packet Sequence Number This Packet:	12
PSNLR	: Packet Sequence Number Last Received:	25
DELTALR	: Delta Last Received:	4 seconds
DELTALS	: Delta Last Sent:	-

Metrics Needed

- The metrics left to be calculated are end-to-end time and round-trip delay (network time).

- This will be calculated by Host A when it receives Packet 2.

Packet 2 Received

- Packet 2 is received at Host A. Remember, its time is set to one hour earlier than Host B. Internally, it must note:

- Packet Sequence Number of the last packet received: 12
- Time the last packet was received : 10:00:12

- Note, this timestamp is in Host A time. It has nothing whatsoever to do with Host B time.

End-to-End Time

- Now, Host A can calculate total end-to-end time.
- End-to-End Time = Time Last Received - Time Last Sent
- Packet 1 was sent by Host A at 10:00:00. Packet 2 was received by Host A at 10:00:12
- End-to-End time = 10:00:12 - 10:00:00 or 12

- This metric we will call DELTALS or Delta Last Sent

Network TIme

- We can now also calculate round trip delay (network time). The formula is:

- Round trip delay = DELTALS - DELTALR

- Or: End-to-end time – Server Delay

- Round trip delay = 12 - 4 or 8

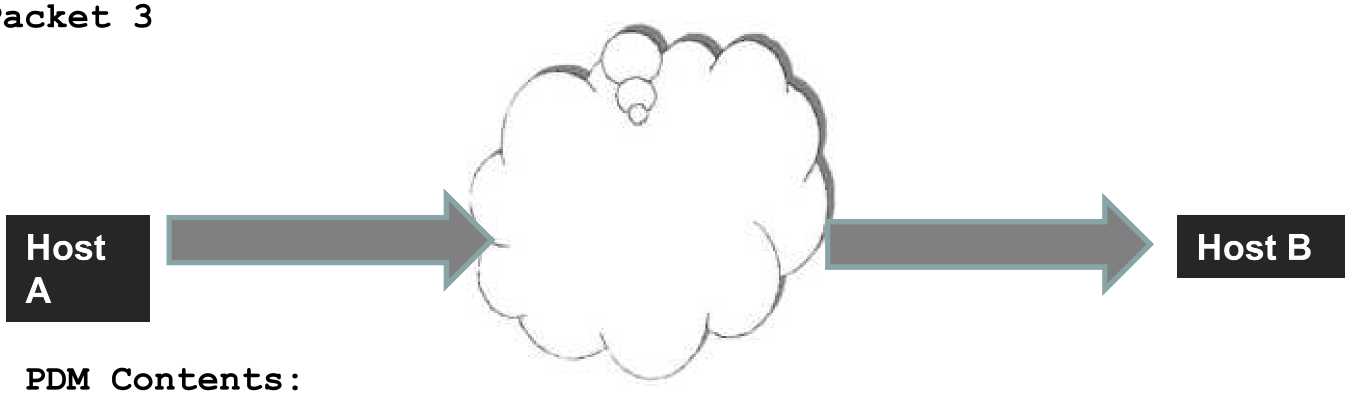
How to Communicate?

- Now, the only problem is that at this point all metrics are in Host A only and not exposed in a packet.

- To do that, we need a third packet.

Packet 3

Packet 3



PDM Contents:

PSNTP	:	Packet Sequence Number This Packet:	26
PSNLR	:	Packet Sequence Number Last Received:	12
DELTALR	:	Delta Last Received:	0
DELTALS	:	Delta Last Sent:	12 seconds

Questions from IETF91 (Answered in IETF 92: See Appendix)

1. Does PDM have enough variables to actually diagnose problems?
2. Are all PDM fields necessary?
3. Why is the proposal for an IPv6 extension header rather than a TCP option? Only TCP is important.
4. Does PDM create too much overhead?
5. Will PDM work for complex apps not just simple applications with one send and one receive?

Ping to Loopback (::1)

pdm072019-02.pcap

File Edit View Go Capture Analyze Statistics Telephony Wireless Tools Help

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

No.	Time	Source	Destination	Protocol	Info
1	0.000000	::1	::1	ICMPv6	Echo (ping) request id=0x0367, seq=0, hop limit=64 (reply in 2)
2	0.000032	::1	::1	ICMPv6	Echo (ping) reply id=0x0367, seq=0, hop limit=64 (request in 1)
3	1.018026	::1	::1	ICMPv6	Echo (ping) request id=0x0367, seq=1, hop limit=64 (reply in 4)
4	1.018055	::1	::1	ICMPv6	Echo (ping) reply id=0x0367, seq=1, hop limit=64 (request in 3)
5	2.048797	::1	::1	ICMPv6	Echo (ping) request id=0x0367, seq=2, hop limit=64 (reply in 6)
6	2.048823	::1	::1	ICMPv6	Echo (ping) reply id=0x0367, seq=2, hop limit=64 (request in 5)
7	3.108793	::1	::1	ICMPv6	Echo (ping) request id=0x0367, seq=3, hop limit=64 (reply in 8)
8	3.108821	::1	::1	ICMPv6	Echo (ping) reply id=0x0367, seq=3, hop limit=64 (request in 7)

```
> Frame 1: 76 bytes on wire (608 bits), 76 bytes captured (608 bits)
> Null/Loopback
v Internet Protocol Version 6, Src: ::1, Dst: ::1
    0110 .... = Version: 6
  > .... 0000 0000 .... .... .... .... .... = Traffic Class: 0x00 (DSCP: CS0, ECN: Not-ECT)
    .... .... .... 0000 0000 0000 0000 0000 = Flow Label: 0x00000
    Payload Length: 32
    Next Header: Destination Options for IPv6 (60)
    Hop Limit: 64
    Source: ::1
    Destination: ::1
  v Destination Options for IPv6
      Next Header: ICMPv6 (58)
      Length: 1
```

Destination Options: IANA

Destination Options and Hop-by-Hop Options

Registration Procedure(s)
IESG Approval, IETF Review or Standards Action

Reference
[RFC8200][RFC2780]

Note
From [RFC8200] IPv6 Option Types are 8-bit values, structured as three subfields, are defined in Section 4.2 of [RFC8200].

Each distinct 8-bit Option Type identifies a different option, i.e., the high-order 3 bits are considered part of the option identification. However, it is recommended that Option Types be assigned with distinct values in the "rest" subfield, until and unless that 5-bit space becomes full.

Available Formats
CSV

Hex Value	Binary Value			Description	Reference
	act	chg	rest		
0x00	00	0	00000	Pad1	[[IPV6]]
0x01	00	0	00001	PadN	[[IPV6]]
0xC2	11	0	00010	Jumbo Payload	[RFC2675]
0x63	01	1	00011	RPL Option	[RFC6553]
0x04	00	0	00100	Tunnel Encapsulation Limit	[RFC2473]
0x05	00	0	00101	Router Alert	[RFC2711]
0x26	00	1	00110	Quick-Start	[RFC4782][RFC Errata 2034]
0x07	00	0	00111	CALIPSO	[RFC5570]
0x08	00	0	01000	SMF_DPD	[RFC6621]
0xC9	11	0	01001	Home Address	[RFC6275]
0x8A	10	0	01010	Endpoint Identification (DEPRECATED)	[[CHARLES LYNN]]
0x8B	10	0	01011	ILNP Nonce	[RFC6744]
0x8C	10	0	01100	Line-Identification Option	[RFC6788]
0x4D	01	0	01101	Deprecated	[RFC7731]
0x6D	01	1	01101	MPL Option	[RFC7731]
0xEE	11	1	01110	IP_DFF	[RFC6971]
0x0F	00	0	01111	Performance and Diagnostic Metrics (PDM)	[RFC8250]
			10000-11101	Unassigned	

https://www.iana.org/assignments/ipv6-parameters/ipv6-parameters.xhtml#ipv6-parameters-2

PDM Destination Option

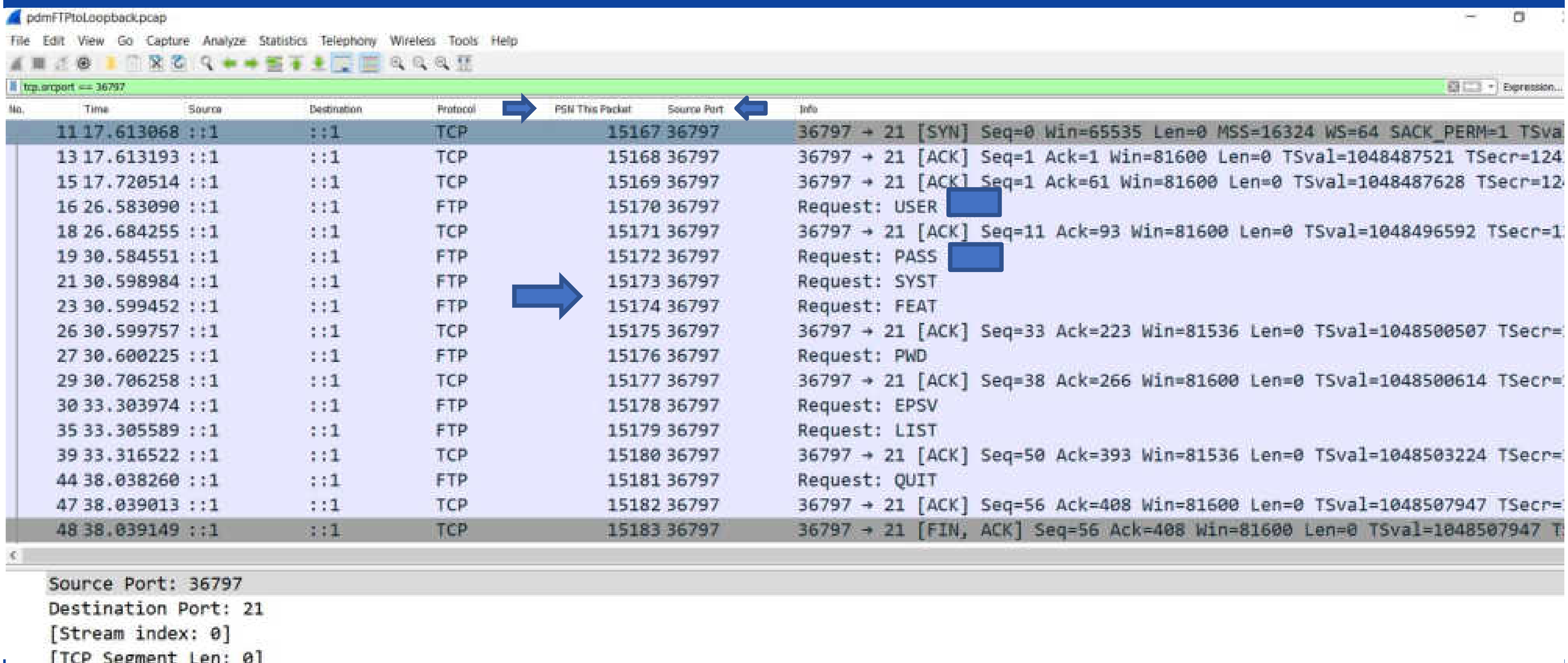
```
> Frame 1: 76 bytes on wire (608 bits), 76 bytes captured (608 bits)
> Null/Loopback
v Internet Protocol Version 6, Src: ::1, Dst: ::1
    0110 .... = Version: 6
  > .... 0000 0000 .... .... .... .... .... = Traffic Class: 0x00 (DSCP: CS0, ECN: Not-ECT)
    .... .... .... 0000 0000 0000 0000 0000 = Flow Label: 0x00000
    Payload Length: 32
    Next Header: Destination Options for IPv6 (60)
    Hop Limit: 64
    Source: ::1
    Destination: ::1
  v Destination Options for IPv6
      Next Header: ICMPv6 (58)
      Length: 1
      [Length: 16 bytes]
    v Performance and Diagnostic Metrics
      > Type: Performance and Diagnostic Metrics (0x0f)
        Length: 10
        Scale DTLR: 0
        Scale DTLS: 0
        PSN This Packet: 7015
        PSN Last Received: 0
        Delta Time Last Received: 0
        Delta Time Last Sent: 0
    > PadN
> Internet Control Message Protocol v6
```

PDM Layout

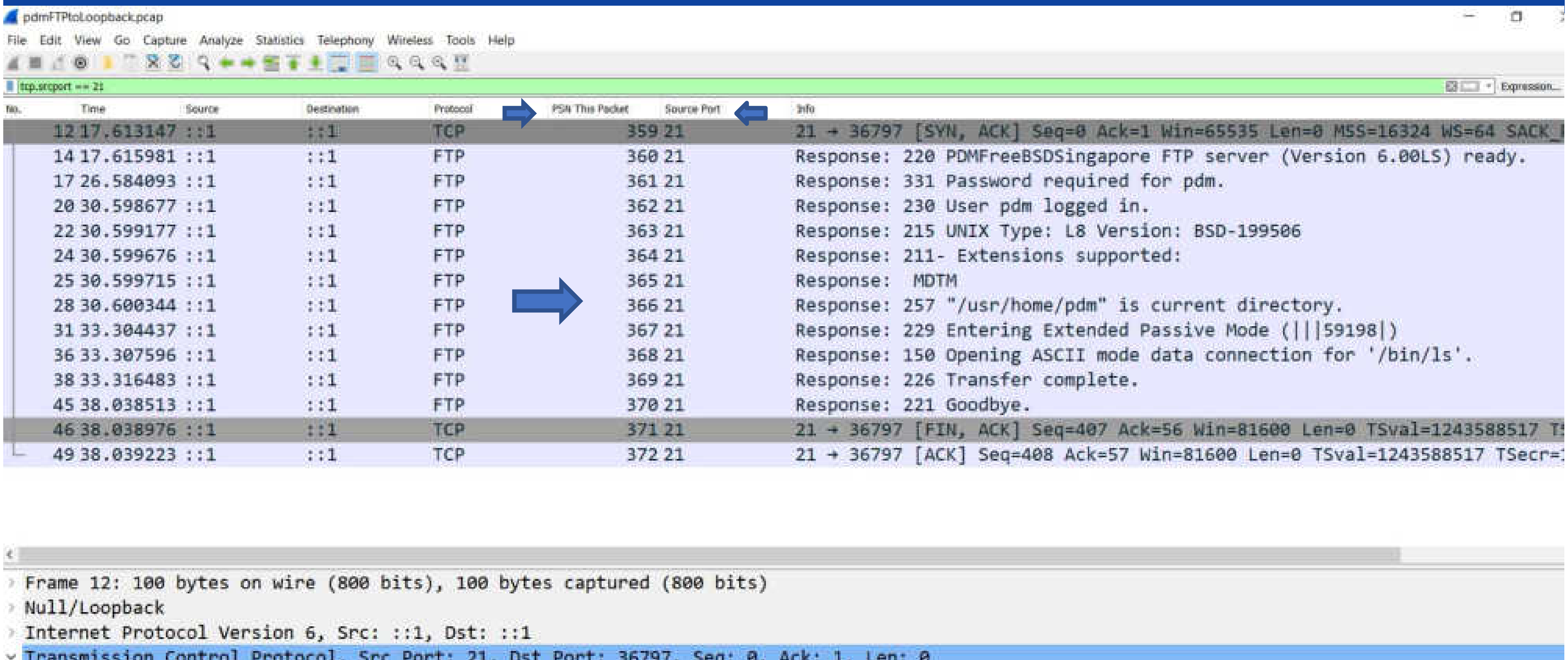
Performance and Diagnostic Metrics Destination Option (PDM) contains the following fields: (by 5-tuple)

- PSNTP : Packet Sequence Number This Packet
- PSNLR : Packet Sequence Number Last Received
- DELTALR : Delta Last Received
- DELTALS : Delta Last Sent
- SCALEDL : Scale for Delta Last Received
- SCALEDS : Scale for Delta Last Sent

FTP to Loopback



The Other Way



SSH to PDM Enabled Server

pdmjune07-03-SSH-PDM-BothWays.pcap

File Edit View Go Capture Analyze Statistics Telephony Wireless Tools Help

tcp.srcport == 22

No.	Time	Source	Destination	Protocol	PSN This Packe	Source Port	Info
3	3.714796	2001:19f0:5401:2cf4:5400:2ff:fe18:eefb	2001:19f0:5401:5b9:5400:2ff:fe18:dbdf	TCP	61485	22	22 → 39535 [SYN, ACK] Seq
6	3.723501	2001:19f0:5401:2cf4:5400:2ff:fe18:eefb	2001:19f0:5401:5b9:5400:2ff:fe18:dbdf	SSHv2	61486	22	Server: Protocol (SSH-2.0
8	3.725120	2001:19f0:5401:2cf4:5400:2ff:fe18:eefb	2001:19f0:5401:5b9:5400:2ff:fe18:dbdf	SSHv2	61487	22	Server: Key Exchange Init
10	3.732302	2001:19f0:5401:2cf4:5400:2ff:fe18:eefb	2001:19f0:5401:5b9:5400:2ff:fe18:dbdf	SSHv2	61488	22	Server: Elliptic Curve Di
13	5.970880	2001:19f0:5401:2cf4:5400:2ff:fe18:eefb	2001:19f0:5401:5b9:5400:2ff:fe18:dbdf	TCP	61489	22	22 → 39535 [ACK] Seq=1443
15	5.971391	2001:19f0:5401:2cf4:5400:2ff:fe18:eefb	2001:19f0:5401:5b9:5400:2ff:fe18:dbdf	SSHv2	61490	22	Server: Encrypted packet
17	6.077869	2001:19f0:5401:2cf4:5400:2ff:fe18:eefb	2001:19f0:5401:5b9:5400:2ff:fe18:dbdf	TCP	61491	22	22 → 39535 [ACK] Seq=1487
18	6.391451	2001:19f0:5401:2cf4:5400:2ff:fe18:eefb	2001:19f0:5401:5b9:5400:2ff:fe18:dbdf	SSHv2	61492	22	Server: Encrypted packet
20	6.402328	2001:19f0:5401:2cf4:5400:2ff:fe18:eefb	2001:19f0:5401:5b9:5400:2ff:fe18:dbdf	SSHv2	61493	22	Server: Encrypted packet
25	10.553858	2001:19f0:5401:2cf4:5400:2ff:fe18:eefb	2001:19f0:5401:5b9:5400:2ff:fe18:dbdf	SSHv2	61494	22	Server: Encrypted packet
27	10.555114	2001:19f0:5401:2cf4:5400:2ff:fe18:eefb	2001:19f0:5401:5b9:5400:2ff:fe18:dbdf	SSHv2	61495	22	Server: Encrypted packet
29	10.556233	2001:19f0:5401:2cf4:5400:2ff:fe18:eefb	2001:19f0:5401:5b9:5400:2ff:fe18:dbdf	SSHv2	61496	22	Server: Encrypted packet
31	10.660192	2001:19f0:5401:2cf4:5400:2ff:fe18:eefb	2001:19f0:5401:5b9:5400:2ff:fe18:dbdf	SSHv2	61497	22	Server: Encrypted packet
33	10.661750	2001:19f0:5401:2cf4:5400:2ff:fe18:eefb	2001:19f0:5401:5b9:5400:2ff:fe18:dbdf	SSHv2	61498	22	Server: Encrypted packet
34	10.663759	2001:19f0:5401:2cf4:5400:2ff:fe18:eefb	2001:19f0:5401:5b9:5400:2ff:fe18:dbdf	SSHv2	61499	22	Server: Encrypted packet
36	10.665957	2001:19f0:5401:2cf4:5400:2ff:fe18:eefb	2001:19f0:5401:5b9:5400:2ff:fe18:dbdf	SSHv2	61500	22	Server: Encrypted packet
39	16.823873	2001:19f0:5401:2cf4:5400:2ff:fe18:eefb	2001:19f0:5401:5b9:5400:2ff:fe18:dbdf	SSHv2	61501	22	Server: Encrypted packet
42	16.960961	2001:19f0:5401:2cf4:5400:2ff:fe18:eefb	2001:19f0:5401:5b9:5400:2ff:fe18:dbdf	SSHv2	61502	22	Server: Encrypted packet
45	17.810947	2001:19f0:5401:2cf4:5400:2ff:fe18:eefb	2001:19f0:5401:5b9:5400:2ff:fe18:dbdf	SSHv2	61503	22	Server: Encrypted packet
46	17.812528	2001:19f0:5401:2cf4:5400:2ff:fe18:eefb	2001:19f0:5401:5b9:5400:2ff:fe18:dbdf	SSHv2	61504	22	Server: Encrypted packet
47	17.812672	2001:19f0:5401:2cf4:5400:2ff:fe18:eefb	2001:19f0:5401:5b9:5400:2ff:fe18:dbdf	SSHv2	61505	22	Server: Encrypted packet
51	23.333735	2001:19f0:5401:2cf4:5400:2ff:fe18:eefb	2001:19f0:5401:5b9:5400:2ff:fe18:dbdf	SSHv2	61506	22	Server: Encrypted packet
54	23.787777	2001:19f0:5401:2cf4:5400:2ff:fe18:eefb	2001:19f0:5401:5b9:5400:2ff:fe18:dbdf	SSHv2	61507	22	Server: Encrypted packet

The Other Way: Port 39535

pdmjune07-03-SSH-PDM-BothWays.pcap

File Edit View Go Capture Analyze Statistics Telephony Wireless Tools Help

tcp.srcport == 39535

No.	Time	Source	Destination	Protocol	PSN This Packe	Source Port	Info
2	3.714713	2001:19f0:5401:5b9:5400:2ff:fe18:dbdf	2001:19f0:5401:2cf4:5400:2ff:fe18:eefb	TCP	3628	39535	39535 → 22 [SYN] Seq=0 Wi
4	3.715063	2001:19f0:5401:5b9:5400:2ff:fe18:dbdf	2001:19f0:5401:2cf4:5400:2ff:fe18:eefb	TCP	3629	39535	39535 → 22 [ACK] Seq=1 Ack
5	3.715388	2001:19f0:5401:5b9:5400:2ff:fe18:dbdf	2001:19f0:5401:2cf4:5400:2ff:fe18:eefb	SSHv2	3630	39535	Client: Protocol (SSH-2.0-
7	3.723927	2001:19f0:5401:5b9:5400:2ff:fe18:dbdf	2001:19f0:5401:2cf4:5400:2ff:fe18:eefb	SSHv2	3631	39535	Client: Key Exchange Init
9	3.727324	2001:19f0:5401:5b9:5400:2ff:fe18:dbdf	2001:19f0:5401:2cf4:5400:2ff:fe18:eefb	SSHv2	3632	39535	Client: Elliptic Curve Di
11	3.838590	2001:19f0:5401:5b9:5400:2ff:fe18:dbdf	2001:19f0:5401:2cf4:5400:2ff:fe18:eefb	TCP	3633	39535	39535 → 22 [ACK] Seq=1423
12	5.865109	2001:19f0:5401:5b9:5400:2ff:fe18:dbdf	2001:19f0:5401:2cf4:5400:2ff:fe18:eefb	SSHv2	3634	39535	Client: New Keys
14	5.971264	2001:19f0:5401:5b9:5400:2ff:fe18:dbdf	2001:19f0:5401:2cf4:5400:2ff:fe18:eefb	SSHv2	3635	39535	Client: Encrypted packet
16	5.971667	2001:19f0:5401:5b9:5400:2ff:fe18:dbdf	2001:19f0:5401:2cf4:5400:2ff:fe18:eefb	SSHv2	3636	39535	Client: Encrypted packet
19	6.391986	2001:19f0:5401:5b9:5400:2ff:fe18:dbdf	2001:19f0:5401:2cf4:5400:2ff:fe18:eefb	SSHv2	3637	39535	Client: Encrypted packet
21	6.508606	2001:19f0:5401:5b9:5400:2ff:fe18:dbdf	2001:19f0:5401:2cf4:5400:2ff:fe18:eefb	TCP	3638	39535	39535 → 22 [ACK] Seq=1627
24	10.544339	2001:19f0:5401:5b9:5400:2ff:fe18:dbdf	2001:19f0:5401:2cf4:5400:2ff:fe18:eefb	SSHv2	3639	39535	Client: Encrypted packet
26	10.554459	2001:19f0:5401:5b9:5400:2ff:fe18:dbdf	2001:19f0:5401:2cf4:5400:2ff:fe18:eefb	SSHv2	3640	39535	Client: Encrypted packet
28	10.555755	2001:19f0:5401:5b9:5400:2ff:fe18:dbdf	2001:19f0:5401:2cf4:5400:2ff:fe18:eefb	SSHv2	3641	39535	Client: Encrypted packet
30	10.660142	2001:19f0:5401:5b9:5400:2ff:fe18:dbdf	2001:19f0:5401:2cf4:5400:2ff:fe18:eefb	TCP	3642	39535	39535 → 22 [ACK] Seq=1907
32	10.660661	2001:19f0:5401:5b9:5400:2ff:fe18:dbdf	2001:19f0:5401:2cf4:5400:2ff:fe18:eefb	SSHv2	3643	39535	Client: Encrypted packet
35	10.663960	2001:19f0:5401:5b9:5400:2ff:fe18:dbdf	2001:19f0:5401:2cf4:5400:2ff:fe18:eefb	TCP	3644	39535	39535 → 22 [ACK] Seq=2275
37	10.766893	2001:19f0:5401:5b9:5400:2ff:fe18:dbdf	2001:19f0:5401:2cf4:5400:2ff:fe18:eefb	TCP	3645	39535	39535 → 22 [ACK] Seq=2275
38	16.823628	2001:19f0:5401:5b9:5400:2ff:fe18:dbdf	2001:19f0:5401:2cf4:5400:2ff:fe18:eefb	SSHv2	3646	39535	Client: Encrypted packet
40	16.930580	2001:19f0:5401:5b9:5400:2ff:fe18:dbdf	2001:19f0:5401:2cf4:5400:2ff:fe18:eefb	TCP	3647	39535	39535 → 22 [ACK] Seq=2311
41	16.960719	2001:19f0:5401:5b9:5400:2ff:fe18:dbdf	2001:19f0:5401:2cf4:5400:2ff:fe18:eefb	SSHv2	3648	39535	Client: Encrypted packet
43	17.067619	2001:19f0:5401:5b9:5400:2ff:fe18:dbdf	2001:19f0:5401:2cf4:5400:2ff:fe18:eefb	TCP	3649	39535	39535 → 22 [ACK] Seq=2347
44	17.810761	2001:19f0:5401:5b9:5400:2ff:fe18:dbdf	2001:19f0:5401:2cf4:5400:2ff:fe18:eefb	SSHv2	3650	39535	Client: Encrypted packet

More Information in PDM

No.	Time	Source	Destination	Protocol	PSN This Packe	Source Port	Info
2	3.714713	2001:19f0:5401:5b9:5400:2ff:fe18:dbdf	2001:19f0:5401:2cf4:5400:2ff:fe18:eefb	TCP	3628	39535	39535 → 22 [SYN] Seq=0 Wi

TCP SYN Packet

```
> Frame 2: 110 bytes on wire (880 bits), 110 bytes captured (880 bits)
> Ethernet II, Src: fe:00:02:18:ee:fb (fe:00:02:18:ee:fb), Dst: 56:00:02:18:ee:fb (56:00:02:18:ee:fb)
v Internet Protocol Version 6, Src: 2001:19f0:5401:5b9:5400:2ff:fe18:dbdf, Dst: 2001:19f0:5401:2cf4:5400:2ff:fe18:eefb
    0110 .... = Version: 6
  > .... 0000 0000 .... .... .... .... .... = Traffic Class: 0x00 (DSCP: CS0, ECN: Not-ECT)
    .... .... .... 0011 0101 0010 1001 1111 = Flow Label: 0x3529f
    Payload Length: 56
    Next Header: Destination Options for IPv6 (60)
    Hop Limit: 61
    Source: 2001:19f0:5401:5b9:5400:2ff:fe18:dbdf
    Destination: 2001:19f0:5401:2cf4:5400:2ff:fe18:eefb
  v Destination Options for IPv6
      Next Header: TCP (6)
      Length: 1
      [Length: 16 bytes]
    v Performance and Diagnostic Metrics
      > Type: Performance and Diagnostic Metrics (0x0f)
        Length: 10
        Scale DTLR: 0
        Scale DTLS: 0
        PSN This Packet: 3628
        PSN Last Received: 0
        Delta Time Last Received: 0
        Delta Time Last Sent: 0
```

All fields initialized to zero.
Initial PSN set.

In Next Packet

No.	Time	Source	Destination	Protocol	PSN This Packe	Source Port	Info
3	3.714796	2001:19f0:5401:2cf4:5400:2ff:fe18:eefb	2001:19f0:5401:5b9:5400:2ff:fe18:dbdf	TCP	61485	22	22 → 39535 [SYN, ACK] Seq

```
> Frame 3: 110 bytes on wire (880 bits), 110 bytes captured (880 bits)
> Ethernet II, Src: 56:00:02:18:ee:fb (56:00:02:18:ee:fb), Dst: fe:00:02:18:ee:fb (fe:00:02:18:ee:fb)
v Internet Protocol Version 6, Src: 2001:19f0:5401:2cf4:5400:2ff:fe18:eefb, Dst: 2001:19f0:5401:5b9:5400:2ff:fe18:dbdf
    0110 .... = Version: 6
  > .... 0000 0001 .... .... .... .... .... = Traffic Class: 0x01 (DSCP: CS0, ECN: ECT(1))
    .... .... .... 0010 0110 1001 0001 0101 = Flow Label: 0x26915
    Payload Length: 56
    Next Header: Destination Options for IPv6 (60)
    Hop Limit: 64
    Source: 2001:19f0:5401:2cf4:5400:2ff:fe18:eefb
    Destination: 2001:19f0:5401:5b9:5400:2ff:fe18:dbdf
  v Destination Options for IPv6
      Next Header: TCP (6)
      Length: 1
      [Length: 16 bytes]
    v Performance and Diagnostic Metrics
      > Type: Performance and Diagnostic Metrics (0x0f)
        Length: 10
        Scale DTLR: 30
        Scale DTLS: 0
        PSN This Packet: 61485
        PSN Last Received: 3628
        Delta Time Last Received: 33210
        Delta Time Last Sent: 0
    v PadN
```

TCP SYN-ACK Packet

What is that Delta time?

Time difference from when packet 3628 was received to when packet 61425 is sent.

Application processing time

Add PSN Last Received Column

pdmjune07-03-SSH-PDM-BothWays.pcap

File Edit View Go Capture Analyze Statistics Telephony Wireless Tools Help

tcp | Expression...

No.	Time	Source	Destination	Protocol	PSN This Packe	PSN Last Received	Info
3	3.714796	2001:19f0:5401:2cf4:5400:2ff:fe18:eefb	2001:19f0:5401:5b9:5400:2ff:fe18:dbdf	TCP	61485	3628	22 → 39535 [SYN, ACK] Se
4	3.715063	2001:19f0:5401:5b9:5400:2ff:fe18:dbdf	2001:19f0:5401:2cf4:5400:2ff:fe18:eefb	TCP	3629	61485	39535 → 2
5	3.715388	2001:19f0:5401:5b9:5400:2ff:fe18:dbdf	2001:19f0:5401:2cf4:5400:2ff:fe18:eefb	SSHv2	3630	61485	Client: P
6	3.723501	2001:19f0:5401:2cf4:5400:2ff:fe18:eefb	2001:19f0:5401:5b9:5400:2ff:fe18:dbdf	SSHv2	61486	3630	S : P
7	3.723927	2001:19f0:5401:5b9:5400:2ff:fe18:dbdf	2001:19f0:5401:2cf4:5400:2ff:fe18:eefb	SSHv2	3631	61486	Client: K
8	3.725120	2001:19f0:5401:2cf4:5400:2ff:fe18:eefb	2001:19f0:5401:5b9:5400:2ff:fe18:dbdf	SSHv2	61487	3631	Server: K
9	3.727324	2001:19f0:5401:5b9:5400:2ff:fe18:dbdf	2001:19f0:5401:2cf4:5400:2ff:fe18:eefb	SSHv2	3632	61487	Client: E
10	3.732302	2001:19f0:5401:2cf4:5400:2ff:fe18:eefb	2001:19f0:5401:5b9:5400:2ff:fe18:dbdf	SSHv2	61488	3632	Server: E
11	3.838590	2001:19f0:5401:5b9:5400:2ff:fe18:dbdf	2001:19f0:5401:2cf4:5400:2ff:fe18:eefb	TCP	3633	61488	39535 → 2
12	5.865109	2001:19f0:5401:5b9:5400:2ff:fe18:dbdf	2001:19f0:5401:2cf4:5400:2ff:fe18:eefb	SSHv2	3634	61488	Client: N
13	5.970880	2001:19f0:5401:2cf4:5400:2ff:fe18:eefb	2001:19f0:5401:5b9:5400:2ff:fe18:dbdf	TCP	61489	3634	22 → 39535 [ACK] Seq=144
14	5.971264	2001:19f0:5401:5b9:5400:2ff:fe18:dbdf	2001:19f0:5401:2cf4:5400:2ff:fe18:eefb	SSHv2	3635	61489	Client: Encrypted packet
15	5.971391	2001:19f0:5401:2cf4:5400:2ff:fe18:eefb	2001:19f0:5401:5b9:5400:2ff:fe18:dbdf	SSHv2	61490	3635	Server: Encrypted packet
16	5.971667	2001:19f0:5401:5b9:5400:2ff:fe18:dbdf	2001:19f0:5401:2cf4:5400:2ff:fe18:eefb	SSHv2	3636	61490	Client: Encrypted packet
17	6.077869	2001:19f0:5401:2cf4:5400:2ff:fe18:eefb	2001:19f0:5401:5b9:5400:2ff:fe18:dbdf	TCP	61491	3636	22 → 39535 [ACK] Seq=148
18	6.391451	2001:19f0:5401:2cf4:5400:2ff:fe18:eefb	2001:19f0:5401:5b9:5400:2ff:fe18:dbdf	SSHv2	61492	3636	Server: Encrypted packet
19	6.391986	2001:19f0:5401:5b9:5400:2ff:fe18:dbdf	2001:19f0:5401:2cf4:5400:2ff:fe18:eefb	SSHv2	3637	61492	Client: Encrypted packet
20	6.402328	2001:19f0:5401:2cf4:5400:2ff:fe18:eefb	2001:19f0:5401:5b9:5400:2ff:fe18:dbdf	SSHv2	61493	3637	Server: Encrypted packet
21	6.508606	2001:19f0:5401:5b9:5400:2ff:fe18:dbdf	2001:19f0:5401:2cf4:5400:2ff:fe18:eefb	TCP	3638	61493	39535 → 22 [ACK] Seq=162
24	10.544339	2001:19f0:5401:5b9:5400:2ff:fe18:dbdf	2001:19f0:5401:2cf4:5400:2ff:fe18:eefb	SSHv2	3639	61493	Client: Encrypted packet
25	10.553858	2001:19f0:5401:2cf4:5400:2ff:fe18:eefb	2001:19f0:5401:5b9:5400:2ff:fe18:dbdf	SSHv2	61494	3639	Server: Encrypted packet
26	10.554459	2001:19f0:5401:5b9:5400:2ff:fe18:dbdf	2001:19f0:5401:2cf4:5400:2ff:fe18:eefb	SSHv2	3640	61494	Client: Encrypted packet
27	10.555114	2001:19f0:5401:2cf4:5400:2ff:fe18:eefb	2001:19f0:5401:5b9:5400:2ff:fe18:dbdf	SSHv2	61495	3640	Server: Encrypted packet

Can see that packets 3629 and 3630 both received

IPv6 Extension Headers Dropped

[Docs] [txt|pdf] [draft-ietf-v6op...] [Tracker] [Diff1] [Diff2] [Errata]

INFORMATIONAL
Errata Exist

Internet Engineering Task Force (IETF)
Request for Comments: 7872
Category: Informational
ISSN: 2070-1721

F. Gont
SI6 Networks / UTN-FRH
J. Linkova
Google
T. Chown
Jisc
W. Liu
Huawei Technologies
June 2016

Observations on the Dropping of Packets with IPv6 Extension Headers in the Real World

Abstract

This document presents real-world data regarding the extent to which packets with IPv6 Extension Headers (EHs) are dropped in the Internet (as originally measured in August 2014 and later in June 2015, with similar results) and where in the network such dropping occurs. The aforementioned results serve as a problem statement that is expected to trigger operational advice on the filtering of IPv6 packets carrying IPv6 EHs so that the situation improves over time. This document also explains how the results were obtained, such that the corresponding measurements can be reproduced by other members of the community and repeated over time to observe changes in the handling of packets with IPv6 EHs.

- Controversy at IETF
- Can IPv6 extension headers be used reliably & to what extent?

From RFC7282

Destination Options Header

Dataset	DO8	HBH8	FH512
Web servers	11.88% (17.60%/20.80%)	40.70% (31.43%/40.00%)	30.51% (5.08%/6.78%)
Mail servers	17.07% (6.35%/26.98%)	48.86% (40.50%/65.42%)	39.17% (2.91%/12.73%)
Name servers	15.37% (14.29%/33.46%)	43.25% (42.49%/72.07%)	38.55% (3.90%/13.96%)

Table 1: WIPv6LD Dataset: Packet Drop Rate for Different Destination Types, and Estimated (Best-Case / Worst-Case) Percentage of Packets That Were Dropped in a Different AS

NOTE: As an example, we note that the cell describing the support of IPv6 packets with DO8 for web servers (containing the value "11.88% (17.60%/20.80%)") should be read as: "when sending IPv6 packets with DO8 to public web servers, 11.88% of such packets get dropped. Among those packets that get dropped, 17.60%/20.80% (best case / worst case) of them get dropped at an AS other than the destination AS".

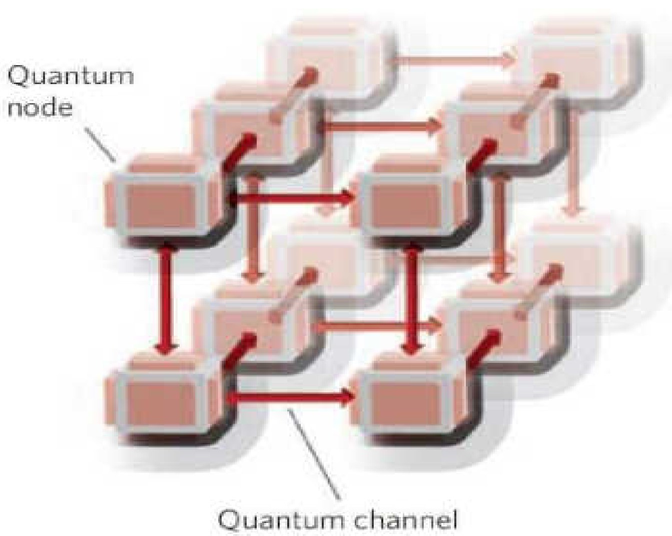
PDM Next Steps

- Currently installed on two Vultr virtual servers
- Expand to multiple
- Write new study
- Co-authors?
- Within enterprise study?
- Please contact me

Now, the future …

- Quantum networks!
- What the heck?
- Quantum Internet Research Group (QIRG) at IRTF https://datatracker.ietf.org/rg/qirg/documents/

Quantum networks: the vision



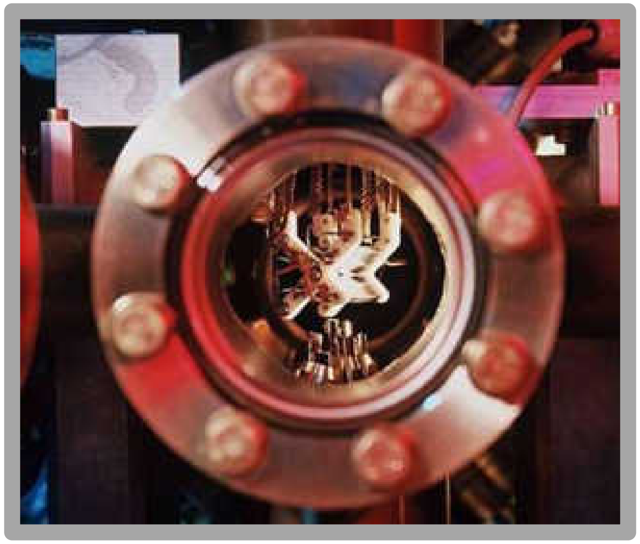
- Quantum nodes at which information is stored and processed.
  - » atoms
- Quantum channels for information transport.
  - » photons

https://datatracker.ietf.org/doc/slides-104-qirg-sessa-tutorial-on-quantum-repeaters/00/

H. J. Kimble, *Nature* 453, 1023 (2008)

Quantum Computing

- Quantum computers have a leg up over traditional computers when it comes to factoring.
- A classical computer uses bits of information, 1s and 0s. A quantum computer uses what are called qubits, which can be a mix of both 1 and 0 simultaneously and which exist in a delicate quantum state called superposition.

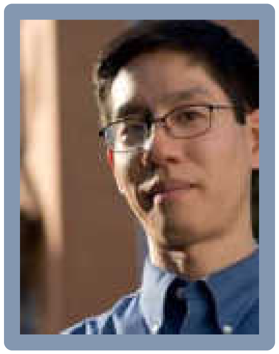


http://physicsworld.com/cws/article/news/2016/mar/04/shors-algorithm-is-implemented-using-five-trapped-ions

http://spectrum.ieee.org/tech-talk/computing/hardware/encryptionbusting-quantum-computer-practices-factoring-in-scalable-fiveatom-experiment

Shor's Algorithm

- Peter Shor, an MIT math professor, came up with an algorithm to factor large numbers with a quantum computer in 1994 but had no way to test it.
- In 2001,Isaac Chuang, an MIT physicist and electrical engineer, managed to use this algorithm to factor the number 15, but the quantum system he used could not be scaled up to factor anything more complicated.



http://spectrum.ieee.org/tech-talk/computing/hardware/encryptionbusting-quantum-computer-practices-factoring-in-scalable-fiveatom-experiment

Factoring Prime Numbers

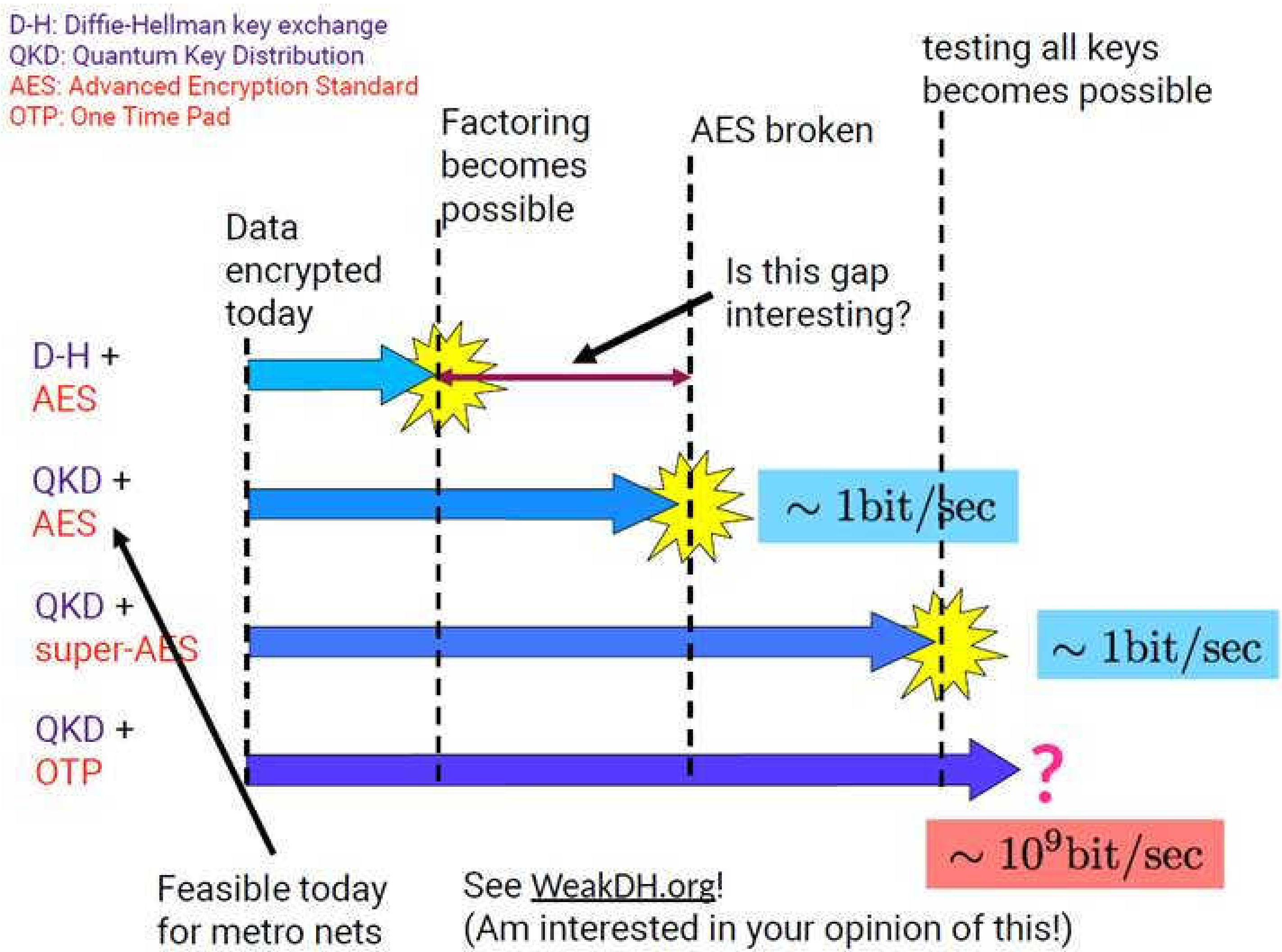
- A **prime number** (or a **prime**) has exactly two *distinct* divisors: 1 and itself.
- The smallest twenty-five prime numbers (all the prime numbers under 100) are:

  2, 3, 5, 7, 11, 13, 17, 19, 23, 29, 31, 37, 41, 43, 47, 53, 59, 61, 67, 71, 73, 79, 83,89, 97

- Prime number factorization is a list of all the prime-number factors of a given number.
- The prime factorization does not include 1, but does include every copy of every prime factor. For instance, the prime factorization of 8 is 2×2×2, not just "2". Yes, 2 is the only factor, but you need three copies of it to multiply back to 8, so the prime factorization includes all three copies

So what?

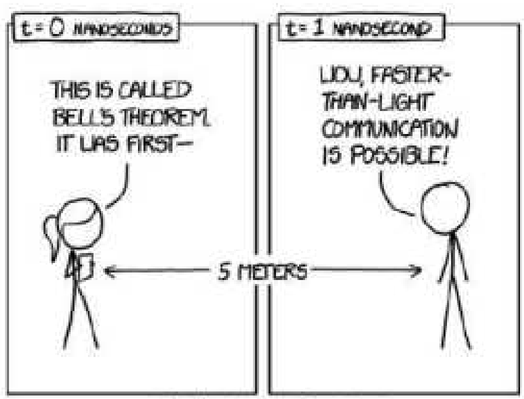
RSA which is one of the cryptographic algorithms we use today relies on the difficulty of prime number factorization



https://datatracker.ietf.org/doc/slides-104-qirg-sessa-tutorial-on-quantum-repeaters/00/

Good for & not good for

Quantum networks are about *new capabilities,* not some path to huge communication bandwidth. Reduced # of communication rounds (asymptotically, theoretically), higher precision, scalability of distributed quantum systems, etc.



BELL'S SECOND THEOREM: MISUNDERSTANDINGS OF BELL'S THEOREM HAPPEN SO FAST THAT THEY VIOLATE LOCALITY.

No faster-than-light communication!

You can each get shared, secret random numbers upon *measuring* shared, entangled states, but that doesn't give you the ability to send messages.

https://datatracker.ietf.org/doc/slides-104-qirg-sessa-tutorial-on-quantum-repeaters/00/

Entanglement （量子もつれ）

Even if they are
far apart!

"Measure" this
one and find its
value...

and you'll also
know what this
one is

https://datatracker.ietf.org/doc/slides-104-qirg-sessa-tutorial-on-quantum-repeaters/00/

Entanglement （量子もつれ）

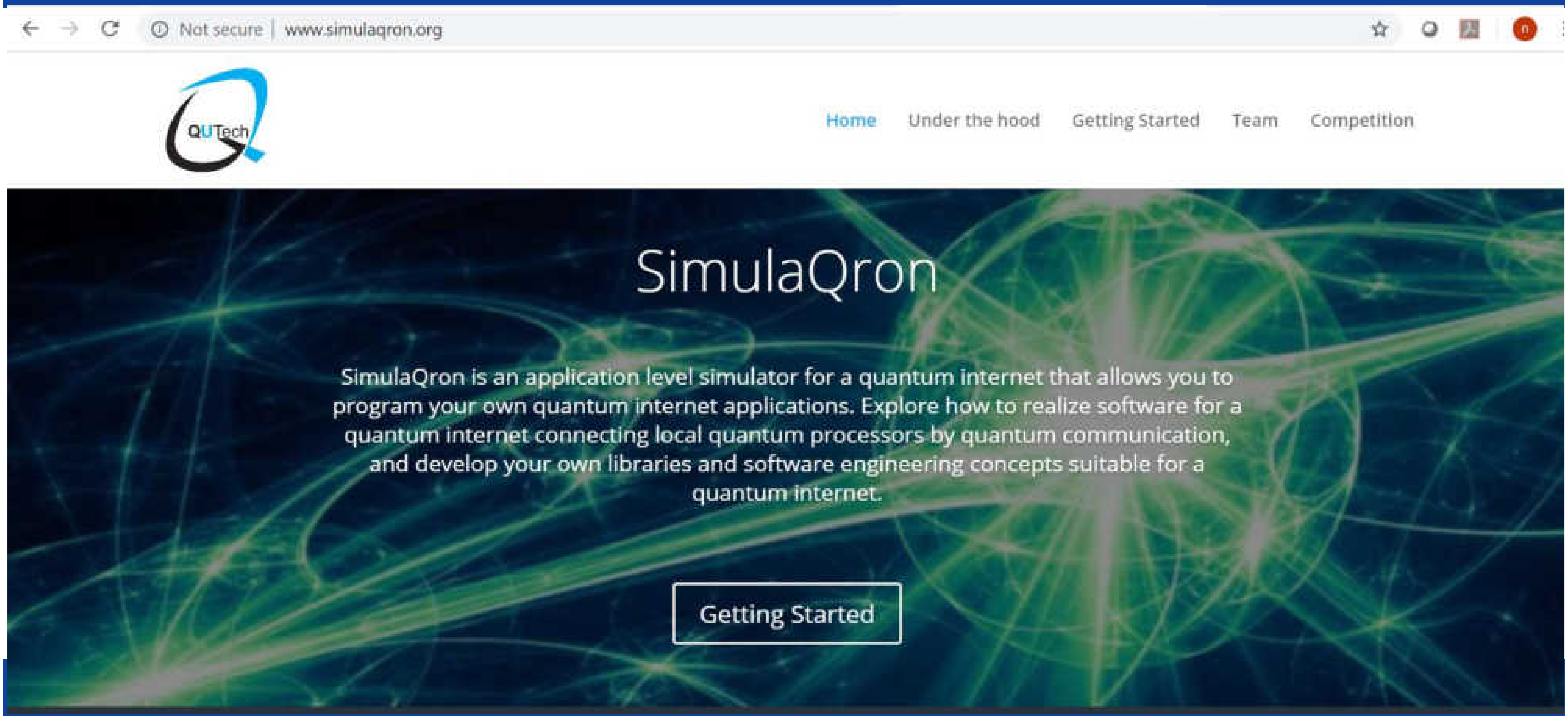
Even if they are
far apart!

"Measure" this
one and find its
value...

and you'll also
know what this
one is

https://datatracker.ietf.org/doc/slides-104-qirg-sessa-tutorial-on-quantum-repeaters/00/

SimulaQron


Not secure | www.simulaqron.org
QUTech
Home
Under the hood
Getting Started
Team
Competition
SimulaQron
SimulaQron is an application level simulator for a quantum internet that allows you to program your own quantum internet applications. Explore how to realize software for a quantum internet connecting local quantum processors by quantum communication, and develop your own libraries and software engineering concepts suitable for a quantum internet.
Getting Started

Trace

No.	Time	Source	Destination	Protocol	Length	Destination Port	Source Port	Info
1	0.000000	127.0.0.1	127.0.0.1	TCP	74	8016	35332	35332 → 8016 [SYN] Seq=0 Win=65495 Len=0 MSS=65495 SACK_PERM=1 TSv
2	0.000012	127.0.0.1	127.0.0.1	TCP	74	35332	8016	8016 → 35332 [SYN, ACK] Seq=0 Ack=1 Win=65483 Len=0 MSS=65495 SACK
3	0.000024	127.0.0.1	127.0.0.1	TCP	66	8016	35332	35332 → 8016 [ACK] Seq=1 Ack=1 Win=65536 Len=0 TSval=2861970108 TS
4	0.001798	127.0.0.1	127.0.0.1	TCP	74	8019	54766	54766 → 8019 [SYN] Seq=0 Win=65495 Len=0 MSS=65495 SACK_PERM=1 TSv
5	0.001810	127.0.0.1	127.0.0.1	TCP	74	54766	8019	8019 → 54766 [SYN, ACK] Seq=0 Ack=1 Win=65483 Len=0 MSS=65495 SACK
6	0.001823	127.0.0.1	127.0.0.1	TCP	66	8019	54766	54766 → 8019 [ACK] Seq=1 Ack=1 Win=65536 Len=0 TSval=2861970110 TS
7	0.002415	127.0.0.1	127.0.0.1	TCP	86	8016	35332	35332 → 8016 [PSH, ACK] Seq=1 Ack=1 Win=65536 Len=20 TSval=2861970
8	0.002423	127.0.0.1	127.0.0.1	TCP	66	35332	8016	8016 → 35332 [ACK] Seq=1 Ack=21 Win=65536 Len=0 TSval=2861970111 T
9	0.003048	127.0.0.1	127.0.0.1	TCP	149	8017	40008	40008 → 8017 [PSH, ACK] Seq=1 Ack=1 Win=512 Len=83 TSval=286197011
10	0.004338	127.0.0.1	127.0.0.1	TCP	74	8019	54766	54766 → 8019 [PSH, ACK] Seq=1 Ack=1 Win=65536 Len=8 TSval=28619701
11	0.004347	127.0.0.1	127.0.0.1	TCP	66	54766	8019	8019 → 54766 [ACK] Seq=1 Ack=9 Win=65536 Len=0 TSval=2861970113 TS
12	0.004372	127.0.0.1	127.0.0.1	TCP	70	8019	54766	54766 → 8019 [PSH, ACK] Seq=9 Ack=1 Win=65536 Len=4 TSval=28619701
13	0.004376	127.0.0.1	127.0.0.1	TCP	66	54766	8019	8019 → 54766 [ACK] Seq=1 Ack=13 Win=65536 Len=0 TSval=2861970113 T
14	0.004953	127.0.0.1	127.0.0.1	TCP	108	8020	41384	41384 → 8020 [PSH, ACK] Seq=1 Ack=1 Win=512 Len=42 TSval=286197011
15	0.005396	127.0.0.1	127.0.0.1	TCP	76	41384	8020	8020 → 41384 [PSH, ACK] Seq=1 Ack=43 Win=512 Len=10 TSval=28619701
16	0.005402	127.0.0.1	127.0.0.1	TCP	66	8020	41384	41384 → 8020 [ACK] Seq=43 Ack=11 Win=512 Len=0 TSval=2861970114 TS
17	0.006120	127.0.0.1	127.0.0.1	TCP	78	40008	8017	8017 → 40008 [PSH, ACK] Seq=1 Ack=84 Win=512 Len=12 TSval=28619701
18	0.006129	127.0.0.1	127.0.0.1	TCP	66	8017	40008	40008 → 8017 [ACK] Seq=84 Ack=13 Win=512 Len=0 TSval=2861970114 TS
19	0.006513	127.0.0.1	127.0.0.1	TCP	148	8017	40008	40008 → 8017 [PSH, ACK] Seq=84 Ack=13 Win=512 Len=82 TSval=2861970
20	0.007274	127.0.0.1	127.0.0.1	TCP	78	40008	8017	8017 → 40008 [PSH, ACK] Seq=13 Ack=166 Win=512 Len=12 TSval=286197
21	0.007591	127.0.0.1	127.0.0.1	TCP	93	8017	40008	40008 → 8017 [PSH, ACK] Seq=166 Ack=25 Win=512 Len=27 TSval=286197
22	0.008194	127.0.0.1	127.0.0.1	TCP	89	40008	8017	8017 → 40008 [PSH, ACK] Seq=25 Ack=193 Win=512 Len=23 TSval=286197
23	0.008459	127.0.0.1	127.0.0.1	TCP	101	8017	40008	40008 → 8017 [PSH, ACK] Seq=193 Ack=48 Win=512 Len=35 TSval=286197
24	0.010239	127.0.0.1	127.0.0.1	TCP	89	40008	8017	8017 → 40008 [PSH, ACK] Seq=48 Ack=228 Win=512 Len=23 TSval=286197
25	0.010548	127.0.0.1	127.0.0.1	TCP	98	8017	40008	40008 → 8017 [PSH, ACK] Seq=228 Ack=71 Win=512 Len=32 TSval=286197
26	0.010751	127.0.0.1	127.0.0.1	TCP	74	40008	8017	8017 → 40008 [PSH, ACK] Seq=71 Ack=260 Win=512 Len=8 TSval=2861970
27	0.011009	127.0.0.1	127.0.0.1	TCP	171	8017	40008	40008 → 8017 [PSH, ACK] Seq=260 Ack=79 Win=512 Len=105 TSval=28619

Packet with Payload

tcp.port == 35332

No.	Time	Source	Destination	Protocol	Length	Destination Port	Source Port	Info
7	0.002415	127.0.0.1	127.0.0.1	TCP	86	8016	35332	35332 → 8016 [PSH, ACK]

```
> Frame 7: 86 bytes on wire (688 bits), 86 bytes captured (688 bits)
> Ethernet II, Src: 00:00:00_00:00:00 (00:00:00:00:00:00), Dst: 00:00:00_00:00:00 (00:00:00:00:00:00)
> Internet Protocol Version 4, Src: 127.0.0.1, Dst: 127.0.0.1
> Transmission Control Protocol, Src Port: 35332, Dst Port: 8016, Seq: 1, Ack: 1, Len: 20
v Data (20 bytes)
    Data: 020100000000000c0000070500001f537f000001
    [Length: 20]
```

```
0000  00 00 00 00 00 00 00 00  00 00 00 00 08 00 45 00   ·········· ·····E·
0010  00 48 bd e7 40 00 40 06  7e c6 7f 00 00 01 7f 00   ·H··@·@· ~·······
0020  00 01 8a 04 1f 50 03 ad  09 6f 64 81 d7 f5 80 18   ·····P·· ·od·····
0030  02 00 fe 3c 00 00 01 01  08 0a aa 96 32 bf aa 96   ···<···· ····2···
0040  32 bc 02 01 00 00 00 00  00 0c 00 00 07 05 00 00   2······· ········
0050  1f 53 7f 00 00 01                                  ·S····
```

Manual Breakout

original Data portion: 020100000000000c0000070500001f537f000001

Breakout

02 : version	unsigned integer (uint8_t)	1 byte	Current version is 2
01 : type	unsigned integer (uint8_t)	1 byte	Message type : CQC_TP_COMMAND
0000 : app_id	unsigned integer (uint16_t)	2 bytes	Application ID,
0000000c : length	unsigned integer (uint32_t)	4 bytes	Total length of the CQC packet

CQC header

0000 : qubit_id	unsigned int (uint16_t)	2 bytes	Qubit ID to perform the operation on
07 : instr	unsigned int (uint8_t)	1 byte	Instruction to perform : CQC_CMD_EPR
05 : options	unsigned int (uint8_t)	1 byte	Options when executing the command

Looks fine so far…

- Lots of fields that don't make sense (not documented?)

- Working with Simulaqron people

Quantum Net Background

- https://www.youtube.com/watch?v=9nfaYAU92tY&feature=youtu.be

Contact Us!

- Nalini.Elkins@insidethestack.com

- Need to do
  - HTTP3 (prob. Sept. timeframe)
  - Quantum network dissector
  - PDM hackathon / draft / testing

- www.industrynetcouncil.org to join
  - Non-profit
  - Free (happy to take donations!)
  - May charge for labs (can put in sweat equity)

Appendix

- Additional PDM information
- Questions / answers from IETF92

Questions from IETF91
(Answered in IETF 92: See Appendix)

1. Does PDM have enough variables to actually diagnose problems?
2. Are all PDM fields necessary?
3. Why is the proposal for an IPv6 extension header rather than a TCP option?   Only TCP is important.
4. Does PDM create too much overhead?
5. Will PDM work for complex apps not just simple applications with one send and one receive?

Why IPv6 Extension Header?

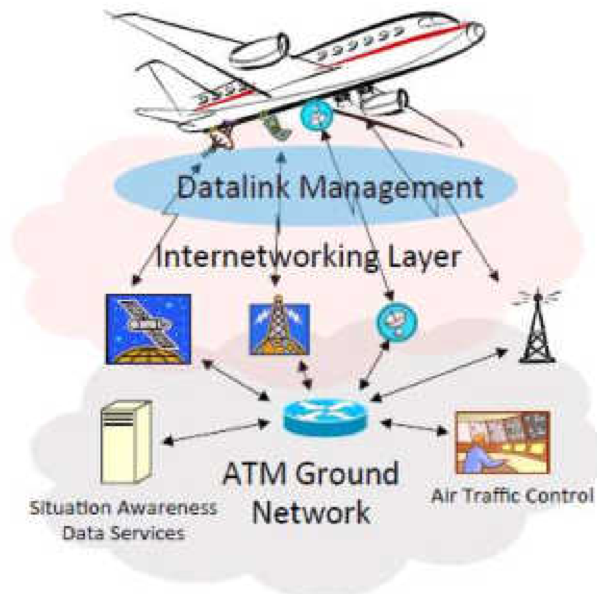
- Question:
  - Why is the proposal for an IPv6 extension header rather than a TCP option? Only TCP is important.

- Answer:
  - Large enterprises have traffic which is non-TCP which will benefit from PDM.
  - Non-TCP traffic includes:
    - IBM's Enterprise Extender, which allows its SNA Peer-to-Peer Networking (APPN) traffic flow over UDP links
    - Some WWW traffic flows as UDP packets
    - TFTP, SNMP, DNS, OSPF, etc.
    - Other/new upper layer protocols (e.g. the new Frame Control Protocol)
  - TCP applications will also benefit from PDM.

From Boeing

From IETF 91: IPv6GEO – GEO Information in IPv6 Packet Headers
http://www.ietf.org/proceedings/91/slides/slides-91-6man-8.pdf

- Aircraft have many links with varying cost, performance, availability profiles.
- Not all links available during all phases of flight.
- Not all links encode geo information at the link---layer
- Wide variety of applications – not all of which are geo---aware
- **IPv6 layer is only commonality**



Only for Simple Apps?

- Question
  - Will PDM work for complex apps not just simple applications with one send and one receive.
- Answer
  - Not at all.
  - Examples follow.

One-Way Flow

Packet	Sender	PSN This Packet	PSN Last Recvd	Delta Last Recvd	Delta Last Sent
1	Server	1	0	0	0
2	Server	2	0	0	5
3	Server	3	0	0	12
4	Server	4	0	0	20

In a one-way flow, only the Delta Last Sent will be seen as used. Recall, Delta Last Sent is the difference between the send of one packet from a device and the next. This is a measure of throughput for the sender - according to the sender's point of view. That is, it is a measure of how fast is the application itself (with stack time included) able to send packets.

How might this be useful? If one is having a performance issue at the client and sees that packet 2, for example, is sent after 5 microseconds from the server but takes much longer to arrive at the destination (deduced from other fields in the packet) then one may safely conclude that there are delays in the path other than at the server which may be causing the delivery issue of that packet. Such delays may include the network links, middle-boxes, etc.

Multiple Send Flow

Assume that two packets are sent with each send from the server.

Packet Sent	Sender	PSN This Packet	PSN Last Recvd	Delta Last Recvd	Delta Last Sent
1	Server	1	0	0	0
2	Server	2	0	0	5
3	Client	1	1	20	0
4	Server	3	1	10	15

Notice that in packet 3, the client has a value of Delta Last Received of 20. Recall that Delta Last Received is the Send time of packet 3 - receive time of packet 2. So, what does one know now? In this case, Delta Last Received is the processing time for the Client to send the next packet.

How to interpret depends on what is actually being sent. Remember, PDM is not being used in isolation, but to supplement the fields found in other headers.

Examples

- Client is sending a standalone TCP ACK. One would find this by looking at the payload length in the IPv6 header and the TCP Acknowledgement field in the TCP header. So, in this case, the client is taking 20 units to send back the ACK. This may or may not be interesting.

- Client is sending data with the packet. Again, one would find this by looking at the payload length in the IPv6 header and the TCP Acknowledgement field in the TCP header. So, in this case, the client is taking 20 units to send back data. This may represent "User Think Time". Again, this may or may not be interesting, in isolation. But, if there is a performance problem receiving data at the server, then taken in conjunction with RTT or other packet timing information, this information may be quite interesting.

Benefit of PDM

- Of course, one also needs to look at the PSN Last Received field to make sure of the interpretation of this data. That is, to make sure that the Delta Last Received corresponds to the packet of interest.

- The benefits of PDM are that we have such information available in a uniform manner for all applications and all protocols without extensive changes required to applications.

Multiple Send with Errors

- Are the functions of PDM better suited to TCP or a TCP option? Let us take the case of how PDM may help in a case of TCP retransmissions in a way that TCP options or TCP ACK / SEQ would not.

- Assume that three packets are sent with each send from the server.

- From the server, this is what is seen:

Pkt	Sender	PSN This Pkt	PSN LastRecvd	Delta LastRecvd	Delta LastSent	TCP SEQ	Data Bytes
1	Server	**1**	0	0	0	123	100
2	Server	**2**	0	0	5	223	100
3	Server	**3**	0	0	5	333	100

At Client

- The client however, does not get all the packets. From the client, this is what is seen for the packets sent from the server.

Pkt	Sender	PSN This Pkt	PSN LastRecvd	Delta LastRecvd	Delta LastSent	TCP SEQ	Data Bytes
1	Server	**1**	0	0	0	123	100
2	Server	**3**	0	0	5	333	100

- Notice that the packet with PSN = 2 from the server is not received

Server Retransmits

- Let's assume that the server now retransmits the packet. (Obviously, a duplicate acknowledgment sequence for fast retransmit or a retransmit timeout would occur. To illustrate the point, these packets are being left out.)

- So, then if a TCP retransmission is done, then from the client, this is what is seen for the packets sent from the server.

Pkt	Sender	PSN This Pkt	PSN LastRecvd	Delta LastRecvd	Delta LastSent	TCP SEQ	Data Bytes
1	Server	**4**	0	0	30	223	100

- The server has resent the old packet 2 with TCP sequence number of 223. The retransmitted packet now has a PSN This Packet value of 4.
- The Delta Last Sent is 30. That is the time between sending the packet with PSN of 3 and

Server Retransmits AGAIN

- Let's say that packet 4 STILL does not make it. Then, after some amount of time (RTO) then the packet with TCP sequence number of 223 is resent.

- From the client, this is what is seen for the packets sent from the server.

```
   Pkt Sender    PSN        PSN        Delta     Delta     TCP     Data
                This Pkt  LastRecvd  LastRecvd  LastSent   SEQ     Bytes
======================================================================
   1   Server    5          0          0         60                223
100
```