Layer 4¾
Fantastic quirks
and where to find them

ANRW @ IETF 114, July 27 2022

Lucas Pardue
Senior Engineer, Protocols Team, Cloudflare
Co-chair, IETF QUIC Working Group
Layer 4¾ - fantastic quirks and where to find them

Layer 7
Layer 6
Layer 5
Layer 4
Layer 3
Layer 2
Layer 1
The Cake is a Lie
Layer 4¾ - fantastic quirks and where to find them

Layer 1
Layer 2
Layer 3
Layer 4
Layer 5
Layer 6
Layer 7
Layer 4¾ - fantastic quirks and where to find them

To Swansea, West Wales, Penarth, Barry & Radyr

To Newport, Bristol, London & Valley Lines

https://upload.wikimedia.org/wikipedia/commons/9/99/Cardiff_Central_plan.png
CC-BY-SA-2.0
Layer 4¾ - fantastic quirks and where to find them

Layer 1
Layer 2
Layer 3
Layer 4
Layer 5
Layer 6
Layer 7
Layer 4¾ - fantastic quirks and where to find them

https://upload.wikimedia.org/wikipedia/commons/9/99/Cardiff_Central_plan.png
CC-BY-SA-2.0
Layer 4¾ - fantastic quirks and where to find them

To Swansea, West Wales, Penarth, Barry & Radyr

https://upload.wikimedia.org/wikipedia/commons/9/99/Cardiff_Central_plan.png
CC-BY-SA-2.0
Layer 4¾ - fantastic quirks and where to find them

To Swansea, West Wales, Penarth, Barry & Radyr

To Newport, Bristol, London & Valley Lines

https://upload.wikimedia.org/wikipedia/commons/9/99/Cardiff_Central_plan.png
CC-BY-SA-2.0
Layer 4¾ - fantastic quirks and where to find them

Layer 7
Layer 4
Layer 4¾ - fantastic quirks and where to find them

- QUIC
- HTTP
- HTTP/1.x
- HTTP/2
- HTTP/3
- TLS
- WebSocket
- gRPC
Layer 4¾ - fantastic quirks and where to find them

Spectrum

Overview

Protocols per plan

About
Get started
How to
Reference

Protocols per plan

On this table, you have information about which protocols are available per plan.

<table>
<thead>
<tr>
<th></th>
<th>Free</th>
<th>Pro</th>
<th>Business</th>
<th>Enterprise</th>
</tr>
</thead>
<tbody>
<tr>
<td>TCP</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>UDP</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Minecraft*</td>
<td>✔</td>
<td>✔</td>
<td></td>
<td>✔</td>
</tr>
<tr>
<td>SSH</td>
<td>✔</td>
<td>✔</td>
<td></td>
<td>✔</td>
</tr>
<tr>
<td>RDP</td>
<td></td>
<td></td>
<td></td>
<td>✔</td>
</tr>
</tbody>
</table>

*Minecraft Java Edition is supported but Minecraft Bedrock Edition is not supported.

https://developers.cloudflare.com/spectrum/protocols-per-plan/
(2022/04/27)
"Minecraft Java Edition is supported but Minecraft Bedrock Edition is not supported"

- https://wiki.vg/Protocol_version_numbers
- https://wiki.vg/Bedrock_Protocol
Layer 4¾ - fantastic quirks and where to find them

- https://github.com/aresrpg/minecraft-dissector
HTTP means Hypertext Transport Protocol?
Mark Nottingham at IETF 99:

"Roy Fielding knows whenever you call it Hypertext Transport Protocol."

https://httpwg.org/wg-materials/ietf99/minutes.html
The view from our edge
### HTTP vs. HTTPS
Breakdown of Internet traffic into HTTP and HTTPS over the selected time period. Learn about the differences between HTTP and HTTPS and why websites should use HTTPS.

<table>
<thead>
<tr>
<th></th>
<th>HTTP</th>
<th>HTTPS</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>1%</td>
<td>99%</td>
</tr>
</tbody>
</table>

### IPv4 vs. IPv6
Breakdown of IP addresses into IPv4 and IPv6 over the selected time period. Learn more about the differences between IPv4 and IPv6.

<table>
<thead>
<tr>
<th></th>
<th>IPv4</th>
<th>IPv6</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>74%</td>
<td>26%</td>
</tr>
</tbody>
</table>

### HTTP/1.x vs HTTP/2 vs HTTP/3
Learn how Cloudflare supports HTTP/2 and HTTP/3 to speed up your websites without requiring any changes to your existing codebase.

<table>
<thead>
<tr>
<th></th>
<th>HTTP/1.x</th>
<th>HTTP/2</th>
<th>HTTP/3</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>9%</td>
<td>68%</td>
<td>24%</td>
</tr>
</tbody>
</table>

### TLS1.2 vs TLS1.3 vs QUIC
Learn about the differences between TLS 1.3 and TLS 1.2 and the advantages of using the latest TLS version.

<table>
<thead>
<tr>
<th></th>
<th>TLS1.2</th>
<th>TLS1.3</th>
<th>QUIC</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>13%</td>
<td>63%</td>
<td>24%</td>
</tr>
</tbody>
</table>
Layer 4 ¾ - fantastic quirks and where to find them

<table>
<thead>
<tr>
<th></th>
<th>HTTP/1.1</th>
<th>HTTP/2</th>
<th>HTTP/3</th>
</tr>
</thead>
<tbody>
<tr>
<td>TCP</td>
<td>TCP</td>
<td>TCP</td>
<td>UDP</td>
</tr>
<tr>
<td>TLS record</td>
<td>TLS record</td>
<td>QUIC pkt</td>
<td>QUIC pkt</td>
</tr>
<tr>
<td>HTTP Request/Response</td>
<td>HTTP Request/Response</td>
<td>HTTP Request/Response</td>
<td></td>
</tr>
</tbody>
</table>

[Source](https://blog.cloudflare.com/a-primer-on-proxies/)
Layer 4¾ - fantastic quirks and where to find them

Worldwide Data

Trends

HTTP/3 (QUIC) traffic, in the selected time period, by country. Learn more about the advantages of HTTP/3 (QUIC).

<table>
<thead>
<tr>
<th>Country</th>
<th>Change</th>
<th>%</th>
</tr>
</thead>
<tbody>
<tr>
<td>Argentina</td>
<td>+1.60%</td>
<td>37.44%</td>
</tr>
<tr>
<td>Peru</td>
<td>+1.86%</td>
<td>36.55%</td>
</tr>
<tr>
<td>Nepal</td>
<td>+6.38%</td>
<td>36.24%</td>
</tr>
<tr>
<td>North Macedonia</td>
<td>+1.20%</td>
<td>36.01%</td>
</tr>
<tr>
<td>Belarus</td>
<td>+1.50%</td>
<td>35.76%</td>
</tr>
<tr>
<td>Moldova</td>
<td>+0.54%</td>
<td>35.61%</td>
</tr>
<tr>
<td>Serbia</td>
<td>+0.80%</td>
<td>35.50%</td>
</tr>
<tr>
<td>Tunisia</td>
<td>-1.58%</td>
<td>35.42%</td>
</tr>
<tr>
<td>Ecuador</td>
<td>+1.44%</td>
<td>35.19%</td>
</tr>
<tr>
<td>Guatemala</td>
<td>+1.34%</td>
<td>34.79%</td>
</tr>
</tbody>
</table>

Last 30 days of traffic, on 2022/04/27
Layer 4¾ - fantastic quirks and where to find them

- [https://blog.cloudflare.com/quic-version-1-is-live-on-cloudflare/](https://blog.cloudflare.com/quic-version-1-is-live-on-cloudflare/)
- Posted 2021/28/05

According to Cloudflare Radar, we’re seeing around 12% of Internet traffic using QUIC with HTTP/3 already. We look forward to this increasing now that RFC 9000 is out and raising awareness of the stability of things.

![HTTP/1.x vs HTTP/2 vs HTTP/3 Worldwide (Exclude bots / Last 7 days)](image)
HTTP RFCs have evolved: A Cloudflare view of HTTP usage trends

06/06/2022

Lucas Pardue  David Belson

https://blog.cloudflare.com/cloudflare-view-http3-usage/
Long-term trends - Global HTTP requests by version, all browsers
Long-term trends - Global HTTP requests by version, stacked %

Chrome

Firefox

Safari

HTTP/1.1
HTTP/2
HTTP/3
Long-term trends - Global HTTP/3 requests by browsers
Layer 4¾ - fantastic quirks and where to find them

Long-term trends - Global HTTP/3 requests by browsers, stacked %
Layer 4¾ - fantastic quirks and where to find them

Long-term trends - Global HTTP/3 requests by browsers (w/o Chrome)
Long-term trends - Global HTTP requests by version, bots

GoogleBot
- May 2021
- May 2022

BingBot
- May 2021
- May 2022

FacebookBot
- May 2021
- May 2022

LinkedInBot
- May 2021
- May 2022
Binary framing, the solution to all our HTTP problems?
Text-based HTTP has pedigree in providing quirks
Recent examples include "request smuggling"
- https://portswigger.net/web-security/request-smuggling
Despite the baggage, HTTP/1 is quite well exercised and understood by technically-minded folks
Yet, HTTP/2 and HTTP/3 constitute the majority of Web traffic
How do we characterize their quirks and performance?
Layer 4¾ - fantastic quirks and where to find them

- h2spec - [https://github.com/summerwind/h2spec](https://github.com/summerwind/h2spec)
- h3spec - [https://github.com/kazu-yamamoto/h3spec](https://github.com/kazu-yamamoto/h3spec)
- h2load - [https://nghttp2.org/documentation/h2load-howto.html](https://nghttp2.org/documentation/h2load-howto.html)
- netlog
- qlog?
- `<insert your favourite>`
- Useful tools, but in general our coverage over HTTP/2 or HTTP/3, as an Internet community, seems lacking
ERR_SPDY_PROTOCOL_ERROR
ERR_SPDY_PROTOCOL_ERROR

"Flush the SPDY pockets"

https://kinsta.com/knowledgebase/err_spdy_protocol_error/#method-4-flush-the-spdy-pockets
ERR_HTTP2_PROTOCOL_ERROR
ERR_HTTP2_PROTOCOL_ERROR

"All I had to do was turn my antivirus off and then on again haha"

ERR_QUIC_PROTOCOL_ERROR
ERR_QUIC_PROTOCOL_ERROR

"If you're still stuck after trying all these methods, contact Google customer support for help."

Layer 4¾ - fantastic quirks and where to find them

**Issue 1121658: QUIC & HTTP/3 Network Error Logging Granularity Parity**

Reported by lucas...@gmail.com on Tue, Aug 25, 2020, 6:38 PM GMT+1

UserAgent: Mozilla/5.0 (X11; Ubuntu; Linux x86_64; rv:79.0) Gecko/20100101 Firefox/79.0

Steps to reproduce the problem:
1. Open the source code at https://chromium.googlesource.com/chromium/src/+/refs/tags/B7.0.4244.2/net/network_error_logging
   /network_error_logging_service.cc#97
2. See there is one single entry for ERR_QUIC_PROTOCOL_ERROR that maps to h3.protocol.error
3. Maybe try and generate some QUIC or HTTP/3 errors and be flumuxed in your debugging using Network Error Logging because the level of detail is not sufficient

What is the expected behavior?
It would be nice for {QUIC & HTTP/3} NEL error types to have parity with {TCP, TLS & HTTP/2}. TLS errors are probably common, so this is really about surveying the TCP and HTTP/2 error codes and seeing what maps and devs would find useful.
The list is

Comment 4 by b...@chromium.org on Wed, Aug 26, 2020, 7:29 PM GMT+1

If I understand correctly, this is a fairly large project. It's not that Network Error Logging errors are not granular enough, but in fact the net errors are not granular enough. One would need to add a couple dozen new QUIC error codes to net/base/net_error_list.h (and a couple dozen more HTTP/2 error codes would also be very helpful), and emit them in the right places of the network stack.

This is definitely a worthwhile task, because these error codes are user visible, and would therefore give our users a lot more information.

I'll see what I can do in the near future.

Comment 5 by lucas...@gmail.com on Wed, Aug 26, 2020, 8:00 PM GMT+1

Improving HTTP/2 granularity would be greatly appreciated too.

https://bugs.chromium.org/p/chromium/issues/detail?id=1121658
"Lucas's Malformed"

- [https://quicwg.org/base-drafts/draft-ietf-quic-http.html#section-4.1.3](https://quicwg.org/base-drafts/draft-ietf-quic-http.html#section-4.1.3)
  - ... the absence of mandatory pseudo-header fields, ...

- Every request needs a method:
  - Malformed requests or responses that are detected **MUST** be treated as a stream error (Section 8) of type H3_MESSAGE_ERROR. (0x010E)
  - For malformed requests, a server **MAY** send an HTTP response indicating the error prior to closing or resetting the stream.
“Lucas's Malformed” - HTTP/3 server reaction to missing :method

<table>
<thead>
<tr>
<th>Server host</th>
<th>Result</th>
</tr>
</thead>
<tbody>
<tr>
<td>lucaspardue.com</td>
<td>400 Bad Request</td>
</tr>
<tr>
<td>google.com</td>
<td>400 Bad Request</td>
</tr>
<tr>
<td>facebook.com</td>
<td>400 Bad Request</td>
</tr>
<tr>
<td>ietf.akaquic.com</td>
<td>400 Bad Request</td>
</tr>
<tr>
<td>test.privateoctopus.com:4433</td>
<td>405 Method Not Allowed</td>
</tr>
<tr>
<td>mew.org</td>
<td>RESET_STREAM, 0x0102</td>
</tr>
<tr>
<td>msquic.net</td>
<td>RESET_STREAM, 0x0102</td>
</tr>
<tr>
<td>interop.seemann.io</td>
<td>RESET_STREAM, 0x0101</td>
</tr>
<tr>
<td>quic.aiortc.org</td>
<td>App CONNECTION_CLOSE, 0x010E</td>
</tr>
<tr>
<td>nghttp2.org</td>
<td>App CONNECTION_CLOSE, 0x010E</td>
</tr>
</tbody>
</table>
Layer 4¼ - fantastic quirks and where to find them

https://blog.cloudflare.com/on-the-recent-http-2-dos-attacks/

- **CVE-2019-9512**
  - Some HTTP/2 implementations are vulnerable to ping floods, potentially leading to a denial of service. The attacker sends continual pings to an HTTP/2 peer, causing the peer to build an internal queue of responses. Depending on how efficiently this data is queued, this can consume excess CPU, memory, or both.

https://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2019-9512
• gRPC
  • The idea is simple and powerful: every time a receiver gets a data frame it sends out a BDP ping (a ping with unique data only used by BDP estimator). After this, the receiver starts counting the number of bytes it receives (including the ones that triggered the BDP ping) until it receives the ack for that ping.

● Rust Hyper
  ● Problem
    ● HTTP/2 Adaptive Window sometimes triggers ENHANCE_YOUR_CALM
  ● Fix
    ● This introduces a delay to sending a ping to calculate the BDP that becomes shorter as the BDP is changing, to improve throughput quickly, but then also becomes longer as the BDP stabilizes, to reduce the amount of pings sent.

https://github.com/hyperium/hyper/issues/2526
https://github.com/hyperium/hyper/pull/2550
Layer 4¾ - fantastic quirks and where to find them

Speed Test

Your Internet Speed

Download 105 Mbps
Upload 21.4 Mbps
Ping 20.5 ms
Jitter 3.32 ms

https://speed.cloudflare.com
Layer 4¾ - fantastic quirks and where to find them

Difference between HTTP/1.1 and HTTP/2 upload throughput before changes

Autotuning improvement

What do we measure? Layer 4, layer 7, the whole thing?

- Layers below HTTP will affect its performance
- But they aren't indicative of upper layer success
  - See our IAB Measuring Network Quality submission [1]
- Latency and jitter are fundamental problems
  - See our FCC submission about Broadband Nutrition [2]
- draft-ietf-ippm-responsiveness
  - Responsiveness under Working Conditions
  - Testing the "whole stack", including HTTP/2

Professor Snape in The Order of the Phoenix:

“Well, it may have escaped your notice, but life isn’t fair.”
Http Datagrams and the Capsule Protocol
UDP 443 All the things
  - QUIC DATAGRAM frames
  - HTTP Datagrams
  - Capsule Protocol
  - Proxying UDP over HTTP aka MASQUE
  - WebTransport
  - Media over QUIC (MoQ)
  - ...
4 3\(\frac{3}{4}\) → Layer 7

Layer 4
Thank you

Lucas Pardue
Senior Engineer, Protocols Team, Cloudflare
Co-chair, IETF QUIC Working Group
@SimmerVigor