Debugging QUIC and HTTP/3

with [qlog] and <qvis>

Robin Marx, Maxime Piraux, Wim Lamotte and Peter Quax
Hasselt University and UCLouvain
Robin Marx
@programmingart

Last year PhD Student
Hasselt University, Belgium

Big J.R.R. Tolkien Fan
QUIC and HTTP/3 are quite extensive

6 “Core” specifications:
- QUIC invariants: 10 pages
- Core Transport: 187 pages
- TLS mapping: 59 pages
- Recovery (loss and congestion): 46 pages
- HTTP/3: 72 pages
- QPACK header compression: 44 pages

418 pages total
= 108 more than The Hobbit

Many other drafts/extensions:
- Applicability, manageability, DATAGRAM, load balancing, H3 priorities, ...
- Multipath, ACK frequency, loss bits, ...
packet captures

- Do not contain internal state
Common tool input format?

- **Packet captures**
  - Do not contain internal state

- **Ad-hoc endpoint logs**
  - Are different across implementations
  - Are unstructured

https://youtu.be/nErrFHPatq0?t=4339
https://youtu.be/LiNLz1QuT0s?t=3233
Common tool input format?

- Packet captures
  - Do not contain internal state

- Ad-hoc endpoint logs
  - Are different across implementations
  - Are unstructured

Structured endpoint logs

You can log what you want, just not how you want it

https://youtu.be/nErrFHPatq0?t=4339
https://youtu.be/LiNLz1QuT0s?t=3233
https://github.com/quiclog/internet-drafts
```json
{
  "connectionid": "0x763f8eaf61aa3ffe84270c0644bd8bd2b0d",
  "starttime": 1543917600,
  "fields": [
    "time",
    "category",
    "type",
    "trigger",
    "data"
  ],
  "events": [
    [50, "TLS", "RTT_KEY", "PACKET_RX", {"key": ...}],
    [51, "HTTP", "STREAM_OPEN", "PUSH", {"id": 0, "headers": ...}],
    ...,
    [200, "TRANSPORT", "PACKET_RX", "STREAM", {"nr": 50, "contents": "GET /ping.html"}],
    [201, "HTTP", "STREAM_OPEN", "GET", {"id": 16, "headers": ...}],
    [201, "TRANSPORT", "STREAMFRAME_NEW", "PACKET_RX", {"id": 16, "contents": "pong", ...}],
    [203, "RECOVERY", "PACKETSpawnED", "CWND_EXCEEDED", {"nr": 67, "frames": [16, ...]}],
    [250, "TRANSPORT", "ACK_NEW", "ACK_PKT", {"nr": 67, "acked": 60, ...}],
    [251, "RECOVERY", "CWND_UPDATE", "ACK_PKT", {"nr": 51, "cwnd": 14600, ...}],
    [252, "TRANSPORT", "PACKET_TX", "CWND_UPDATE", {"nr": 67, "frames": [16, ...]}],
    ...,
    [1001, "RECOVERY", "LOSS_DETECTED", "ACK_PKT", {"nr": a, "frames": ...}],
    [2002, "RECOVERY", "PACKET_NEW", "EARLY_RETRANS", {"nr": x, "frames": ...}],
    [3003, "RECOVERY", "PACKET_NEW", "TAIL_LOSS_PROBE", {"nr": y, "frames": ...}],
    [4004, "RECOVERY", "PACKET_NEW", "TIMEOUT", {"nr": z, "frames": ...}],
  ]
}
```
2 years later...

12/18 QUIC implementations support qlog

- Facebook, Cloudflare, Mozilla, NodeJS (ngtcp2), ...
- 2 more with plans to add qlog in the future
- 2 others use a (different) structured format

Facebook has deployed it in production

- Store over 30 billion qlog events daily

https://crates.io/crates/qlog
https://github.com/quicwg/base-drafts/wiki/Implementations
But... why?

Expert survey
- Recruited via QUIC mailing list (and gentle prodding)
- 28 participants
- at least 1 participant from all but 2 of the 18 implementations
- All QUIC developers (22) and researchers (6)
+ in-depth interview with Facebook

Debugging and analysis for QUIC in general
- Which types of logging and why?
- Which tools and why?
- Which (future) use cases?

https://qlog.edm.uhasselt.be/anrw
They like qlog because:

1. They want to use 3rd party tools (like `<qvis>`)
2. It makes it easy to create custom tools
3. qlog is flexible

They don’t like qlog because:

4. JSON is verbose and slow
The toolsuite can be found online at:

- https://qvis.edm.uhasselt.be

Example traces can be found at:

- https://qlog.edm.uhasselt.be/anrw
- https://qlog.edm.uhasselt.be/sigcomm
They like qlog because:

1. They want to use 3rd party tools (like <qvis>)
2. It makes it easy to create custom tools
3. qlog is flexible and schemaless

They don’t like qlog because:

4. JSON is verbose and slow
qlog is flexible : 1/3

qlog defines events and fields
- But most are optional
- And other events are explicitly allowed

Used extensively in practice
- Implementation-specific state (e.g., BBR parameters)
- New QUIC extensions (Mulitpath, DATAGRAM, Ack Frequency, loss bits, ...)
- 1 implementation completely switched from ad-hoc to qlog

→ No need to wait for a qlog or qvis update to visualize new things
qlog is flexible : 2/3

Easy to use and parse

- Facebook streams individual events to a database
  - Later uses queries to find interesting traces (e.g., % of packet_lost events)
- Log-based unit testing
  - "Was the spin-bit spinning?" → are there qlog spin_bit_updated events?

```cpp
std::vector<int> indices = getQLogEventIndices(QLogEventType::AppLimitedUpdate, qLogger);
EXPECT_EQ(indices.size(), 2);

auto event = qLogger->logs[indices[0]];
EXPECT_EQ(event->limited, true);

auto event2 = qLogger->logs[indices[1]];
EXPECT_EQ(event2->limited, false);
```

https://github.com/aiortc/aioquic
https://github.com/facebookincubator/mvfst
qlog is flexible : 3/3

Easy to transform from/to other formats
- pcap2qlog, netlog2qlog, quictrace2qlog, etc.

Easy to extend to other protocols
- DNS over QUIC, DNS over HTTP/3
- TCP + TLS + HTTP/2
  → combine pcaps with eBPF kernel probes and H2 browser logs

https://github.com/quiclog/pcap2qlog
https://github.com/quiclog/quictrace2qlog
https://github.com/moonfalir/quicSim-docker/tree/master/tcpebpf
https://github.com/triplewy/qvis/tree/master/visualizations/src/components/filemanager/netlogconverter
They like qlog because:

1. They want to use 3\textsuperscript{rd} party tools (like <qvis>)
2. It makes it easy to create custom tools
3. qlog is flexible and schemaless

They don’t like qlog because:

4. JSON is verbose and slow
The IETF QUIC Timeline

Debug Implementation

Fine-tune Deployment

Wide-spread Adoption

Debug Deployment

Teaching And Research

Logging needs to run at scale

Google creates HIP (HTTP over IP)
Connection tracing at scale?

packet captures

- Are large because QUIC is encrypted
- Privacy and security concerns
Connection tracing at scale?

**packet captures**
- Are large because QUIC is encrypted
- Privacy and security concerns

**spin and loss bits**
- The nays have it?
- Would still be fairly limited
Connection tracing at scale?

- Packet captures are large because QUIC is encrypted.
- Privacy and security concerns.
- Spin and lose bits.
- The nays have it?
- Would still be fairly limited.

Structured endpoint logs

Log only what you need.
JSON does not scale

Binary format would be better

- Counter-argument: much less flexible!
- (Semi) Counter-argument: Facebook uses qlog in production
- Counter-argument: JSON compresses well

https://github.com/quiclog/internet-drafts/issues/30
JSON does not scale

Binary format would be better

- Counter-argument: much less flexible!
- (Semi) Counter-argument: Facebook uses qlog in production
- Counter-argument: JSON compresses well

<table>
<thead>
<tr>
<th>format</th>
<th>raw</th>
</tr>
</thead>
<tbody>
<tr>
<td>pcap</td>
<td>561.57</td>
</tr>
<tr>
<td>JSON</td>
<td>276.02</td>
</tr>
<tr>
<td>CBOR</td>
<td>215.53</td>
</tr>
<tr>
<td>protobuf</td>
<td>66.15</td>
</tr>
</tbody>
</table>

500 MB file download

resulting log file sizes in MB

https://github.com/quiclog/internet-drafts/issues/30
JSON does not scale

Binary format would be better

- Counter-argument: much less flexible!
- (Semi) Counter-argument: Facebook uses qlog in production
- Counter-argument: JSON compresses well

<table>
<thead>
<tr>
<th>format</th>
<th>raw</th>
<th>gzip6</th>
<th>brotli4</th>
</tr>
</thead>
<tbody>
<tr>
<td>pcap</td>
<td>561.57</td>
<td>529.01</td>
<td>528.85</td>
</tr>
<tr>
<td>JSON</td>
<td>276.02</td>
<td>19.15</td>
<td>19.40</td>
</tr>
<tr>
<td>CBOR</td>
<td>215.53</td>
<td>17.78</td>
<td>18.90</td>
</tr>
<tr>
<td>protobuf</td>
<td>66.15</td>
<td>14.56</td>
<td>10.71</td>
</tr>
</tbody>
</table>

500 MB file download

resulting log file sizes in MB

https://github.com/quiclog/internet-drafts/issues/30
Solution: Pick your poison

qlog is a loose schema, implementers choose the format

- JSON is the default
- But updated definitions to make it easier to define a binary setup
  - Binary to JSON (e.g., for tooling) should be easy

Will be in qlog draft-02 (this week or next)
- Will need additional evaluation over time

https://github.com/quiclog/internet-drafts/issues/30
In conclusion

Tooling has really helped in debugging QUIC

(we even got people to output raw JSON...)

Structured logging can be the way to go for wider deployment

(but more work needed to determine scaling requirements)
Future work + why IETF?

Can qlog solve the spinbit use case for network operators?

- Endpoint owners sharing logs? How to scale and automate that?
- Similar concepts discussed in IPPM right after this!

How do we define privacy and security guidelines?

- Which fields should we strip? Anonymize?

Should this be bigger than just QUIC and HTTP/3?

robin.marx@uhasselt.be

https://huitema.wordpress.com/2020/07/21/scrubbing-quic-logs-for-privacy/
Image sources

- https://img.icons8.com/cotton/2x/survey.png
- https://cdn4.vectorstock.com/i/1000x1000/20/13/thumb-up-and-down-icon-vector-20072013.jpg