SAME SPORT, DIFFERENT STRATEGIES: a study of QUIC and HTTP/3 game plan diversity

ROBIN MARX, JORIS HERBOTS, WIM LAMOTTE AND PETER QUAX

HASSELT UNIVERSITY - EDM

https://qlog.edm.uhasselt.be/epiq
TCP IS OUT, QUIC IS IN!

MORE OPTIONS FOR INDIVIDUAL STACKS TO TWEAK BEHAVIOUR

- Stream multiplexing
- User space congestion control
- 0-RTT
- Binary framing
QUIC EVOLUTION
QUIC EVOLUTION

And several others!

https://github.com/quicwg/base-drafts/wiki/Implementations
METHODOLOGY

ARE YOU WATCHING CLOSELY?
METHODOLOGY

structured endpoint logs

interactive tooling

[ qlog ]

< qvis > manual analysis

METHODOLOGY

QUIC interop runner
- Automated tests, run daily
- Client-side behaviour

[ ] qlog

Custom aioquic client
- Point at public interop endpoints
- Server-side behaviour

METHODOLOGY

QUIC interop runner
- Automated tests, run daily
- Client-side behaviour

Custom aioquic client
- Point at public interop endpoints
- Server-side behaviour

Verify results

1. Source code review
2. Ask the original implementers

http://quicdev.slack.com
METHODOLOGY

18 teams

15 analyzed

At least 9 per test
FLOW CONTROL

DON’T MAKE THAT BUFFER SUFFER
FLOW CONTROL

TCP 2020

Single connection-level buffer
RECEIVE WINDOW

1. Connection-level limit
2. Stream-level limits
3. Stream count limit

{}  increase often
OR
sending can STALL

QUIC LEAGUE 2020
3 MAIN FC APPROACHES

1. static: 5000 received, you get 5000 more
2. dynamic: 5000 received, you get 10000 more
3. autotune: fluctuate based on RTT/application behavior

![Graph showing static and dynamic FC allowance](image)
## FLOW CONTROL

<table>
<thead>
<tr>
<th>Flow control approach</th>
<th>Adoption</th>
</tr>
</thead>
<tbody>
<tr>
<td>static</td>
<td>8/12</td>
</tr>
<tr>
<td>growing</td>
<td>3/12</td>
</tr>
<tr>
<td>autotune</td>
<td>1/12</td>
</tr>
</tbody>
</table>

“We have not yet spent time fine-tuning or testing Flow Control”
Flow control limit updates in small increments

Bytes in flight never reach $cwnd$
MULTIPLEXING

THE SPICE MUST FLOW
STREAM BANDWIDTH DISTRIBUTION

Stream 1

Stream 2

just 1 QUIC connection
STREAM BANDWIDTH DISTRIBUTION

Stream 1

Stream 2

Sequential OR Round-Robin
VARIETY IS THE SPICE OF LIFE

Round-Robin

The weird ones in the middle

Sequential

https://h3.edm.uhasselt.be
VARIETY IS THE SPICE OF LIFE

- Server Push
- Header Compression
- Prioritization

https://h3.edm.uhasselt.be
### QUIC MULTIPLEXING

<table>
<thead>
<tr>
<th>Multiplexer</th>
<th>Adoption</th>
</tr>
</thead>
<tbody>
<tr>
<td>Round-Robin</td>
<td>9/13</td>
</tr>
<tr>
<td>Sequential</td>
<td>4/13</td>
</tr>
<tr>
<td>(experimental) HTTP/3 prioritization</td>
<td>5/18</td>
</tr>
</tbody>
</table>

“waiting for HTTP/3 prioritization to fine-tune”
RETRANSMISSION SCHEDULING

TCP
2U2
0P0

send order

retransmission order
RETRANSMISSION SCHEDULING

TCP 2020

send order

retransmission order

QUIC LEAGUE 2020

send order

retransmission order
RETRANSMISSION SCHEDULING

1. No special treatment
2. Highest priority, default scheduler
3. Highest priority, different scheduler
4. (HTTP/3) Prioritization-driven

Example for nr. 3:
# Retransmission Scheduling

<table>
<thead>
<tr>
<th>Retransmission approach</th>
<th>Adoption</th>
</tr>
</thead>
<tbody>
<tr>
<td>1. All data is equal</td>
<td>2/13</td>
</tr>
<tr>
<td>2. TCP-alike</td>
<td>9/13</td>
</tr>
<tr>
<td>3. TCP-alike, change scheduler</td>
<td>1/13</td>
</tr>
</tbody>
</table>

“Unclear which performs best/if it matters”
CONGESTION CONTROL

THE NEED FOR SPEED
CONGESTION CONTROL (MOST INACCURATE TIMELINE EVER)

- TAHOE
- (NEW) RENO
- (CU)BIC
- BBR

NEW RENO

?????????????

congestion response

congestion response
CONGESTION CONTROL (MOST INACCURATE TIMELINE EVER)

TCP 2U2U2P0

TAHOE (NEW) RENO (CU)BIC BBR

SACK ECN TAIL LOSS PROBE RACK

congestion response
congestion/loss detection

FEEL FREE TO ROLL YOUR OWN

NEW RENO

congestion response
congestion/loss detection

ALL THE GOOD STUFF

QUIC LEAGUE 2020
**CONGESTION CONTROL**

<table>
<thead>
<tr>
<th>Congestion controller</th>
<th>Adoption</th>
</tr>
</thead>
<tbody>
<tr>
<td>New Reno</td>
<td>9/15</td>
</tr>
<tr>
<td>CUBIC</td>
<td>6/15</td>
</tr>
<tr>
<td>(with hystart)</td>
<td>4/6</td>
</tr>
<tr>
<td>BBR v1</td>
<td>4/15</td>
</tr>
<tr>
<td>BBR v2, COPA, ...</td>
<td>3/15</td>
</tr>
</tbody>
</table>

“Often too complex to implement a new one”
congestion := congestion.NewCubicSender(
    congestion.DefaultClock{},
    rttStats,
    true, // use Reno
    tracer,
)
THE DEVIL IS IN THE DETAILS

<table>
<thead>
<tr>
<th>Initial congestion window</th>
<th>Adoption</th>
</tr>
</thead>
<tbody>
<tr>
<td>12-15 kB</td>
<td>11/14</td>
</tr>
<tr>
<td>40+ kB</td>
<td>3/14</td>
</tr>
<tr>
<td>smart tweaking</td>
<td>2/14</td>
</tr>
</tbody>
</table>

“We just looked at what Google was doing”

<table>
<thead>
<tr>
<th>Pacing</th>
<th>Adoption</th>
</tr>
</thead>
<tbody>
<tr>
<td>Yes</td>
<td>8/15</td>
</tr>
<tr>
<td>No</td>
<td>7/15</td>
</tr>
</tbody>
</table>

“Complex to get right”
THE DEVIL IS IN THE DETAILS

<table>
<thead>
<tr>
<th>ACK every X packets</th>
<th>Adoption</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>2/12</td>
</tr>
<tr>
<td>1 - 38</td>
<td>10/12</td>
</tr>
<tr>
<td>ACK frequency extension</td>
<td>4/12</td>
</tr>
</tbody>
</table>

“Read from socket in large batches, ACK per batch”

“Lower ACK frequencies are better on constrained networks”

0-RTT

OFF-BY-ONE ERRORS ARE THE BEST
ROUND TRIPS ARE THE WORST

1. SYN/ACK
2. TLS
3. (TLS)
4. HTTP

1. QUIC + TLS
2. HTTP

1. QUIC + TLS + HTTP
SOCCER HAS OFFSIDE, QUIC HAS 0-RTT

**QUIC 0-RTT:**

1. Needs to be encrypted
   → Only from second connection (session ticket)

2. Runs over IP + UDP
   → Send max $3X$ as much as received (amplification limit)

3. Transports HTTP
   → Only idempotent requests
MAKING 0-RTT BETTER

1-RTT

Session ticket enables 0-RTT for the next connection
MAKING 0-RTT BETTER

1-RTT

0-RTT

Server can only send 3X what it received (say 4-5 kB)
MAKING 0-RTT BETTER

1-RTT

0-RTT

0-RTT with padding

Client sends more, server can send more
MAKING 0-RTT BETTER

1-RTT

0-RTT

0-RTT with padding

0-RTT with address token

Address validation token

Send up to initial cwnd
# The Optimization Formerly Known as Crucial

## 0-RTT Support

<table>
<thead>
<tr>
<th>Adoption</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Yes</td>
<td>13/18</td>
</tr>
<tr>
<td>No</td>
<td>5/18</td>
</tr>
</tbody>
</table>

“TLS library doesn’t support it yet”

## Optimizations

<table>
<thead>
<tr>
<th>Adoption</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Extra PADDING</td>
<td>0/9</td>
</tr>
<tr>
<td>NEW_TOKEN</td>
<td>7/13</td>
</tr>
</tbody>
</table>

Amplification bugs | 4/9
SET THE LASERS TO AMPLIFICATION

1. Ignore limit, have a 46kB init cwnd  
   = 36X amplification

2. Do not apply limit to retransmissions of 0-RTT data  
   = 17X amplification

3. Do not apply congestion control to 0-RTT data  
   = 300kB burst if client sends 100kB
FRAMECEPTION

HTTP/3 frames are packet inside QUIC frames which are packet inside QUIC packets, which are sized after MTU discovery

→ Efficiency + Head-of-Line blocking
CONCLUSION

IT’S JUST A GAME
LOOK BEFORE YOU LEAP

1. QUIC is complex: many knobs to turn, easy to make it slower/faster

2. QUIC and HTTP/3 implementations aren’t finished: don’t trust, always verify
   
   QUIC 1 != QUIC 2 != QUIC 3 != QUIC 4
   test different implementations

3. You might want to look at our methodology

[qlog]  <qvis>

IMAGE SOURCES

CREDITS & COPYRIGHTS

• Template created by Showeet.com

• **Free font used:**
  • Calibri (Microsoft font)

• **Copyrights:**
  • Free with Attribution
  • Cannot be resold or redistributed under any circumstances
  • Cf. [https://www.showeet.com/terms-of-use/](https://www.showeet.com/terms-of-use/)
THANK YOU!

Hope you like this template :)
Free creative templates, charts, diagrams and maps for your outstanding presentations

https://www.showeet.com/