Deploying QUIC at Scale
(at Google)

Ian Swett, Martin Duke
Agenda

1. QUIC Load Balancing
2. QUIC blackholing
3. A QUIC outage
4. 0-RTT in IETF QUIC
Load Balancing
A QUIC plug for QUIC-LB
Anycast
A single IP shared using BGP for load balancing
Perfect Linkability
Perfect Unlinkability
One Connection ID format

- 2 Config ID bits
- Server ID (1..15 opaque bytes)
- Nonce (4..18 opaque bytes)
- Connection ID length (optional) or 6 random bits
- Config id ‘11’ = unroutable
- Could be encrypted or not – algorithm depends on length
QUIC Blackholing
(post-handshake)
5-tuple blackholing

A 5-tuple can be blackholed, even if most 5-tuples between two endpoints work

  Maybe it traverses a broken piece of hardware?
  Maybe a machine has a bad line card?
  Maybe the internet is a terrible broken place?

Blackholing can cause QUIC to wait for idle timeout, 30s-minutes
What we’ve done to mitigate it

To reduce the time to connection failure,
  close the connection after consecutive (5) PTOs
Reduces tail latency substantially
Probably closes a few ‘good’ connections, unfortunately
Requests still fail, but many can be retried by the browser or app
We do this on the server or client,
  though it’s unclear why it helps so much on the server side?
A QUIC Solution

**Observation:** Changing only port can drastically change the path
  ie: entirely different datacenters or peering points.

Try a new client ephemeral port!
- Introduces entropy in both directions, direction doesn’t matter
- No need for privileged access

Default enabled in Chromium (ie: Chrome, Cronet, …)
QUIC Exit and Contagion Bugs
A short summary of FB Reliability@Scale (Recording, Slides)
Summary

**Query of death** triggered by resumption information sent from GFEs to clients and back to GFEs caused GFEs to crash.

At peak around **10% of Google GFEs** were crashing, but this distribution was very uneven.

Impact was mostly limited to Europe, and to services served from datacenters.

Total outage time was **1h 44m**.
Contagion: An interaction of distributed systems

Slow rollouts identify most bugs before significant harm
If a bug is found, roll back.

Contagion bugs are not fixed by rollbacks alone.

A single task could cause a global outage.
Persistent state in another system is not rolled back.
In the case of internet clients, cannot rollback.
Example: TLS or QUIC Resumption

- TLS resumption
- QUIC source address tokens
- gQUIC server configs

One GFE gives the client information for a future connection
Another GFE parses it later and something goes wrong.
What happened at Google in November 2021
Mechanism of action

- DC1: canary GFE
  - QUIC handshake to "serve.google.com."
- DC2: prod GFE
  - The browser completes a handshake to a GFE over IETF QUIC.
- DC3: prod GFE
Mechanism of action

“By the way, here’s a token you can use to prove you connected from your IP before. You can send it on your next request.”

The GFE sends the browser an encrypted token, which proves a client owns a specific IP address, limiting amplification attacks.

All GFEs send this token, but canary jobs populated a new field.
Mechanism of action

DC1
- canary GFE

DC2
- prod GFE

DC3
- prod GFE

User's browser

“QUIC handshake to serve.google.com. By the way, I have this token that proves I own this source IP.”

The token is sent by the client on the next connection; after a handshake, the token should be cleared and was if the handshake completed.
Mechanism of action

GFEs receive the IETF QUIC token with the new field, dereferenced a nullptr, and crashed.

Handshake doesn’t complete, so due to a bug, the client keeps using the "poison" token.

“Uh. Hello? Are you there?”
**Mechanism of action**

The client continues trying QUIC with the token, crashing every GFE they speak to.
Mechanism of action

When Chromium clients see a handshake failure, they mark QUIC is "broken", and go into exponential backoff. 5 minutes... 10 minutes...

Everyone I try to speak QUIC to never replies. This must be a broken QUIC server; I'm going to wait an increasing amount of time before trying QUIC again.
From canary to resolution

00:27 PST
4 Canary GFEs receive updated flags
GFEs in Europe begin crashing

00:31 PST
Probers fail and SREs alerted by pages
Canary judge automatically rolls back flags after 4 minutes

00:42 PST
European GFEs continue crashing
London SREs realize all monitoring tools, including crashlogs, are inaccessible

01:38 PST
London SREs learn it’s not a global outage
India, NZ SREs reroute all European UberProxy traffic

01:51 PST
SREs disable QUIC
Page me at 6am to figure out what happened
Challenges of 0-RTT

0-RTT is hard, much harder in IETF QUIC than gQUIC

- IETF QUIC can perform better than gQUIC... after fixing many bugs
- Packet Number Spaces add complexity, particularly in combination with PTO
- Key management is less synchronous than TLS over TCP

Facebook Networking@Scale talk: [Recording](#), [Slides](#)
Thanks!