Moving fast at scale

Experience deploying IETF QUIC at Facebook

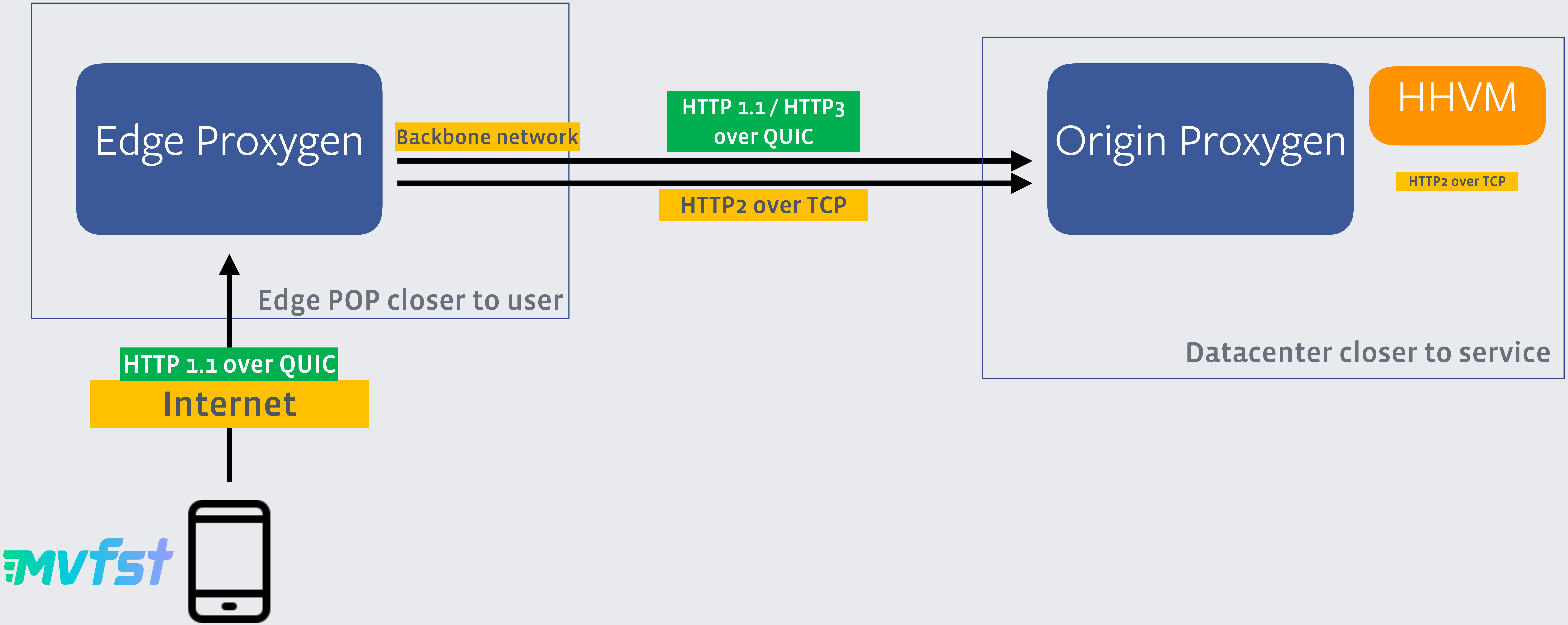
Subodh Iyengar

Luca Niccolini

Overview

- FB Infra and QUIC deployment
- Infrastructure parity between TCP and QUIC
- Results
- Future and current work

Anatomy of our load balancer infra



Infra parity between QUIC and TCP

- QUIC requires unique infrastructure changes
  - Zero downtime restarts
  - Packet routing
  - Connection Pooling
  - Instrumentation

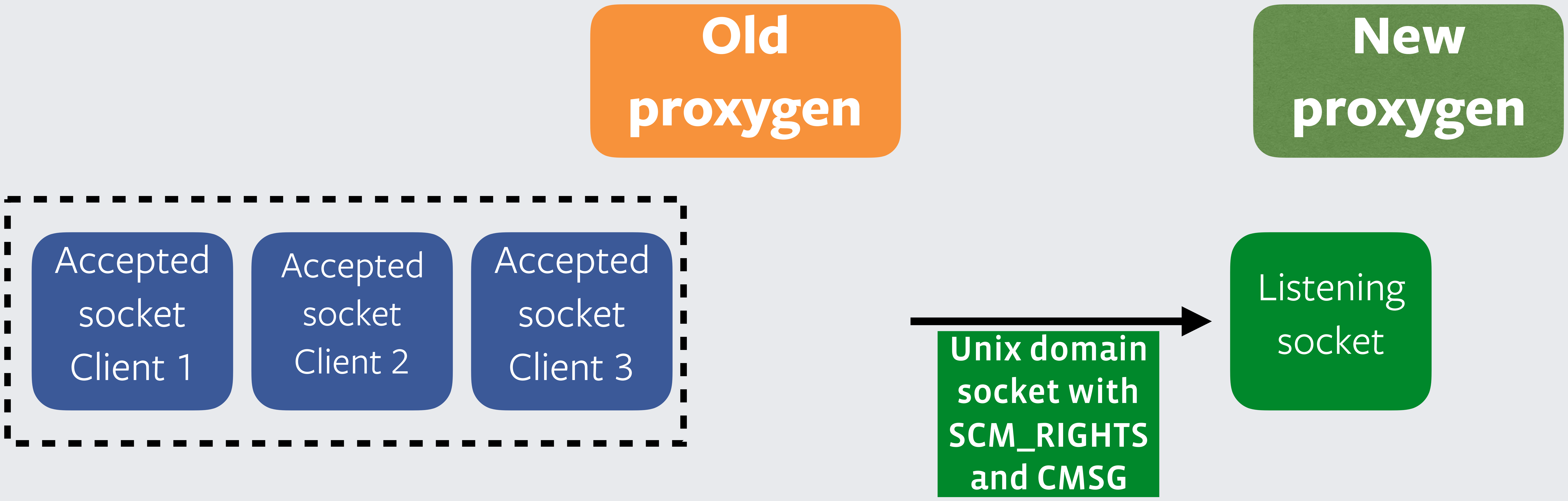
Zero downtime restarts

- We restart proxygen all the time
- Canaries, Binary updates
- Cannot shutdown all requests during restart
- Solution: Keep both old and new versions around for some time

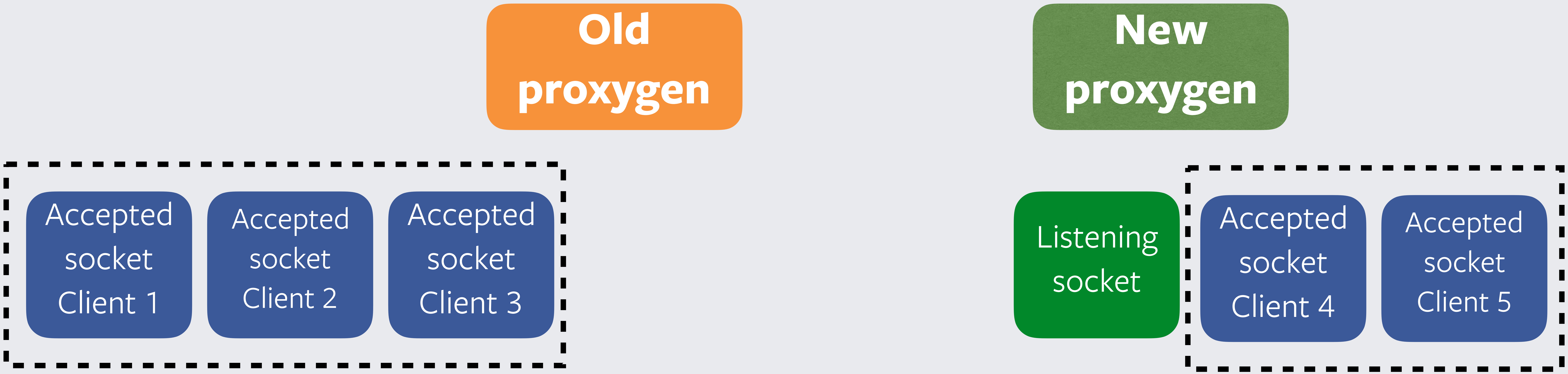
https://www.flickr.com/photos/ell-r-brown/26112857255
https://creativecommons.org/licenses/by-sa/2.0/

Zero downtime restarts in TCP

Zero downtime restarts in TCP



Zero downtime restarts in TCP



Zero downtime restarts in QUIC

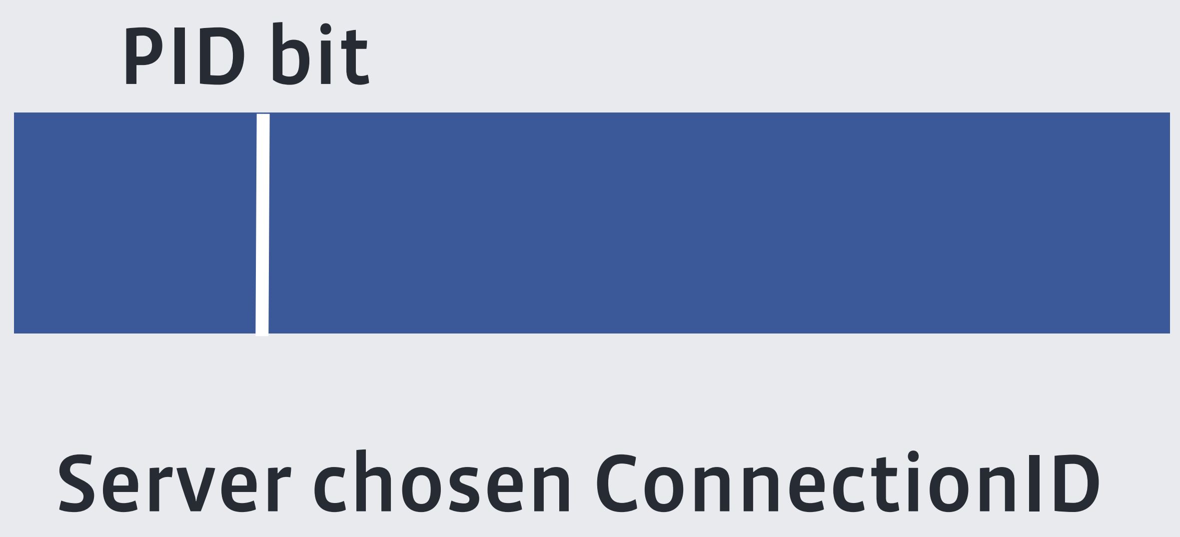
Problems

- No listening sockets in UDP
- Why not SO_REUSEPORT
    - SO_REUSEPORT and REUSEPORT_EBPF does not work on its own

Zero downtime restarts in QUIC

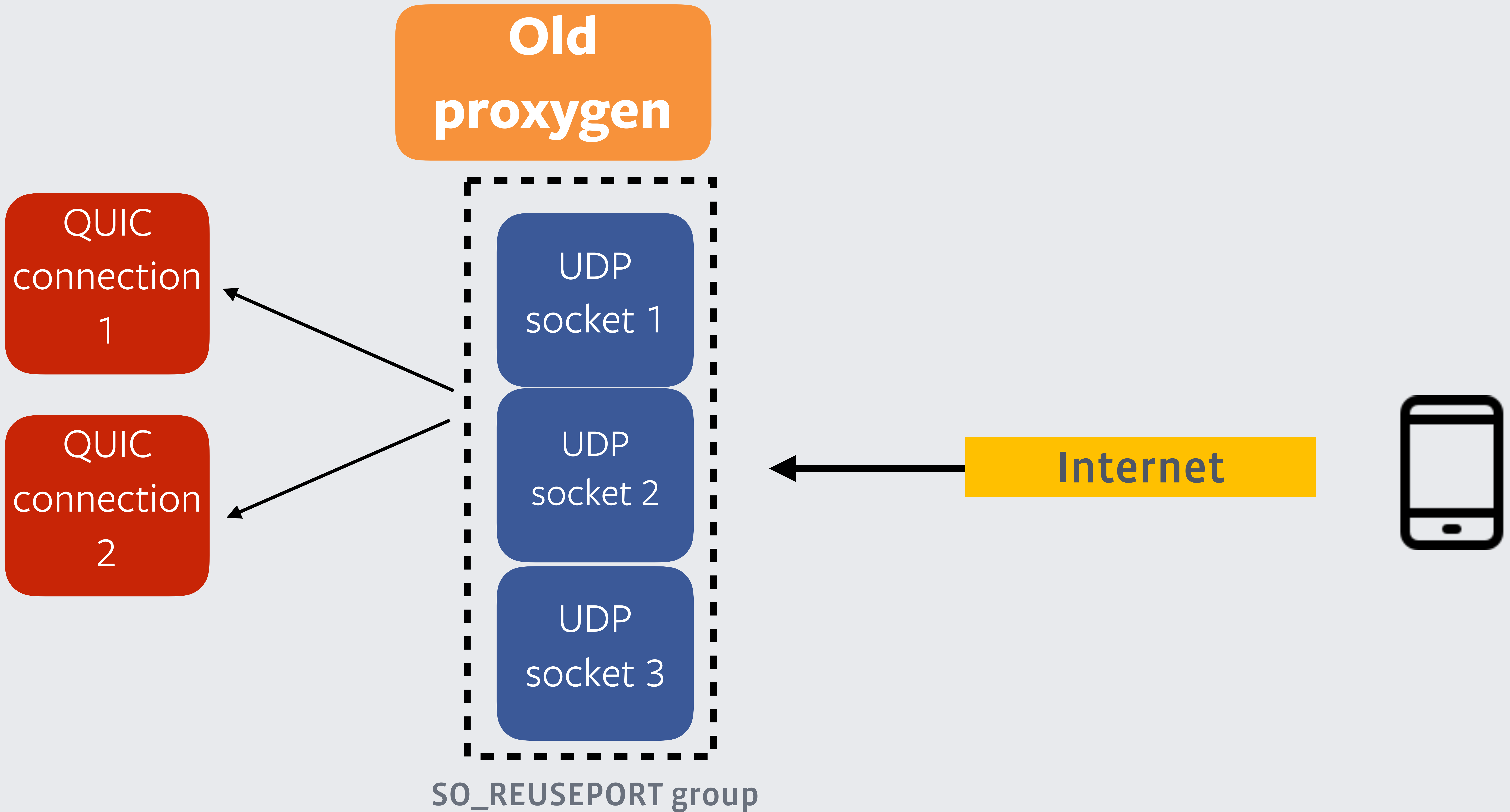
Solution

- Forward packets from new server to old server based on a "ProcessID"
- Each process gets its own ID: 0 or 1
- New connections encode ProcessID in server chosen ConnectionID
- Packets DSR to client



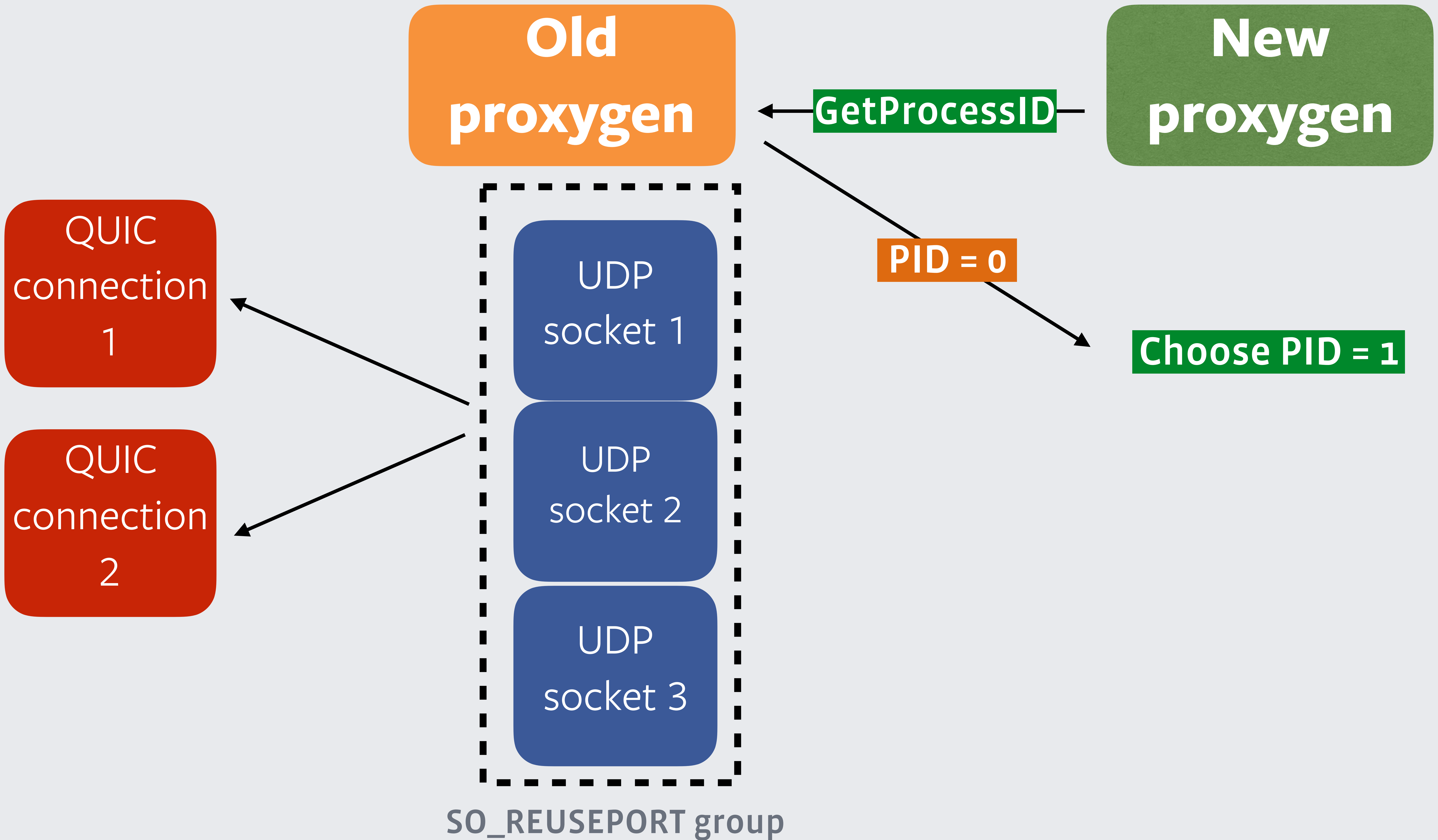
Zero downtime restarts in QUIC

Solution



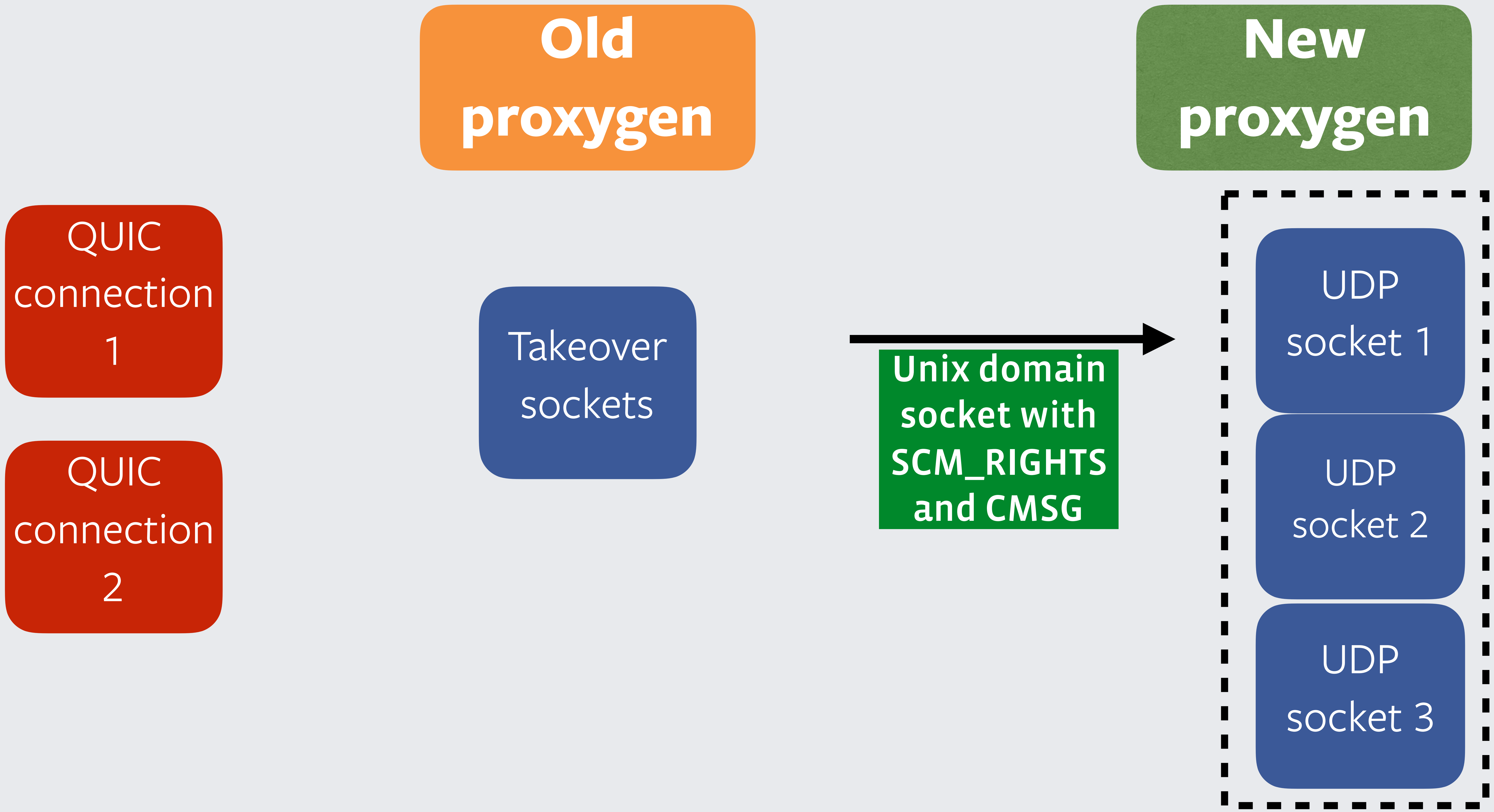
Zero downtime restarts in QUIC

Solution



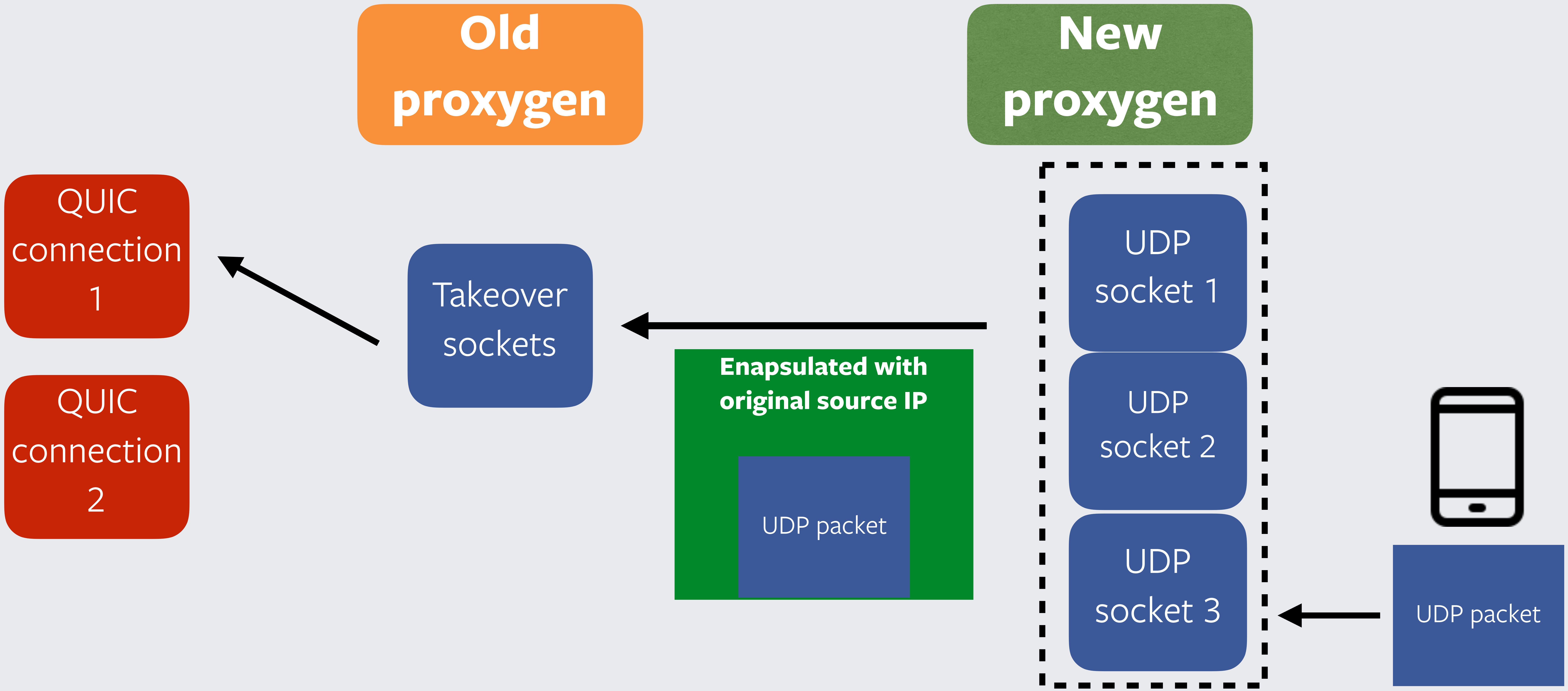
Zero downtime restarts in QUIC

Solution



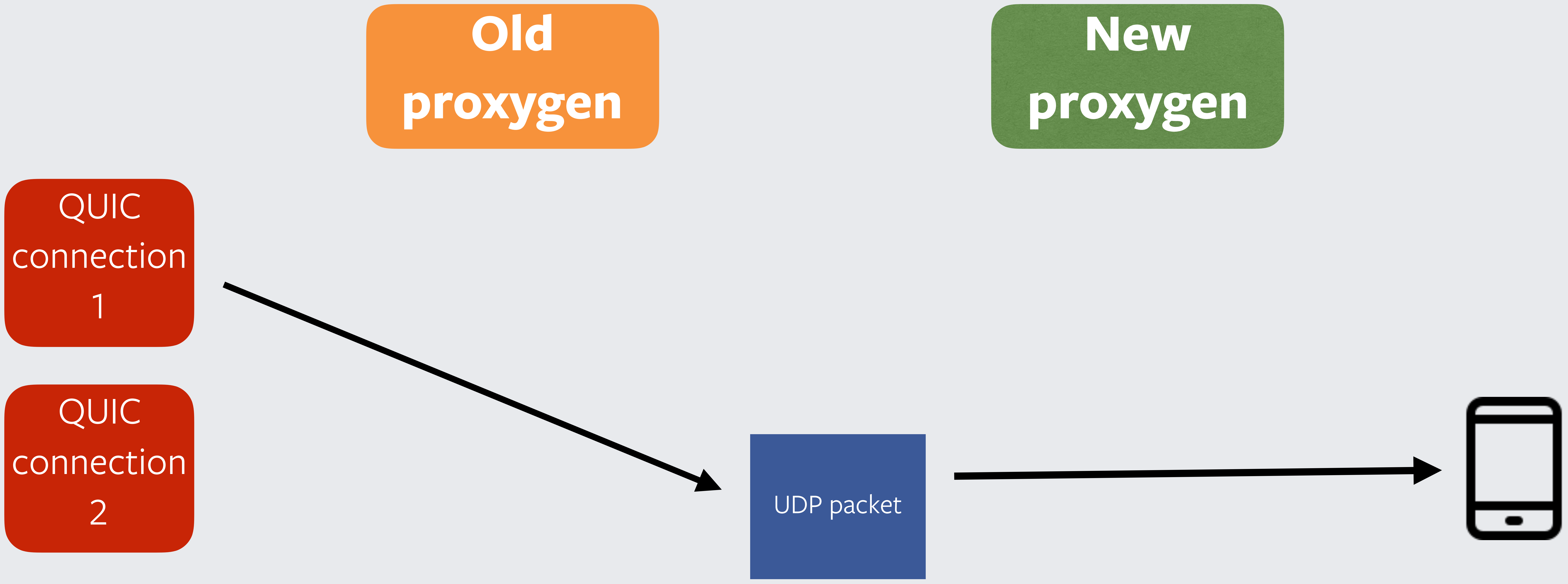
Zero downtime restarts in QUIC

Solution

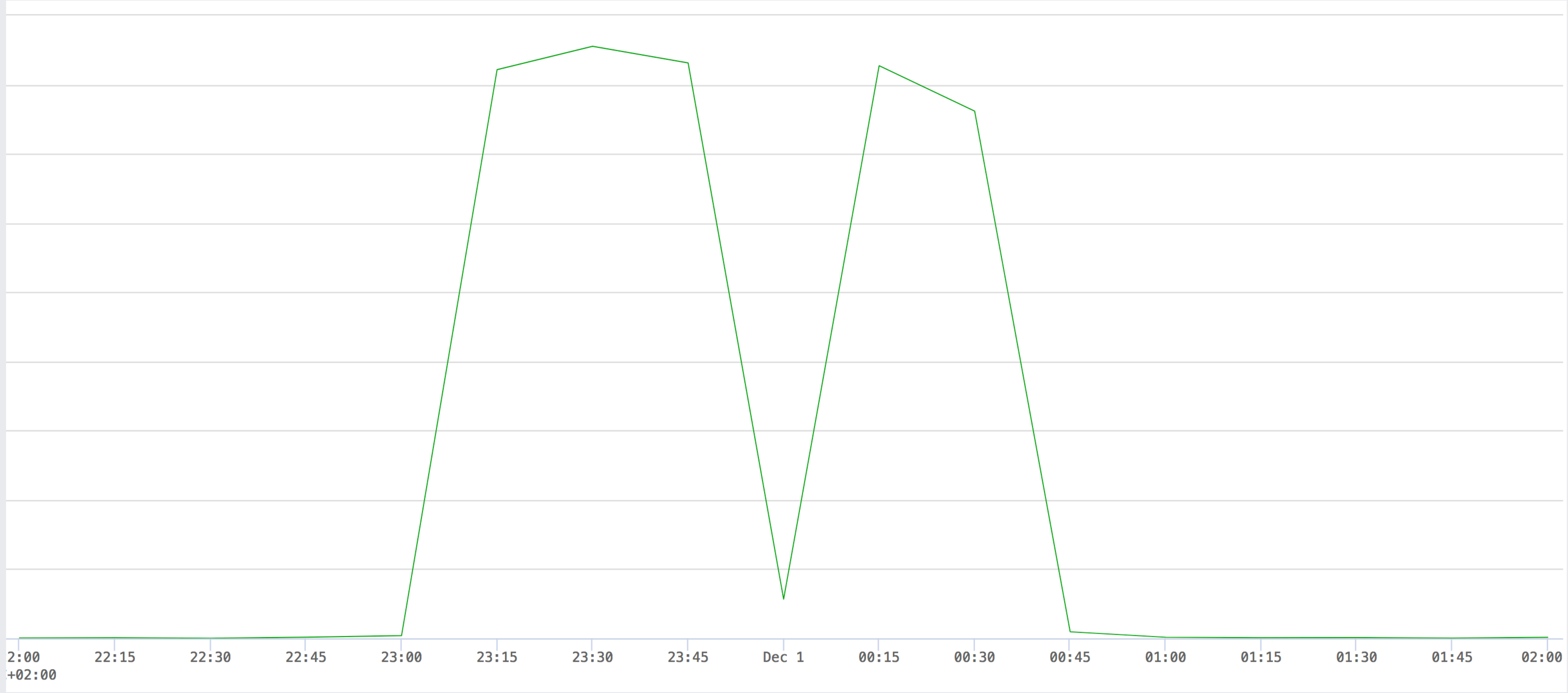


Zero downtime restarts in QUIC

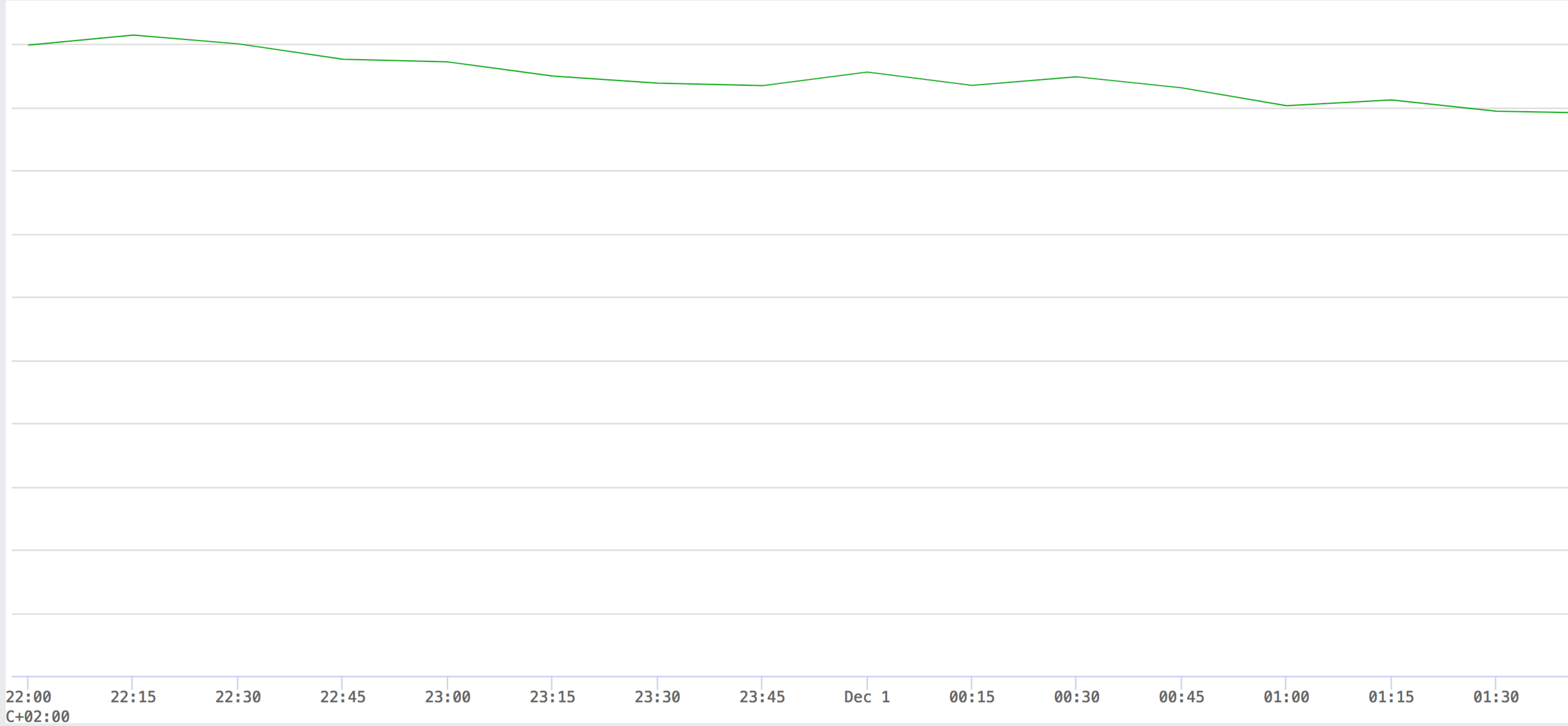
Solution



Results



packets forwarded during restart



packets dropped during restart

The Future

Coming to a 4.19 kernel near you

Introduce BPF_MAP_TYPE_REUSEPORT_SOCKARRAY and BPF_PROG_TYPE_SK_REUSEPORT

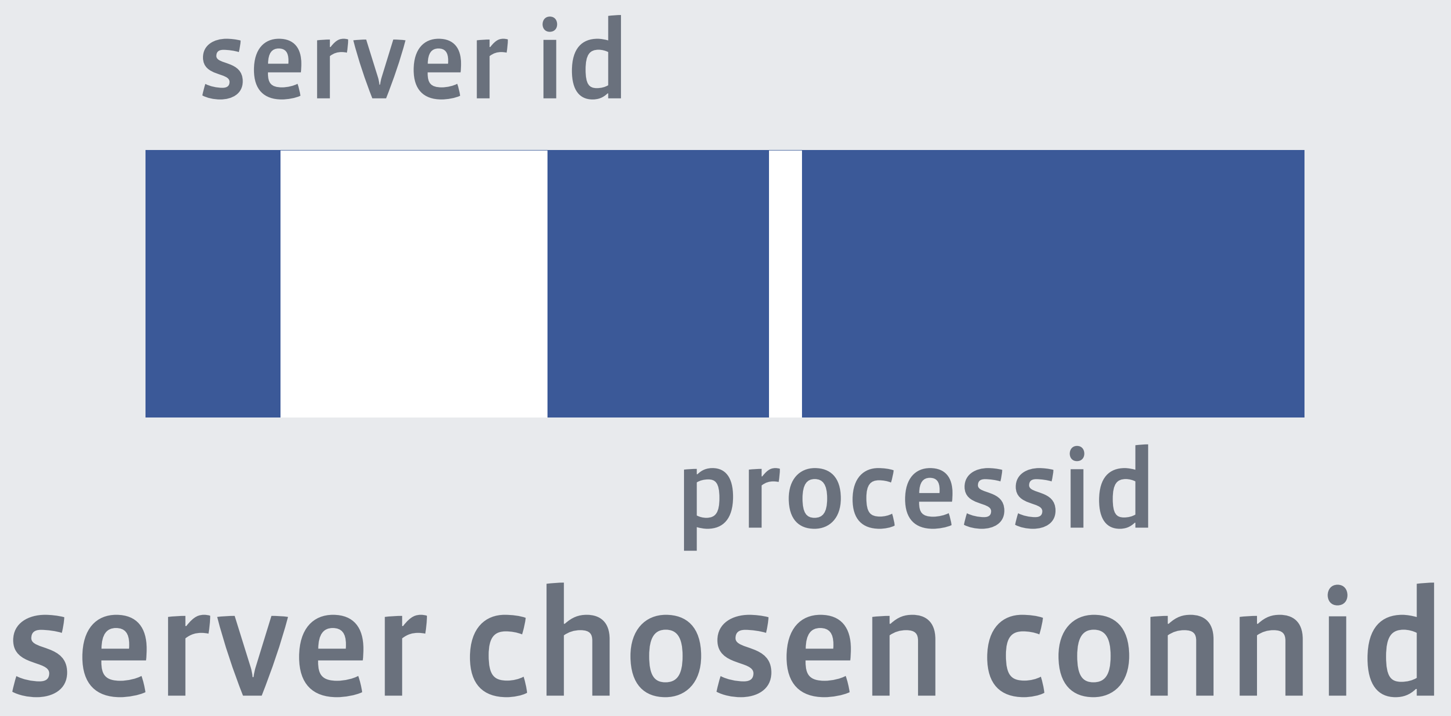
From:	Martin KaFai Lau <kafai-AT-fb.com>
To:	<netdev-AT-vger.kernel.org>
Subject:	[PATCH bpf-next 0/9] Introduce BPF_MAP_TYPE_REUSEPORT_SOCKARRAY and BPF_PROG_TYPE_SK_REUSEPORT
Date:	Wed, 8 Aug 2018 00:59:17 -0700
Message-ID:	<20180808075917.3009181-1-kafai@fb.com>
Cc:	Alexei Starovoitov <ast-AT-fb.com>, Daniel Borkmann <daniel-AT-iogearbox.net>, <kernel-team-AT-fb.com>
Archive-link:	Article

https://lwn.net/Articles/762101/

Stable routing

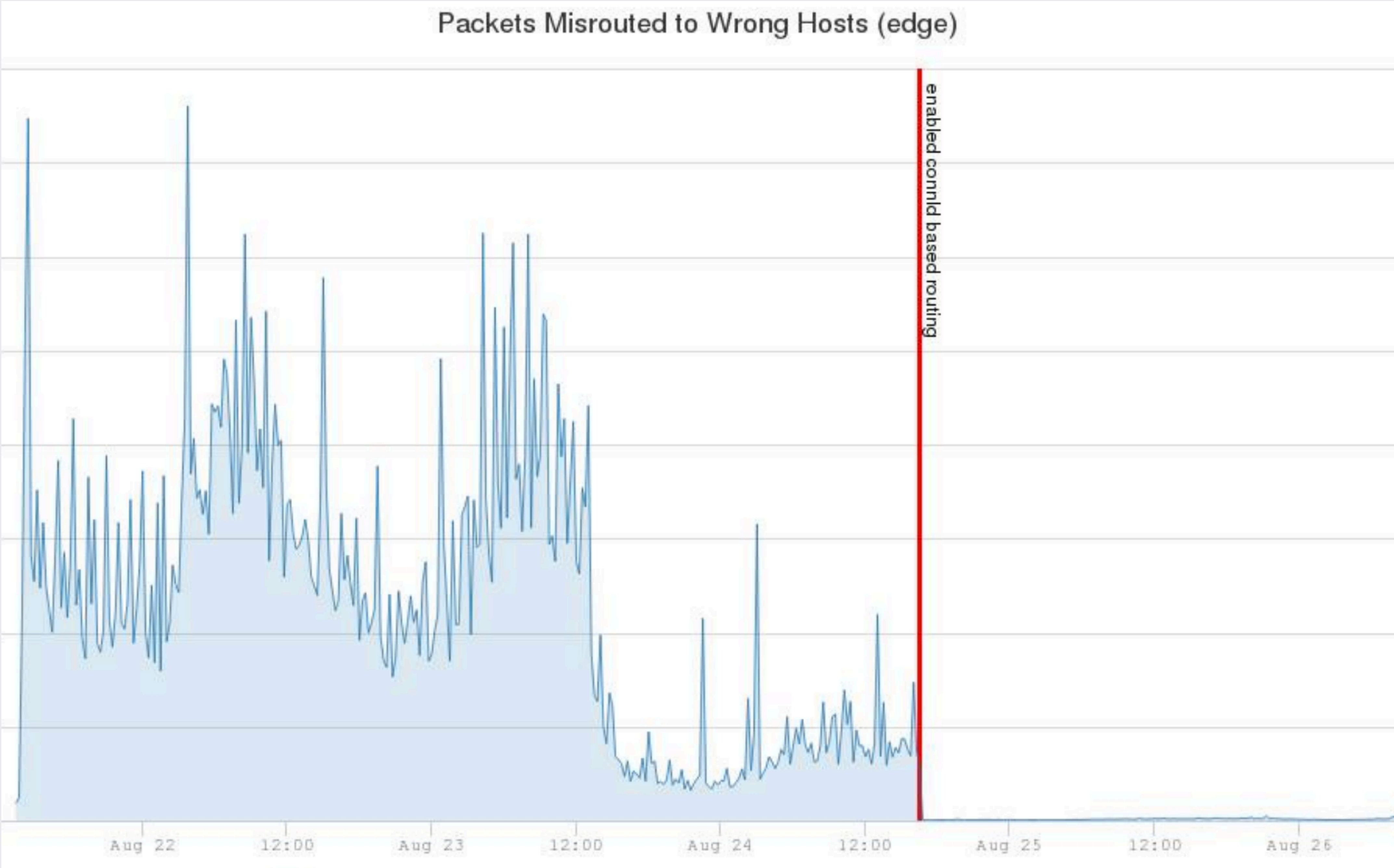
Stable routing of QUIC packets

- We were seeing a large % of timeouts
- We first suspected dead connections
- Implemented resets, even more reset errors
- Could not ship resets
- We suspected misrouting, hard to prove
- Gave every host its unique id
- Packet lands on wrong server, log server id
- Isolate it to cluster level. Cause was misconfigured timeout in L3



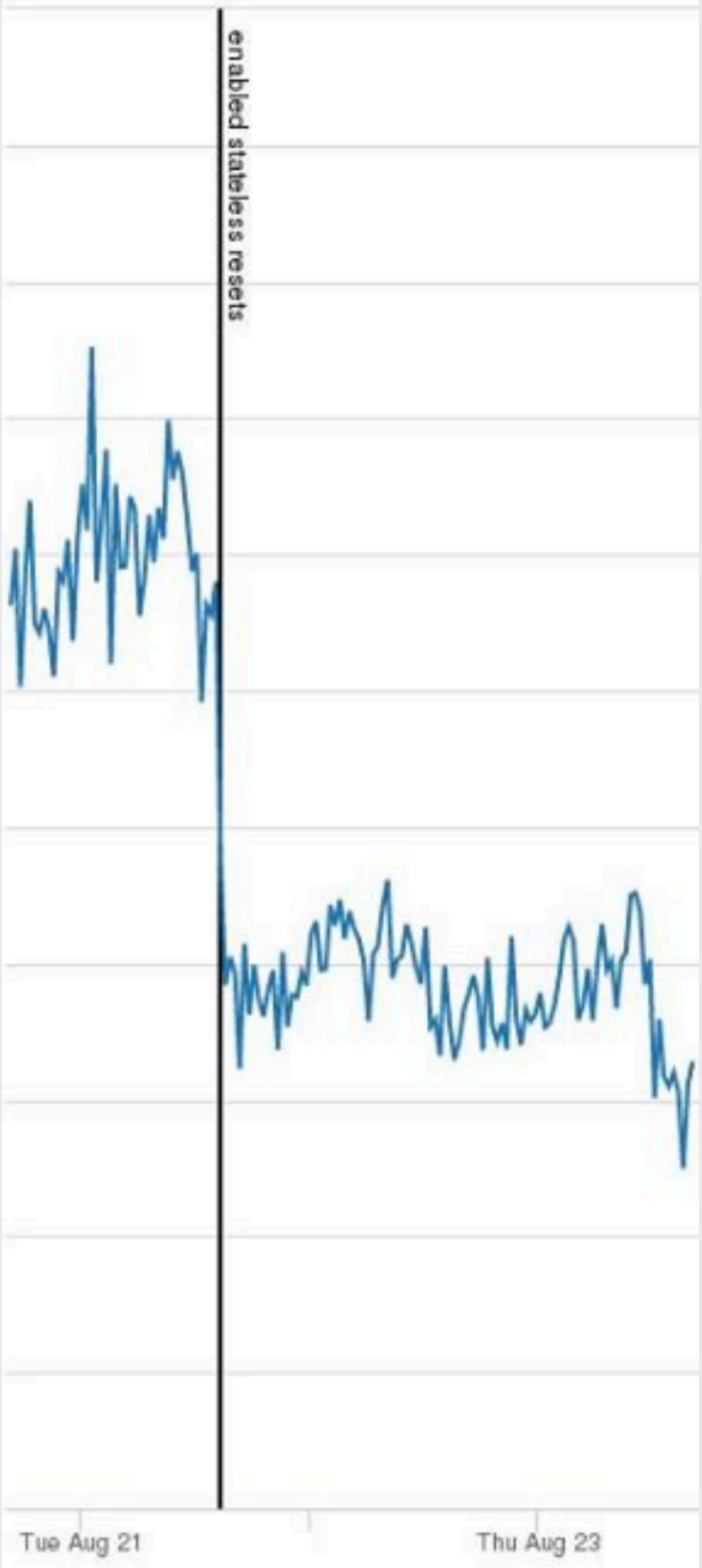
Stable routing of QUIC packets

- We have our own L3 load balancer, katran. Open source
- Implemented support for looking at serverid
- Stateless routing
- Misrouting went down to 0
- We're planning to use this for future features like multi-path and anycast QUIC



Stable routing of QUIC packets

- Now we could implement resets
- -15% drop in request latency without any change in errors



Connection pooling

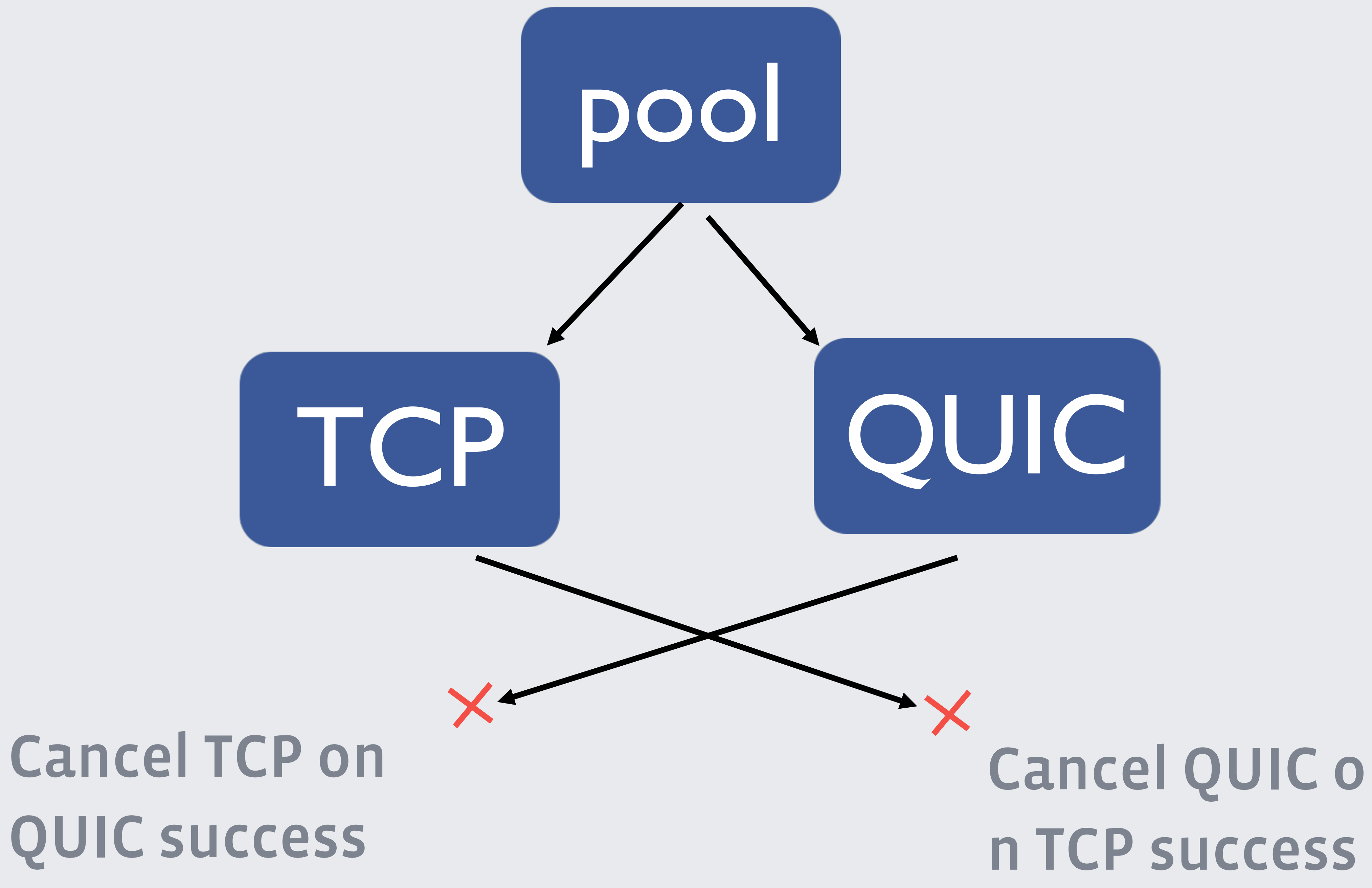
https://pixabay.com/en/swimming-puppy-summer-dog-funny-1502563/

Pooling connections

- Not all networks allow UDP
- Out of a sample size of 25k carriers about 4k had no QUIC usage
- Need to race QUIC vs TCP
- We evolved our racing algorithm
- Racing is non-trivial

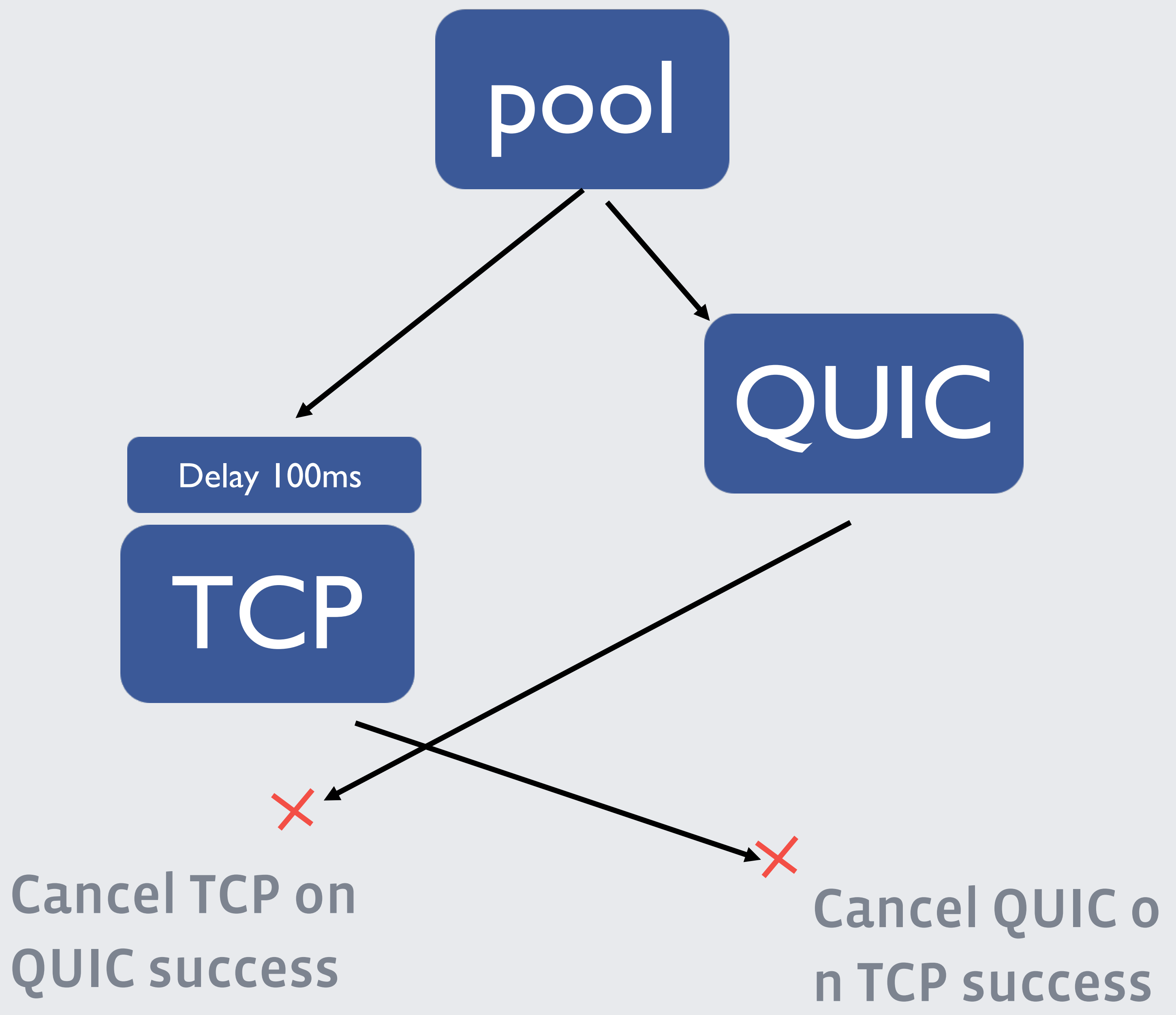
Naive algorithm

- Start TCP / TLS 1.3 0-RTT and QUIC at same time
- TCP success, cancel QUIC
- QUIC success, cancel TCP
- Both error, connection error
- Only 70% usage rate
- Probabilistic loss, TCP middleboxes, also errors: ENETUNREACH



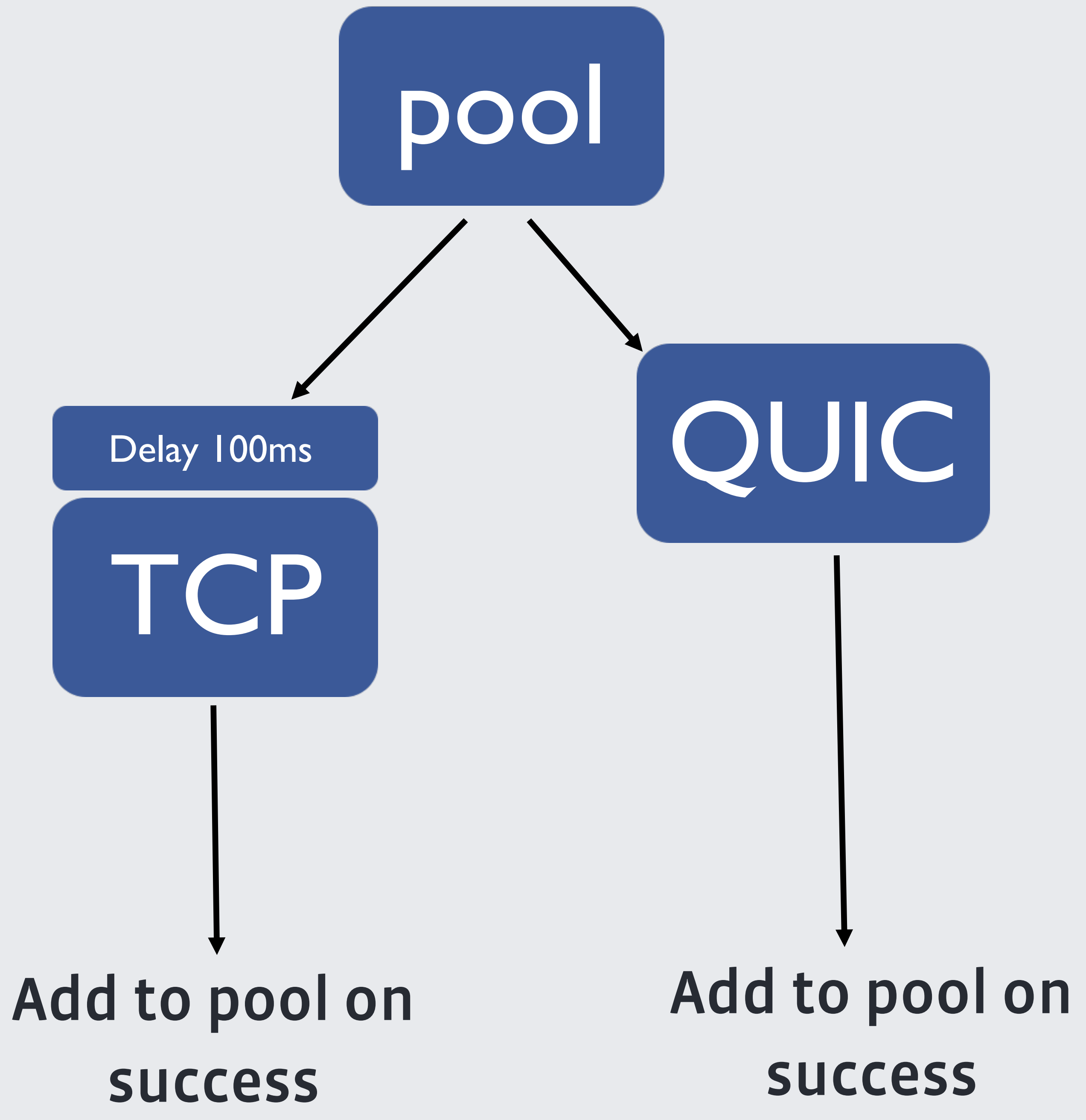
Let's give QUIC a head start

- Let's add a delay to starting TCP
- Didn't improve QUIC use rate
- Suspect radio wakeup delay and middleboxes
- Still seeing random losses even in working UDP networks



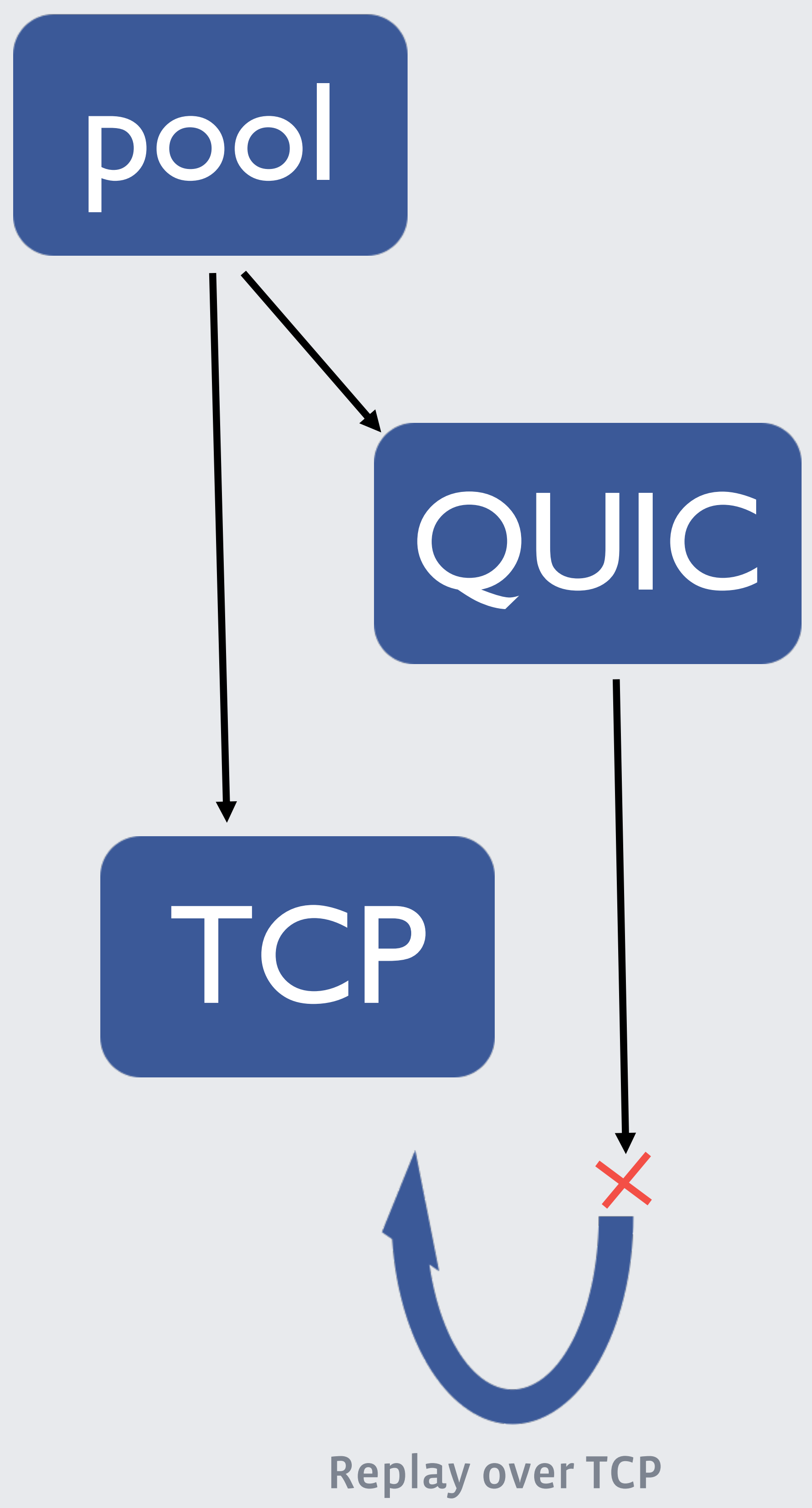
What if we don't cancel?

- Don't cancel QUIC when TCP success
- Remove delay on QUIC error and add delay back on success
- Pool both connections, new requests go over QUIC
- Complicated, needed major changes to pool
- Use rate improved to 93%
- Losses still random, but now can use QUIC even if it loses



What about zero rtt?

- No chance to test the network before sending 0-RTT data
- Conservative: If TCP + TLS 1.3 0-RTT succeeds, cancel requests over QUIC
- Replay requests over TCP



What about happy eyeballs?

- Need to race TCPv6, TCPv4, QUICv6 and QUICv4
- Built native support for Happy eyeballs in mvfst
- Treat Happy eyeballs as a loss recovery timer
- If 150ms fires, re-transmit CHLO on both v6 and v4.
- v6 use rate same between TCP and QUIC

Debugging QUIC in production

- We have good tools for TCP
- Where are the tools for QUIC?
- Solution: We built QUIC trace
- Schema-less logging: very easy to add new logs
- Data from both HTTP as well as QUIC
- All data is stored in scuba

Conn Rel Time	Event Name	Value
117959437822	packet_recvd	1, 1232
117959438154	packet_sent	1, 295, 1, 0
117959438163	cubic_sent	Hystart, 12320, 295, 0
117959515866	packet_recvd	2, 96
117959515972	fst_trace	derived 1-rtt write cipher
117959515987	fst_trace	derived 1-rtt read cipher
117959515995	fst_trace	write nst
117959516033	fst_trace	transport ready
117959516063	packet_sent	2, 34, 0, 1
117959516109	packet_sent	3, 245, 1, 0
117959516119	cubic_sent	Hystart, 12320, 540, 0
117959527316	packet_recvd	3, 1232
117959527331	update_rtt	89143, 4488, 89143, 89143
117959527336	packet_acked	1
117959527347	cubic_ack	Hystart, 12615, 245, 0
117959527392	stream_event	on_eom, 4, 0
117959527719	stream_event	on_headers, 4, 0
117959539589	packet_recvd	4, 1232
117959539994	stream_event	on_headers, 8, 12
117959549363	packet_recvd	5, 965
117959549382	stream_event	on_eom, 8, 22
117959558151	packet_sent	4, 35, 0, 1
117959576756	stream_event	headers, 4, 49

Debugging QUIC in production

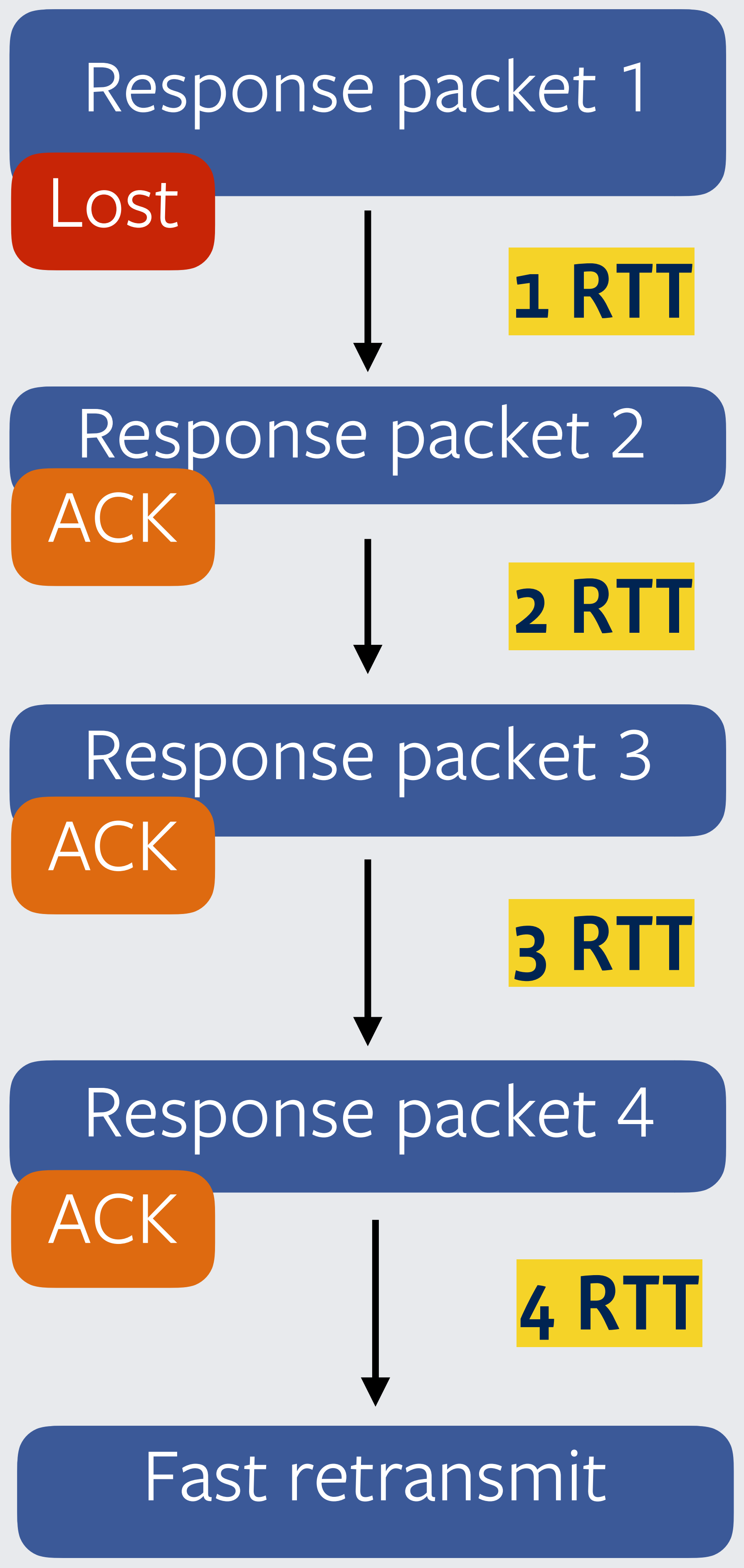
- Find bad requests in the requests table from proxygen
- Join it with the QUIC_TRACE table
- Can answer interesting questions like
  - What transport events happened around the stream id
  - Were we cwnd blocked
  - How long did a loss recovery take

Conn Rel Time	Event Name	Value
117959437822	packet_recvd	1, 1232
117959438154	packet_sent	1, 295, 1, 0
117959438163	cubic_sent	Hystart, 12320, 295, 0
117959515866	packet_recvd	2, 96
117959515972	fst_trace	derived 1-rtt write cipher
117959515987	fst_trace	derived 1-rtt read cipher
117959515995	fst_trace	write nst
117959516033	fst_trace	transport ready
117959516063	packet_sent	2, 34, 0, 1
117959516109	packet_sent	3, 245, 1, 0
117959516119	cubic_sent	Hystart, 12320, 540, 0
117959527316	packet_recvd	3, 1232
117959527331	update_rtt	89143, 4488, 89143, 89143
117959527336	packet_acked	1
117959527347	cubic_ack	Hystart, 12615, 245, 0
117959527392	stream_event	on_eom, 4, 0
117959527719	stream_event	on_headers, 4, 0
117959539589	packet_recvd	4, 1232
117959539994	stream_event	on_headers, 8, 12
117959549363	packet_recvd	5, 965
117959549382	stream_event	on_eom, 8, 22
117959558151	packet_sent	4, 35, 0, 1
117959576756	stream_event	headers, 4, 49

Debugging QUIC in production

- ACK threshold recovery is not enough
- HTTP connections idle for most of time
- In a reverse proxy requests / responses staggered ~TLP timer
- To get enough packets to trigger Fast retransmit can take > 4 RTT

https://github.com/quicwg/base-drafts/pull/1974



Results deploying QUIC

- Integrated mvfst in mobile and proxygen
- HTTP1.1 over QUIC draft 9 with 1-RTT
- Cubic congestion controller
- API style requests and responses
  - Requests about 247 bytes -> 13 KB
  - Responses about 64 bytes -> 500 KB
  - A/B test against TLS 1.3 with 0-RTT
    - 99% 0-RTT attempted

Results deploying QUIC

Latency	p75	p90	p99
Overall latency	-6%	-10%	-23%
Overall latency for responses < 4k	-6%	-12%	-22%
Overall latency for reused conn	-3%	-8%	-21%

Latency reduction at different percentiles for successful requests


Bias

What about bias?

Latency	p75	p90	p99
Latency for later requests	-1%	-5%	-15%
Latency for rtt < 500ms	-1%	-5%	-15%

Latency reduction at different percentiles for successful requests

Takeaways

- Initial 1-RTT QUIC results are very encouraging
- Lots of future experimentation needed
- Some major changes in infrastructure required

Questions?