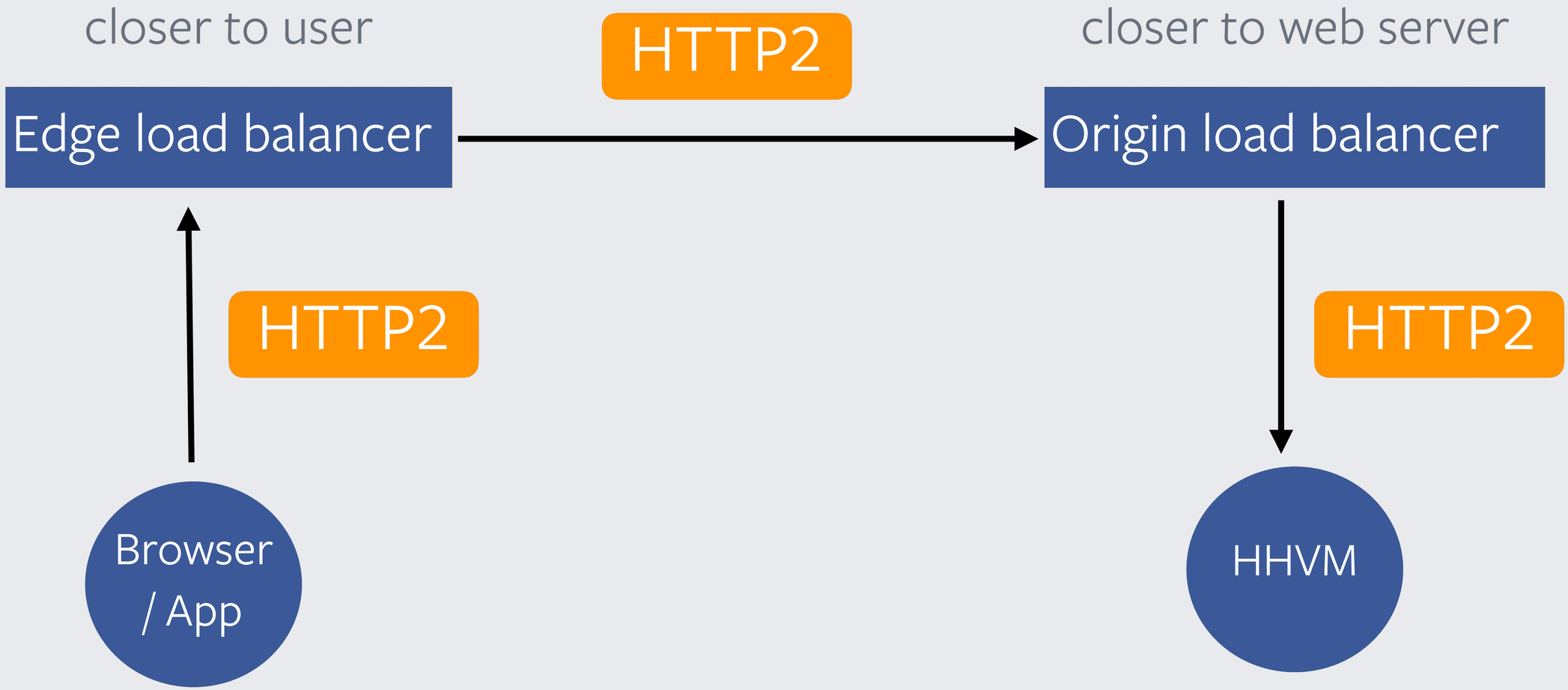
Deploying mvfst

Subodh Iyengar

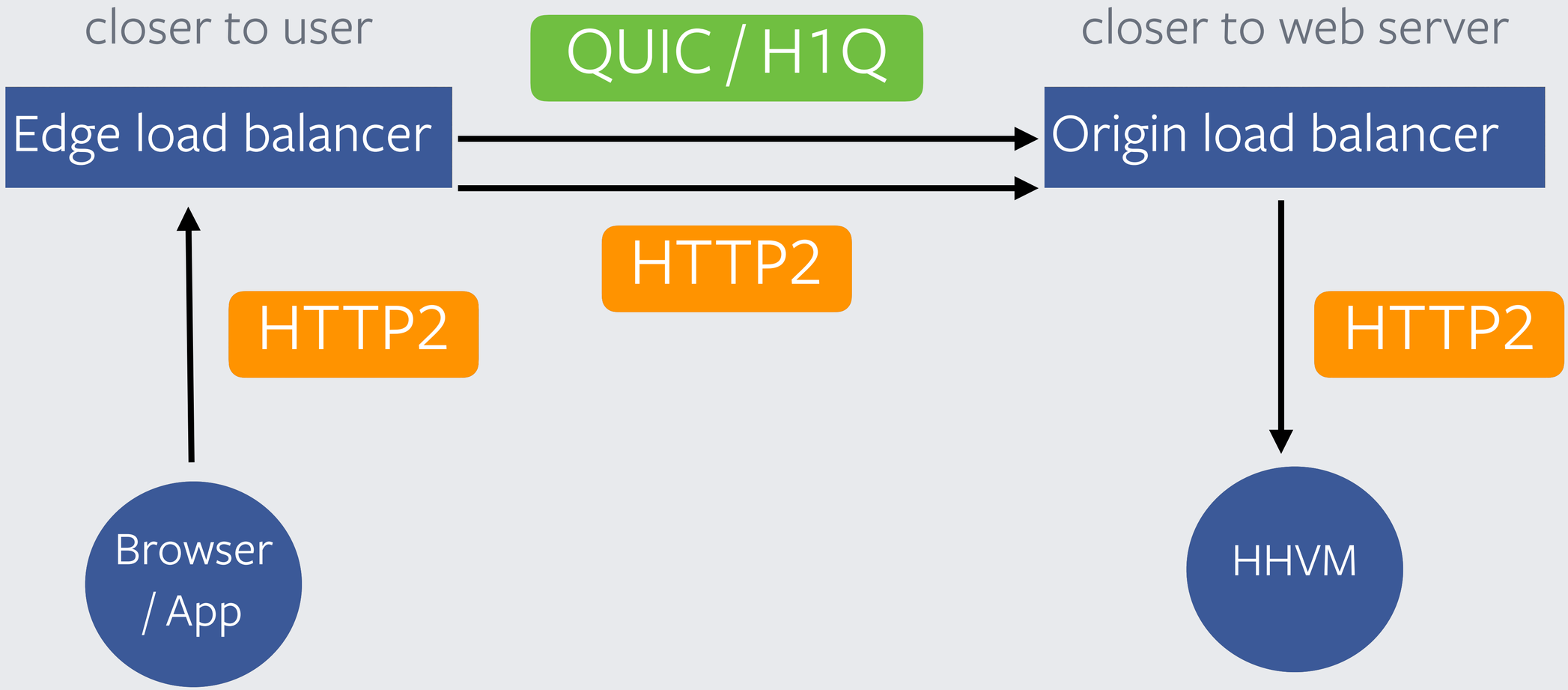
What is mvfst

- Facebook's iQUIC implementation
- For HTTP -> Proxygen
- For TLS 1.3 -> Fizz
- We control a lot of the stack
- Deploying internally to weed out bugs / performance issues

Architecture of our load balancer



Architecture of our load balancer



Test details

- **>> 100 billion** requests a day
- **4K** response size on average to about **70K**
- **~10ms -> ~200ms** RTT
- Almost **0** loss
- HTTP 1.1 over QUIC -> working on moving to HQ

Performance numbers

- Load testing framework
  - **70%** RPS of HTTP/2 over TLS 1.2 for various tests
  - In production, practically CPU idle **1-2%**, non-issue.
- Memory consumption stayed the same
- Latency is mostly on par with TCP
  - For some VIPs its **10%** better on average

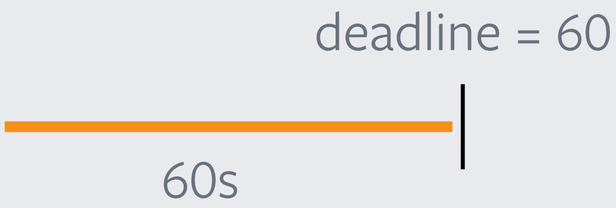
Fun issues: Idle timeout + RTO

- **Idle timeout + RTO**

Section 7.9
”A connection will time out if no packets are sent or received for a period longer than the time specified in the idle_timeout transport parameter”

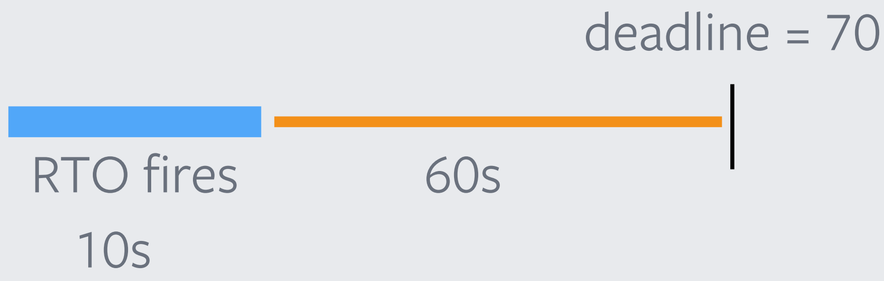
Fun issues: Idle timeout + RTO

- **Idle timeout + RTO**



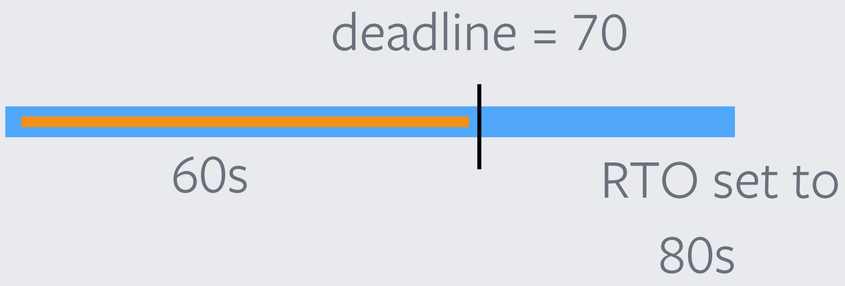
Fun issues: Idle timeout + RTO

- **Idle timeout + RTO**



Fun issues: Idle timeout + RTO

- **Idle timeout + RTO**



Fun issues: Idle timeout

- **Idle timeout**
- Mix of long lived (eg. long polling), and short lived requests
- Coming up with right idle timeout hard

Fun issues: Stateless resets

- Socket takeover normally does 0 downtime
- However in crashes no time to takeover
- We were seeing timeouts when the server crashed
- **Stateless reset helps**

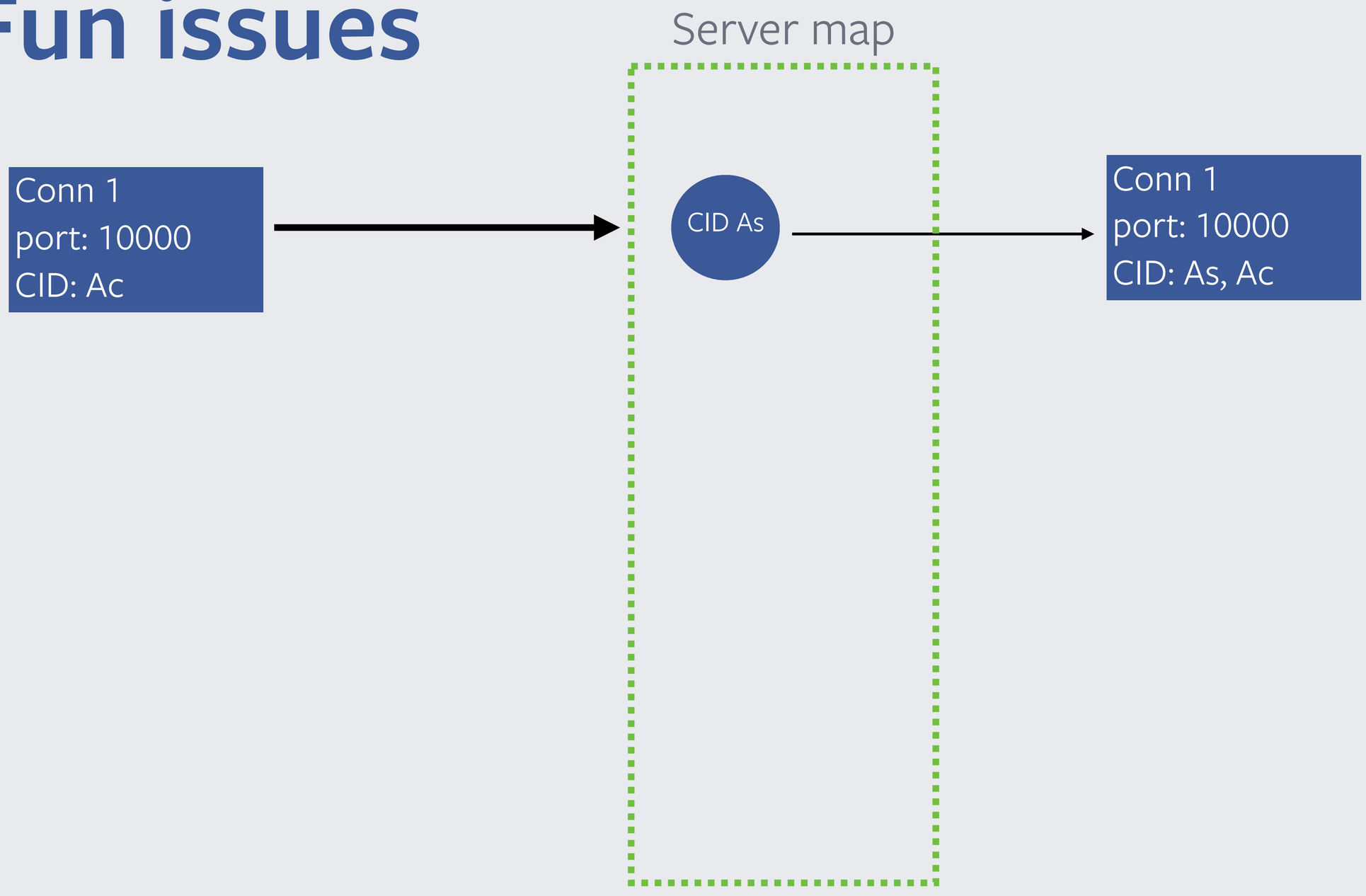
Fun issues: Draining

- TIME_WAIT absorbs packets from other flows if port is reused in TCP
- QUIC can inherently handle that, but also has draining
- On the client we closed without draining
- Server's last flight transmitted packets does not find bound port, ICMPv6 traffic exploded
- **Draining is important**

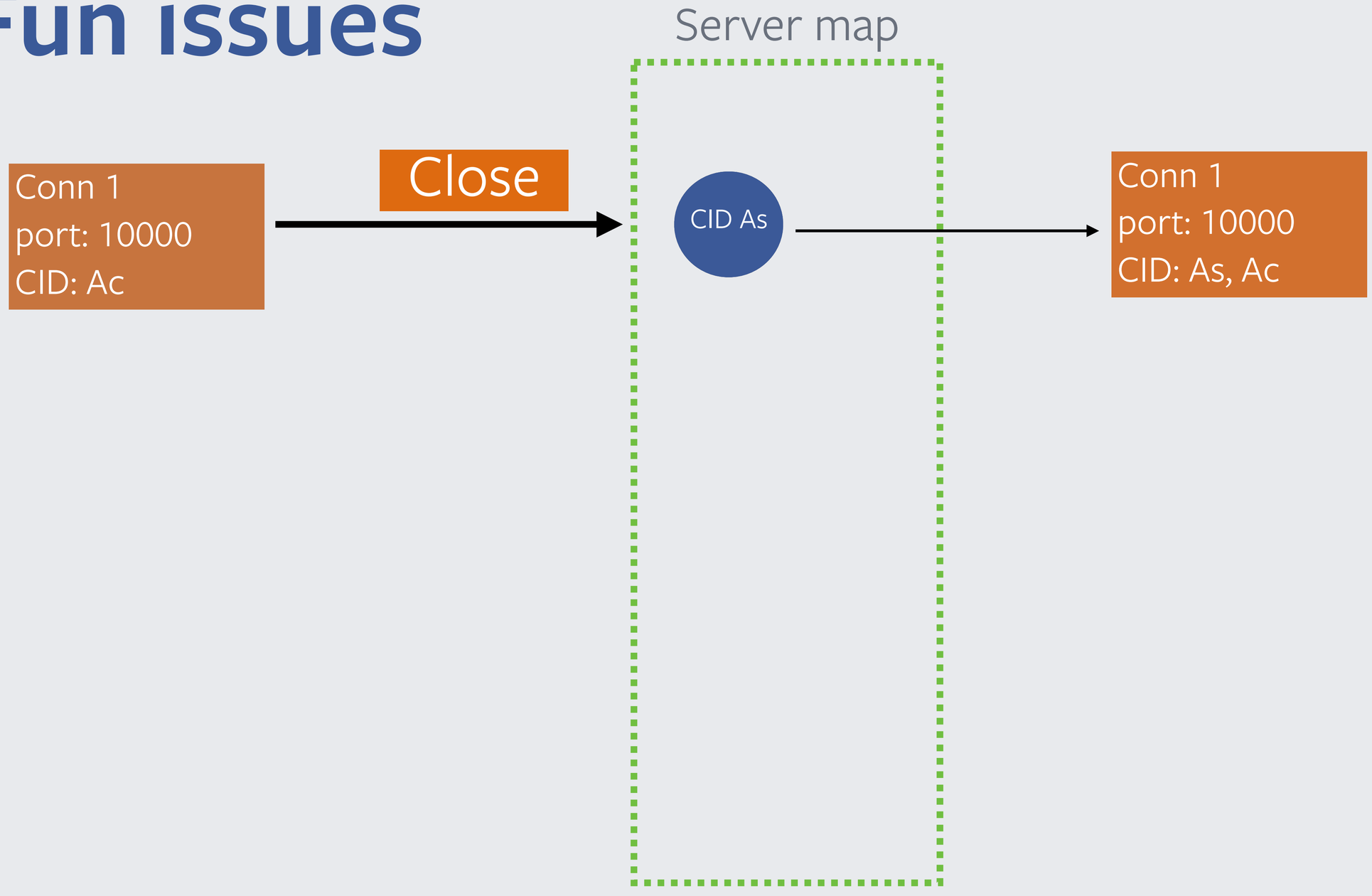
Fun issues: Packet number

- Last byte ACKed time spiked in very few cases
- We wanted to blame packet loss so bad
- TCPdumps showed multiple connections on a 5 tuple !!!!
- **Packet sequence number in TCPdump helped**
- Other UDP sockets swallowed packets due to SO_REUSEADDR

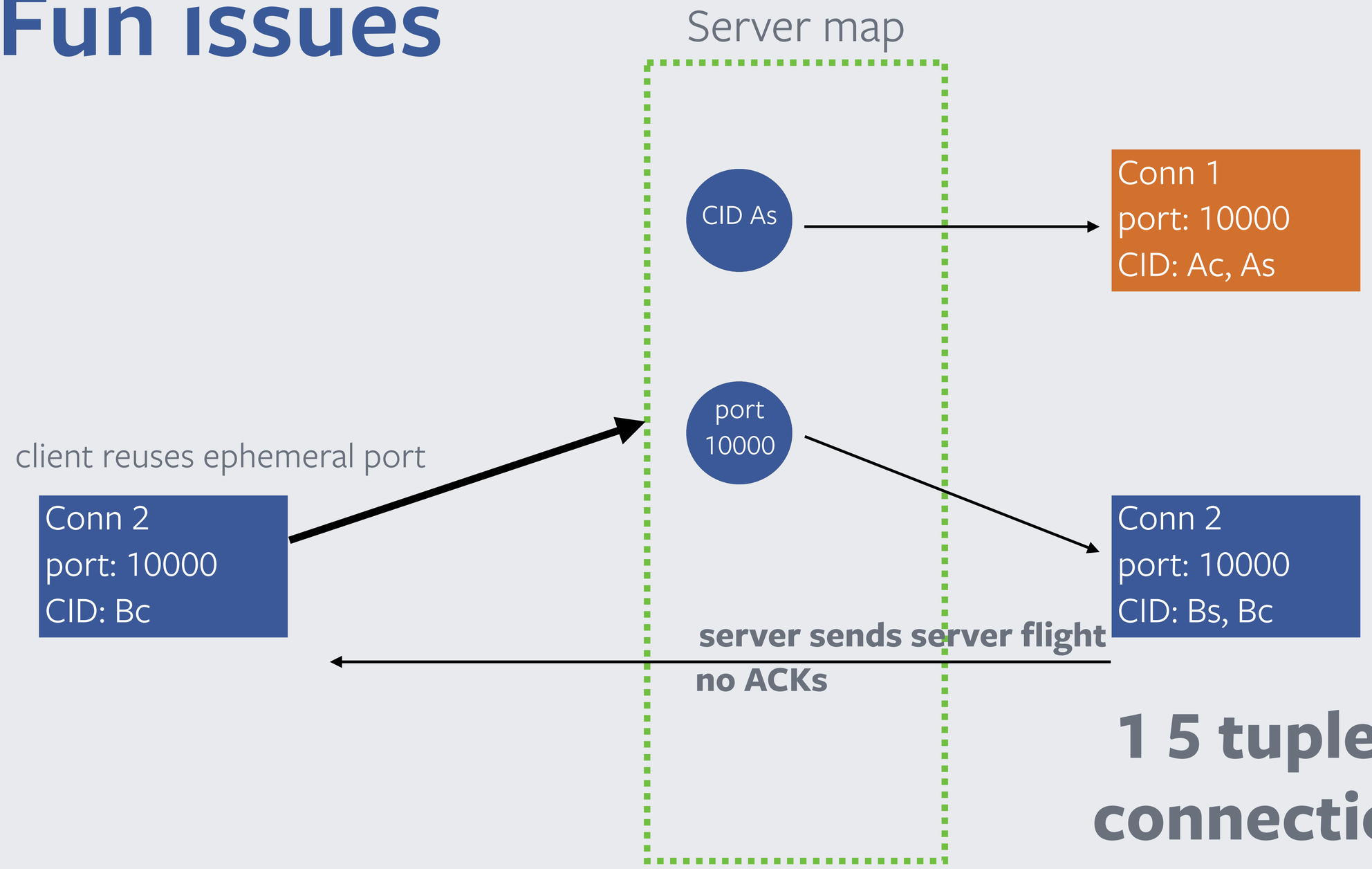
Fun issues


Server map
Conn 1
port: 10000
CID: Ac
CID As
Conn 1
port: 10000
CID: As, Ac

Fun issues

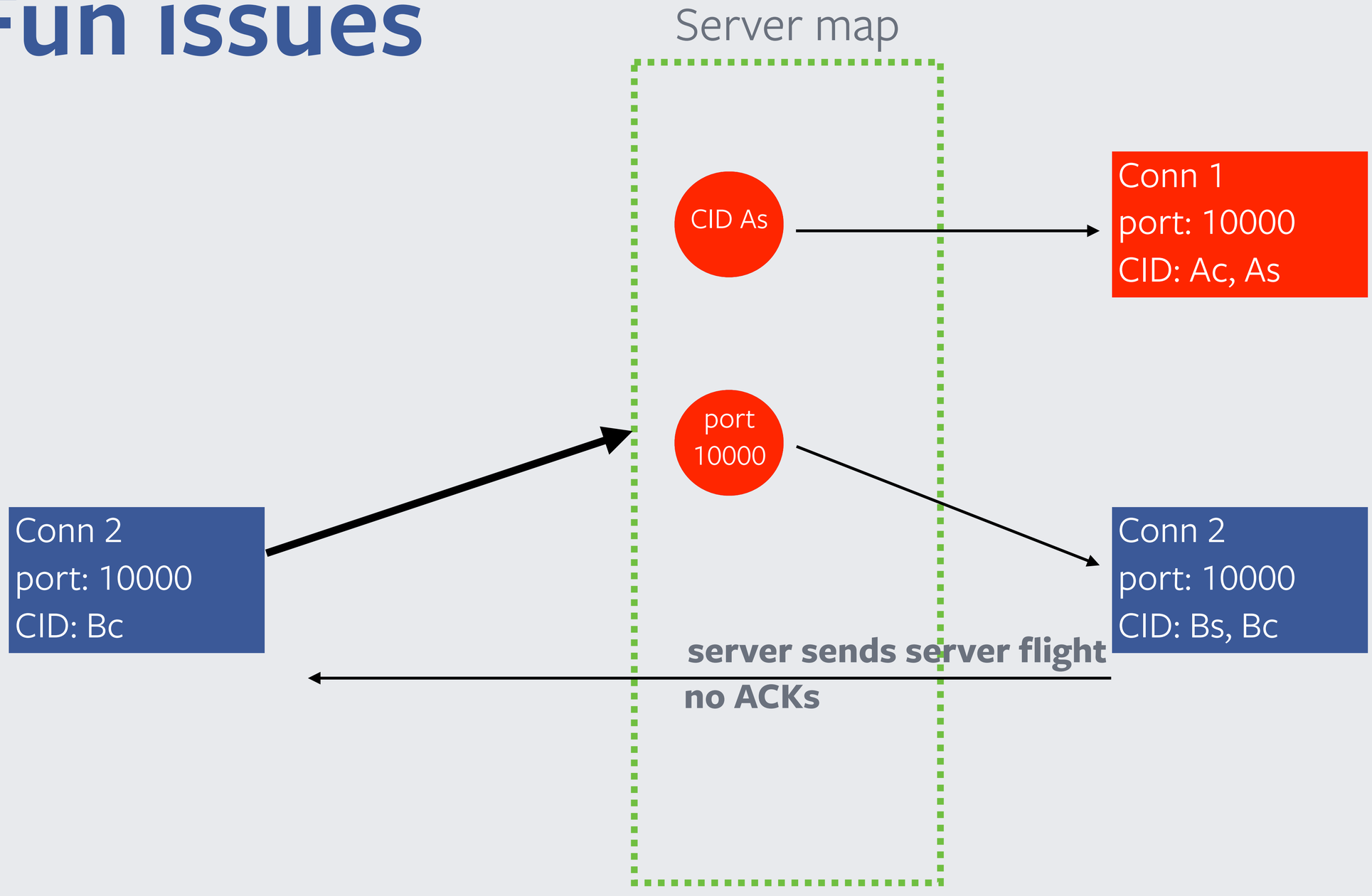

Server map
Conn 1
port: 10000
CID: Ac
Close
CID As
Conn 1
port: 10000
CID: As, Ac

Fun issues

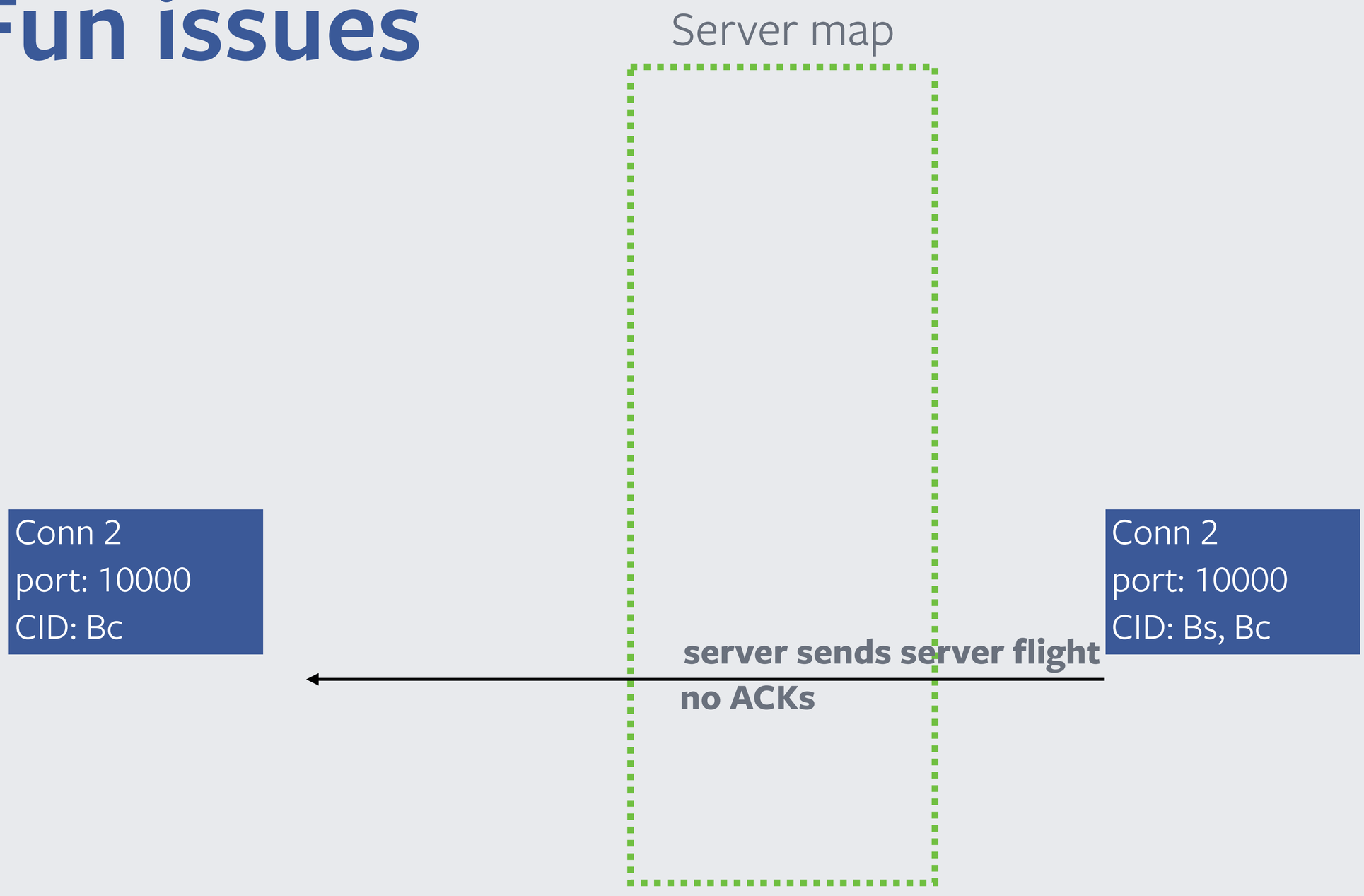


1 5 tuple, 2 connections

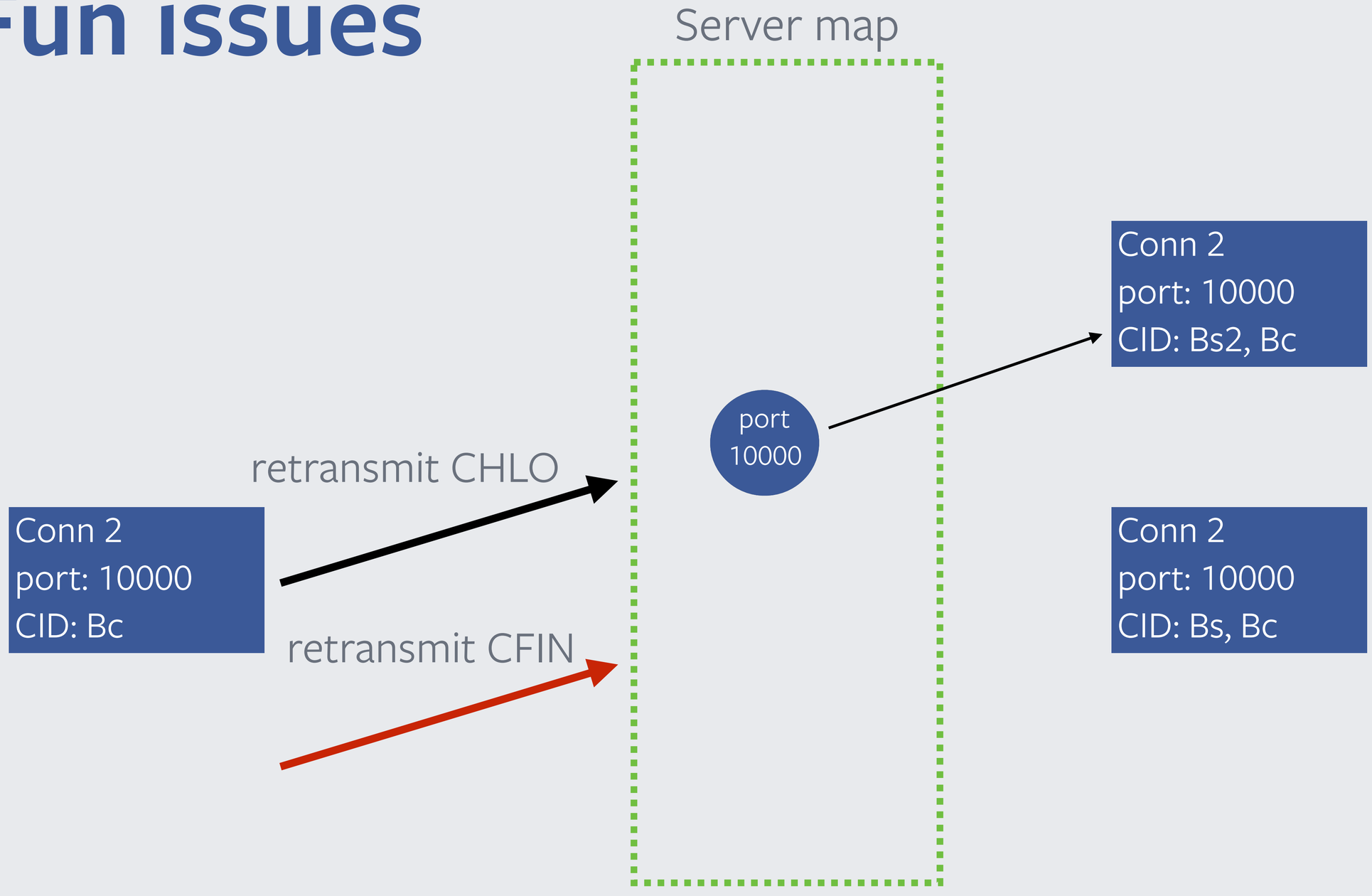
Fun issues


Server map
CID As
Conn 1
port: 10000
CID: Ac, As
port
10000
Conn 2
port: 10000
CID: Bc
Conn 2
port: 10000
CID: Bs, Bc
server sends server flight
no ACKs

Fun issues


Server map
Conn 2
port: 10000
CID: Bc
Conn 2
port: 10000
CID: Bs, Bc
server sends server flight
no ACKs

Fun issues


Server map
Conn 2
port: 10000
CID: Bc
retransmit CHLO
retransmit CFIN
port
10000
Conn 2
port: 10000
CID: Bs2, Bc
Conn 2
port: 10000
CID: Bs, Bc

Fun issues

- Keyed address map by client conn id, 2 tuple
- New optional conn id scheme doesn't allow this
- Need to refcount / use conn state to decide whether to evict from address map

What’s next?

- We’re starting a test from mobile apps to servers