Stateless Reset

QUIC Interim 2017-06, Paris

Manifest Confusion

What is the purpose of a Stateless Reset?

What signals do we want endpoints to generate?

...and who do we want to have consume those signals?

What is the role of a middlebox in QUIC?

Signals Taxonomy

A simplistic taxonomy divides things into sender/receiver

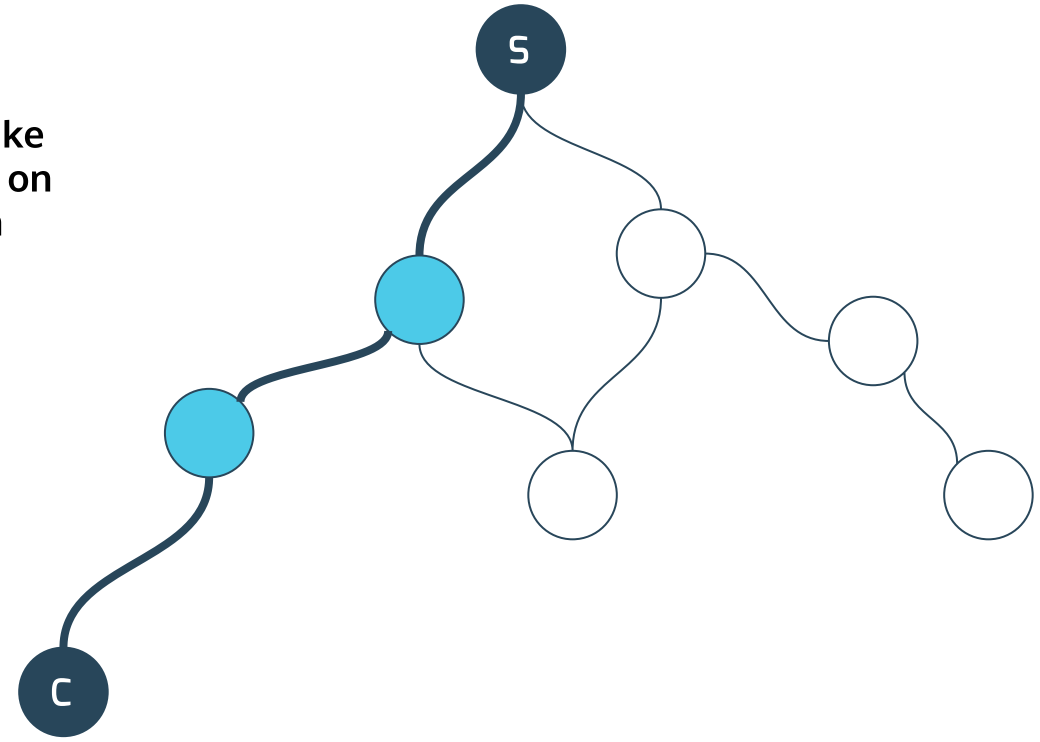
- end-to-end - most things
- end-to-path - a bunch of implicit signals only
- path-to-end - PMTU signals, ECN

One of these is not like the others:

there is only one connection, but multiple paths

Simple Migration

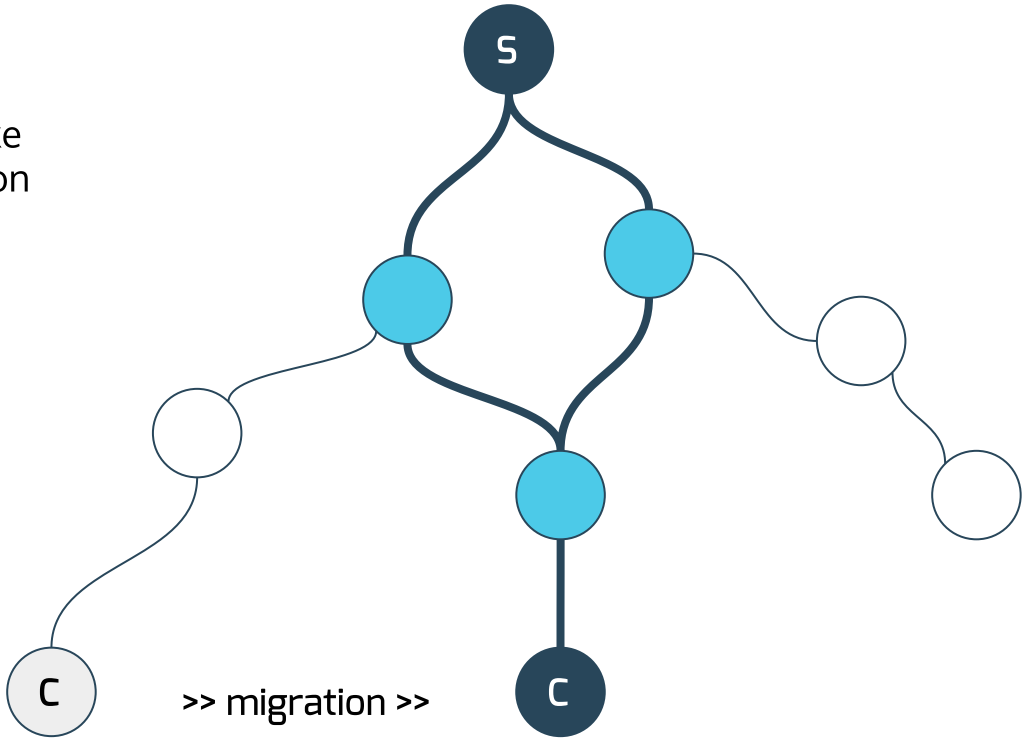
Handshake happens on one path



QUIC

Simple Migration

Handshake happens on one path

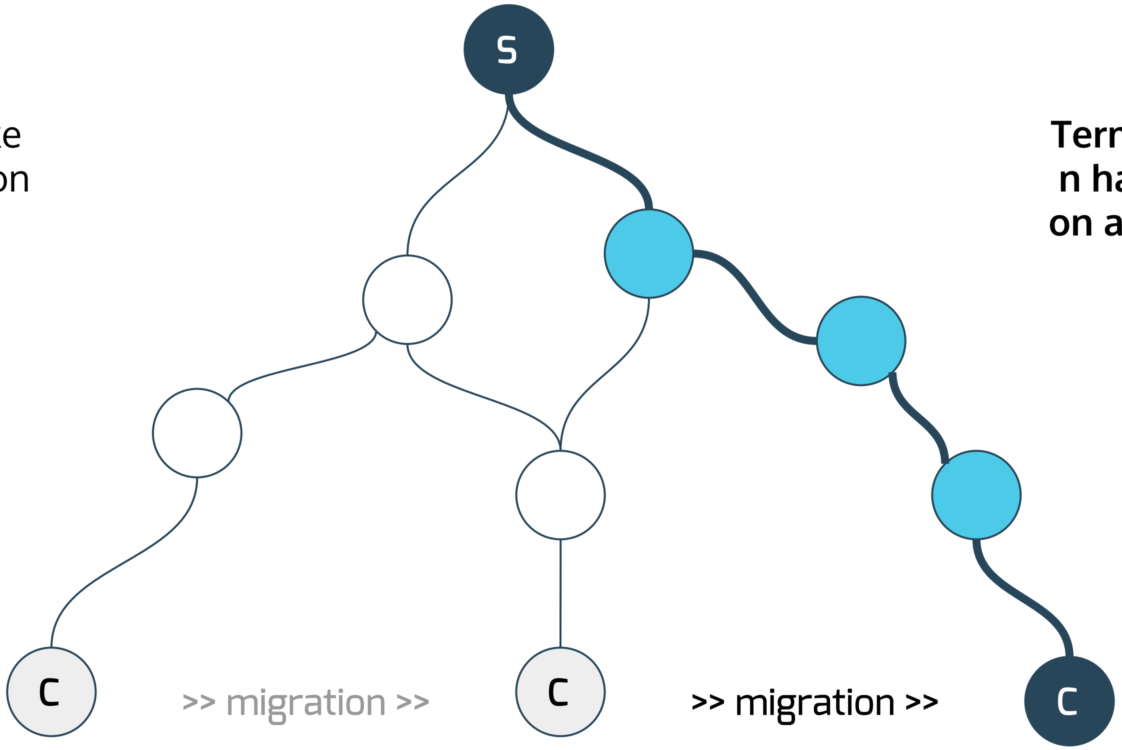


Any number of paths might be used in between

Simple Migration

Handshake happens on one path

Terminatio n happens on another



Any number of paths might be used in between

Limited Scope *Stateless* Reset

An end-to-end signal

Used only when a server (not a client) loses state

Terminates the connection

Not visible to middleboxes (?)

Signal Leakage

As originally designed, Public Reset is an end-to-end signal

… that leaks information to the path

Connection termination also means flow termination

Path elements have an incentive to look for and consume these packets

QUIC

Acting on Partial Information

A path element might act on a spoofed Stateless Reset

That could break a flow, even if the signal is not genuine

TCP RST is used for man-on-the-side DoS attacks

…it would be nice if QUIC weren’t similarly vulnerable

QUIC

Solution Options

A (#20): Expose the verifier and have path elements validate

Problem: path elements won't see the handshake always

Problem: they might only look at the packet type octet

B (Grease): Send lots of fake Stateless Resets

...with (**B1**) or without (**B2**) a publicly visible verifier

Problem: wastes bandwidth and effort

C (Hide): Make the Stateless Reset look like any other packet

QUIC

Proposal: Remove the Leakage (Option C)

Send ***n*** during the handshake, **encrypted**

Stateless Reset looks like a regular packet

Contents are ***n*** plus random padding

Looks like ciphertext but won’t decrypt

- Client compares packet to ***n*** if it doesn’t decrypt

Server generates ***n*** from a static key and connection ID

- e.g., HKDF(K_{static}, connectionID || serverID, ‘reset’, L)

QUIC

Wait!

What about the path?

The only signal the path gets is the handshake

  …and that is only for the first path

For other paths, it's either packets flowing or not

  That means timers, and timers are terrible

Please, propose a separate, explicit end-to-path signal