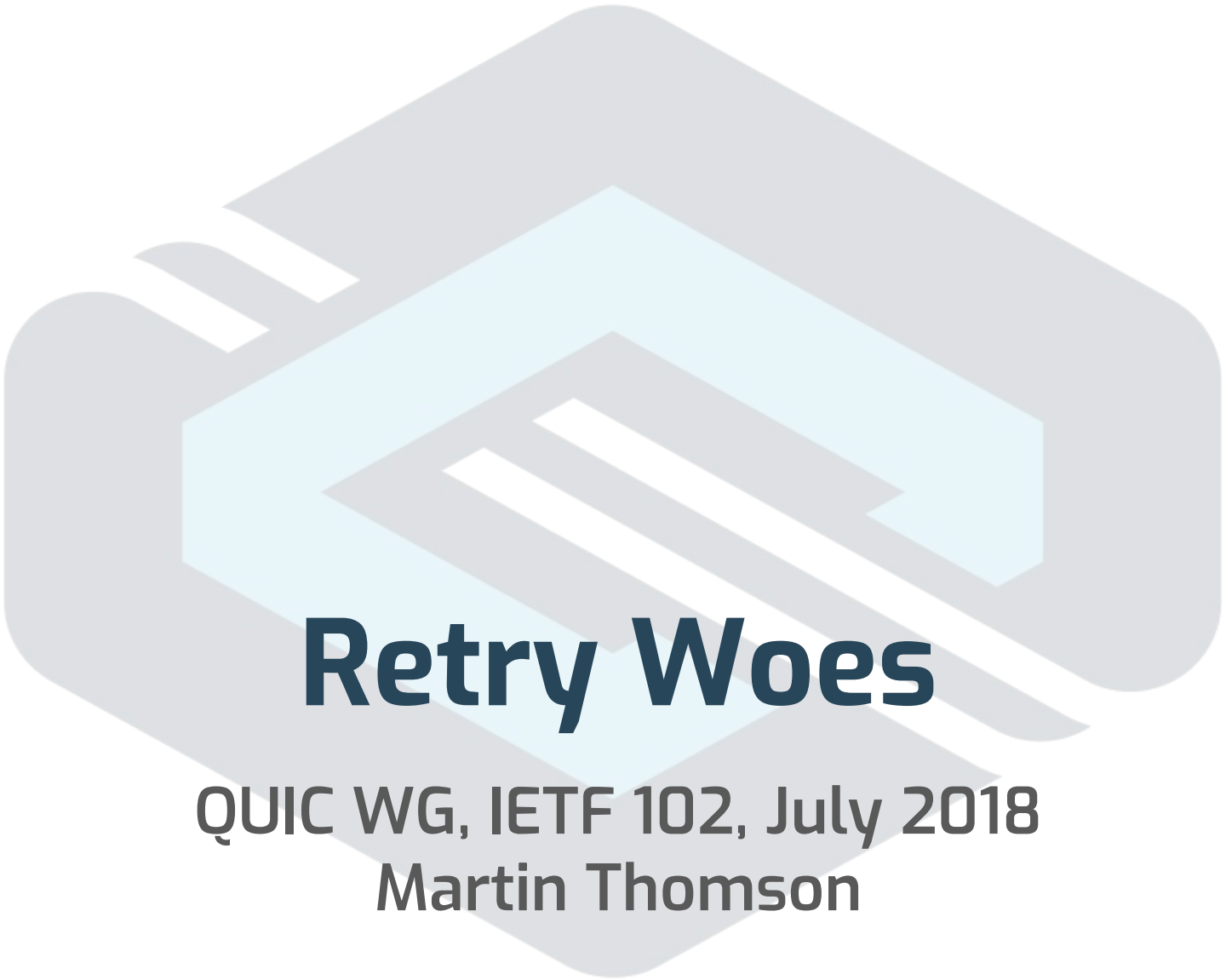


Retry Woes

QUIC WG, IETF 102, July 2018

Martin Thomson

Retry Characteristics

Purpose: stateless client address validation

Design: token-based

Strengths: simplicity

Weaknesses: spoofing, format, looping, 0-RTT, coalescing

Retry Overview

Client sends Initial

Server sends Retry with a token

Client sends another Initial with the token

Retry includes the (random) connection ID from the Initial

Retry can be sent multiple times (clients MUST permit 3)

Format

0xfe	DCIL/SCIL	DCID ...	SCID ...
Length (i)	PN	PN ?????	PN ?????
PN ?????	ODCIL	ODCID ...	Token ...

-13 is a mess

Retry isn't encrypted, but it (apparently) includes a packet number, which is (?) encrypted

Proposal: Don't include a length or packet number field

0xfe	DCIL/SCIL	DCID ...	SCID ...
ODCIL	ODCID ...	Token ...	

New weakness: can't coalesce Retry

New proposal: don't worry about that

Looping

Clients accept multiple Retry packets

There is no reliable way to distinguish Retry from different flights of messages

If a client retransmits the Initial, they might get multiple Retry packets

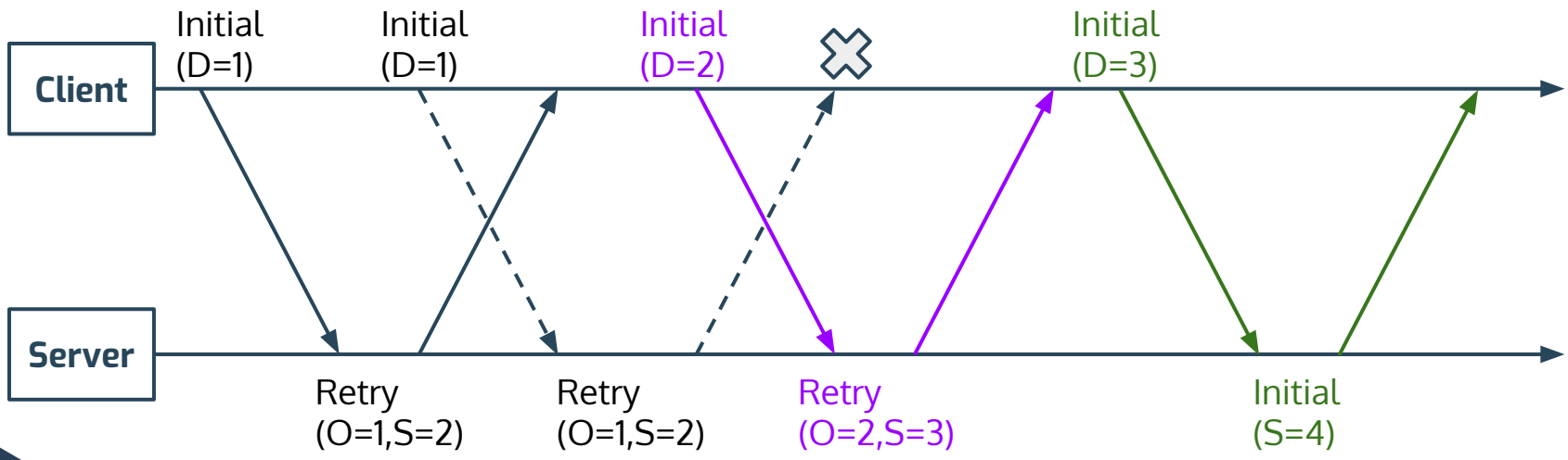
If the Retry from one iteration is reordered relative to another iteration, the handshake can regress - and might fail

Proposed Looping Fix

A non-terminal server MUST provide a new connection ID

A non-terminal server is one that might send another Retry

This means that every Retry will have a unique “Original Destination Connection ID” field



Spoofing

An attacker can spoof a Retry if they can see the Initial

This can cause connection failures either from too many retries, or invalid tokens

An attacker can also alter the connection ID

For instance, a MitM attacker can provide a Retry with its choice of connection ID and strip the token from the subsequent Initial

Spoofing Proposal

Firstly, we don't have to do anything because we don't promise any protection for an attacker with these capabilities during the handshake, but...

If you see an Initial from a server, you can use it

... even if you have sent an Initial in reaction to a Retry

0-RTT and Retry

Question: can a client re-attempt 0-RTT after Retry

Proposal: Yes. Also after version negotiation

Rationale: don't prohibit without strong justification

Catch: Need to resend 0-RTT packets

...with new packet numbers

0-RTT and Retry

Q: Can a server send Retry if it receives a 0-RTT packet?

Draft currently prohibits this

Support for doing so is weak

The case for prohibition is probably equally weak

~~Proposal: ¯_(ツ)_/¯~~

Proposal: SHOULD NOT rather than MUST NOT

Rationale: don't prohibit without better justification