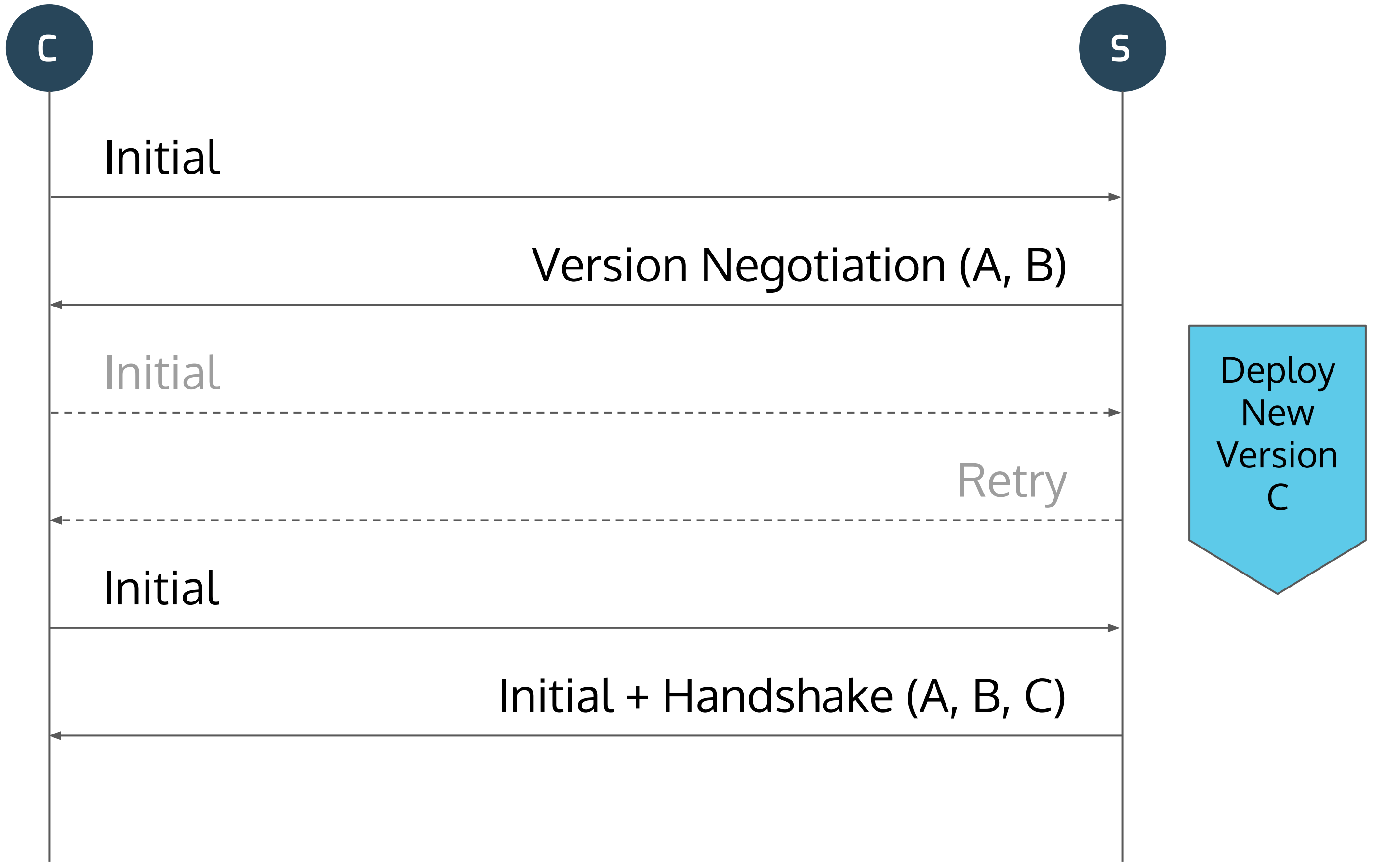
Version Negotiation

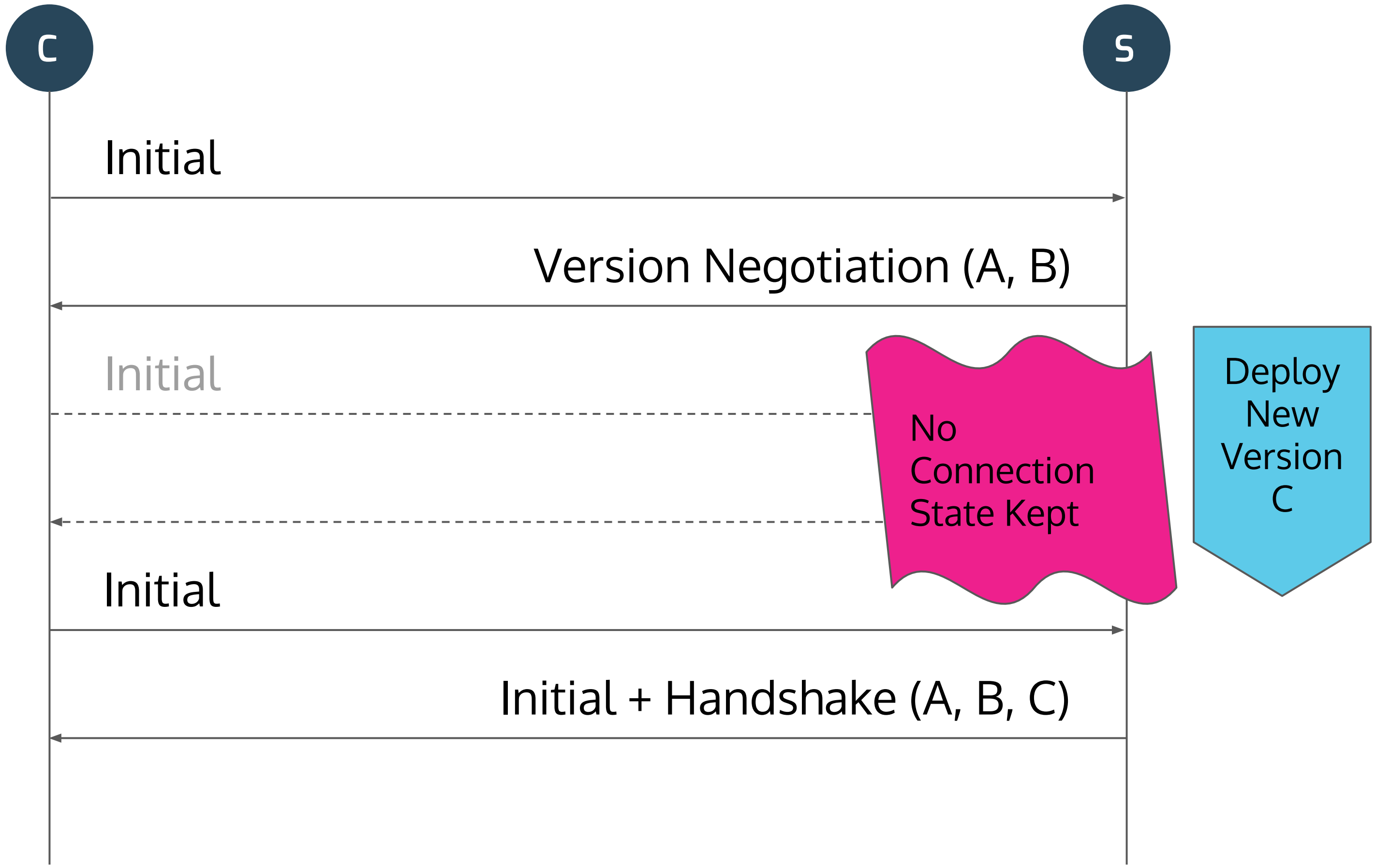
QUIC, IETF 103, November 2018

Martin Thomson

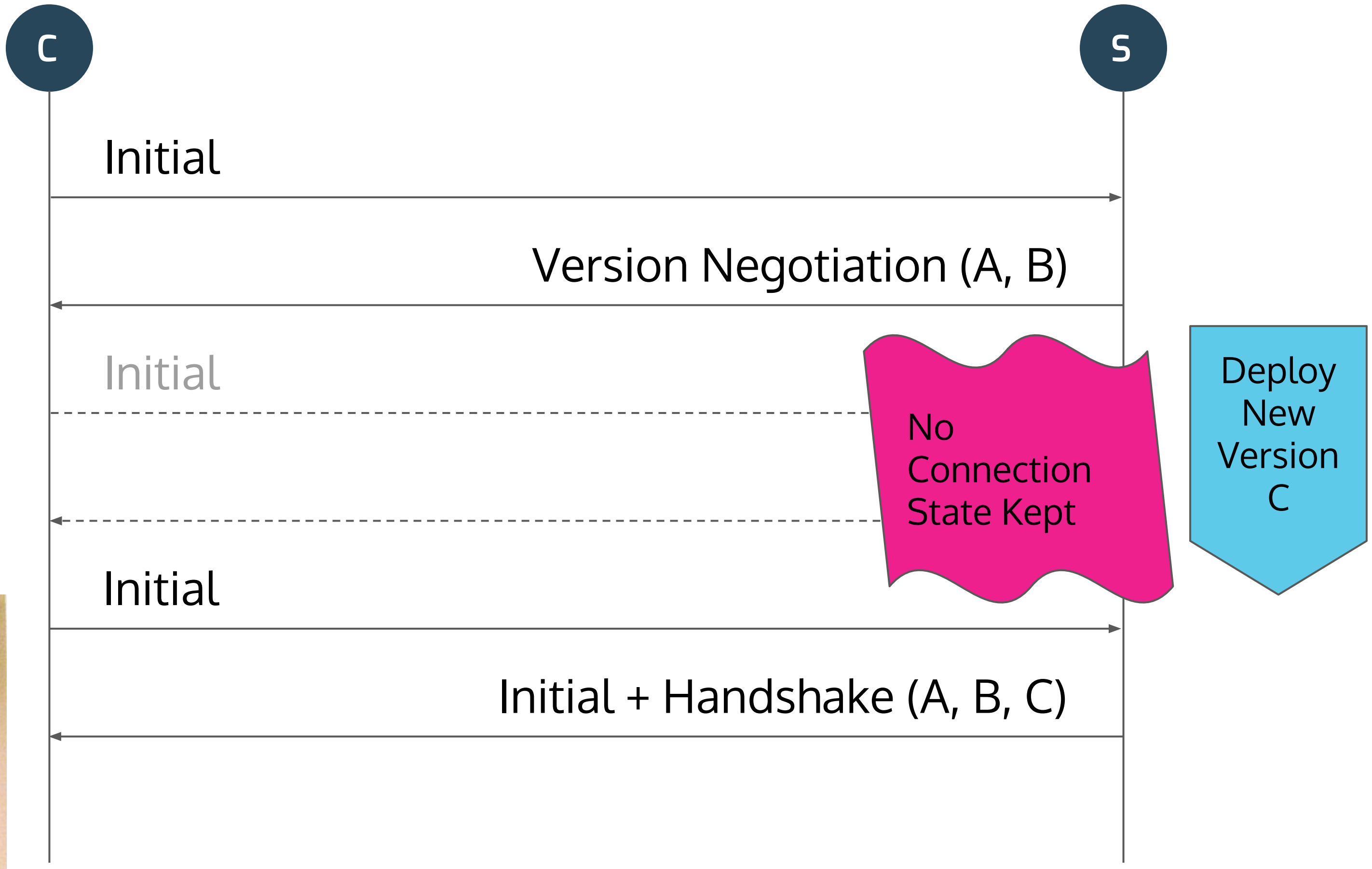
Problem

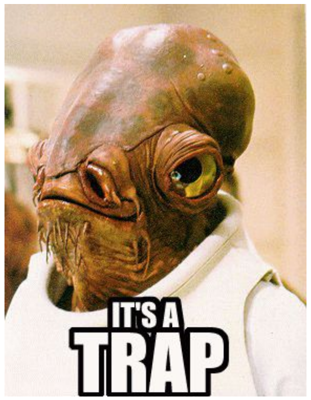

C
S
Initial
Version Negotiation (A, B)
Initial
Retry
Initial
Initial + Handshake (A, B, C)
Deploy New Version C

Problem


C
S
Initial
Version Negotiation (A, B)
Initial
No Connection State Kept
Deploy New Version C
Initial
Initial + Handshake (A, B, C)

Problem


C
S
Initial
Version Negotiation (A, B)
Initial
No Connection State Kept
Deploy New Version C
Initial
Initial + Handshake (A, B, C)


IT'S A TRAP

Fix

The client never has this problem

- The client always has state for a connection

Client transport parameters include final version choice

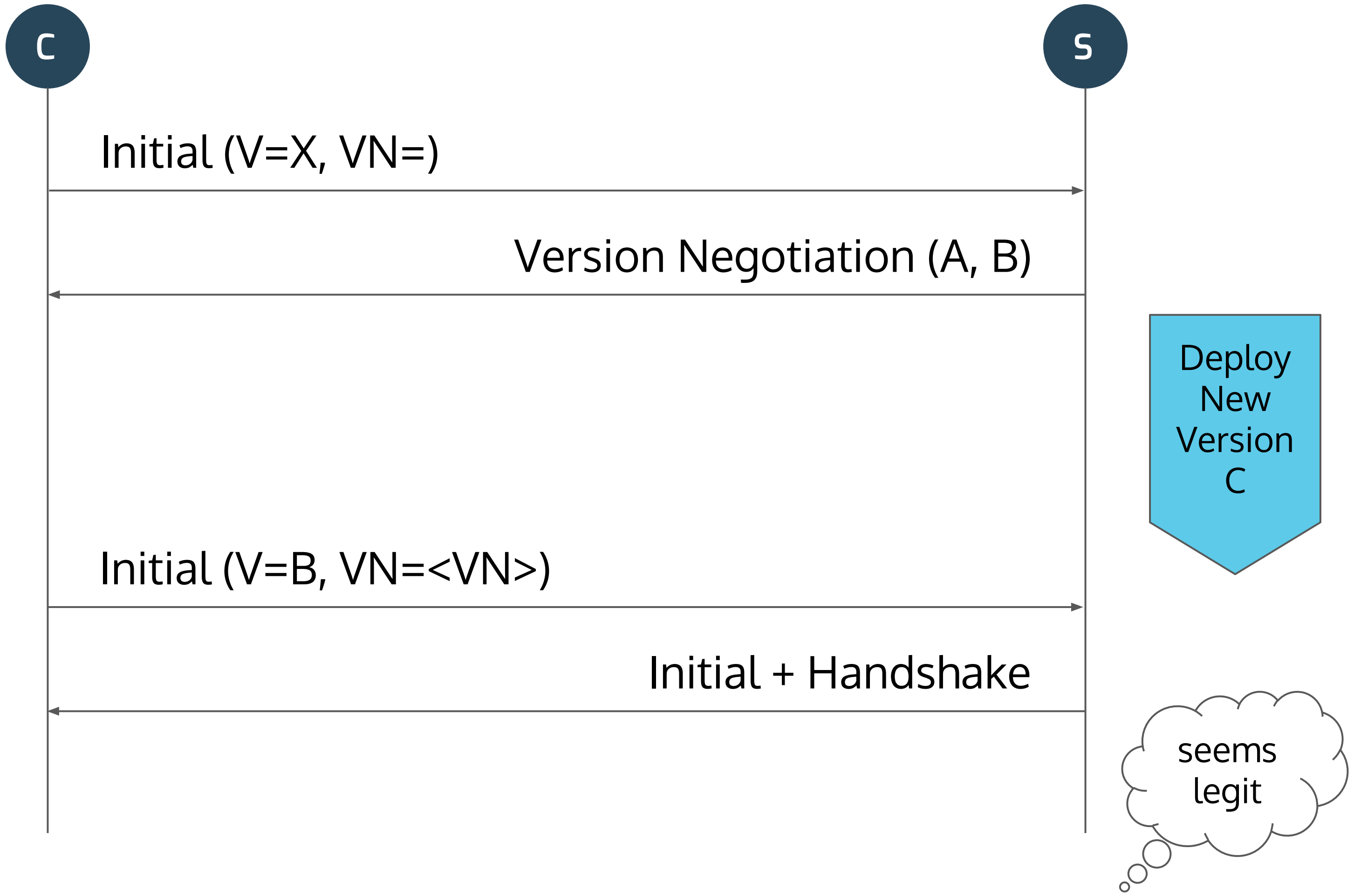
- … plus something to authenticate version negotiation

The server validates versions

Servers allow for versions that are in flux when validating

- e.g., allow either “A, B” or “A, B, C” while C is rolling out

Fix


C
S
Initial (V=X, VN=)
Version Negotiation (A, B)
Deploy New Version C
Initial (V=B, VN=<VN>)
Initial + Handshake
seems legit

How to Authenticate Version Negotiation packets

A. Send a copy of the packet
B. Send a hash of the packet
C. Send a list of versions disabled as a result

Assertion: these are functionally equivalent

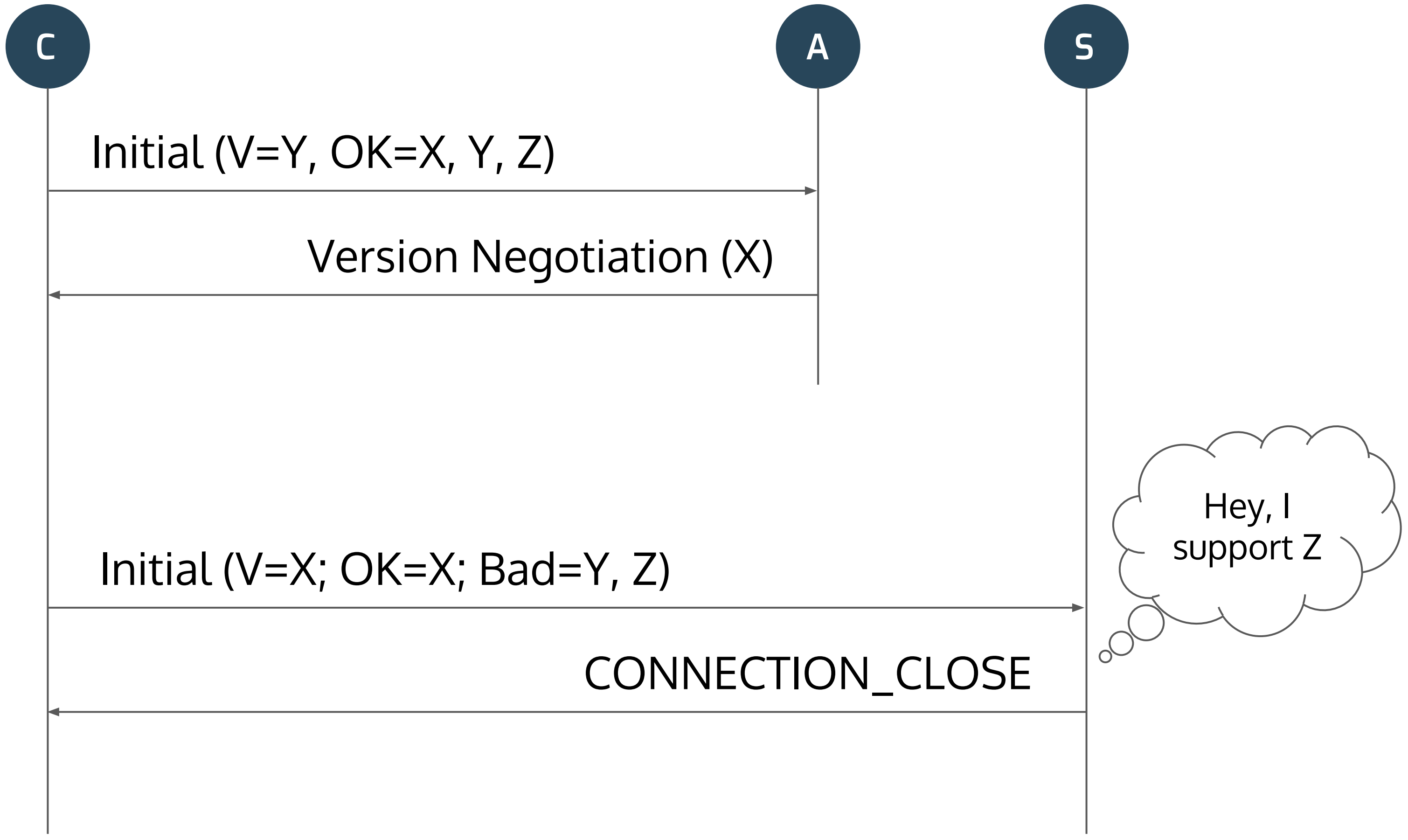
A copy might be big-ish

Hashes capture order, which we don’t care about

Hashes capture greasing versions, confounding validation

Proposal: C

Validating Version Negotiation



Problem/Opportunity

Version Negotiation takes a round trip

Incentive for clients to pick a version that is widely deployed

- As opposed to what they prefer most

Connections get stuck with old versions

Upgrading to a newer version next time possible with new signaling, but clients have to remember (see also tracking)

Alt-Svc/ALTSVC helps HTTP, but maybe not other protocols

Proposal

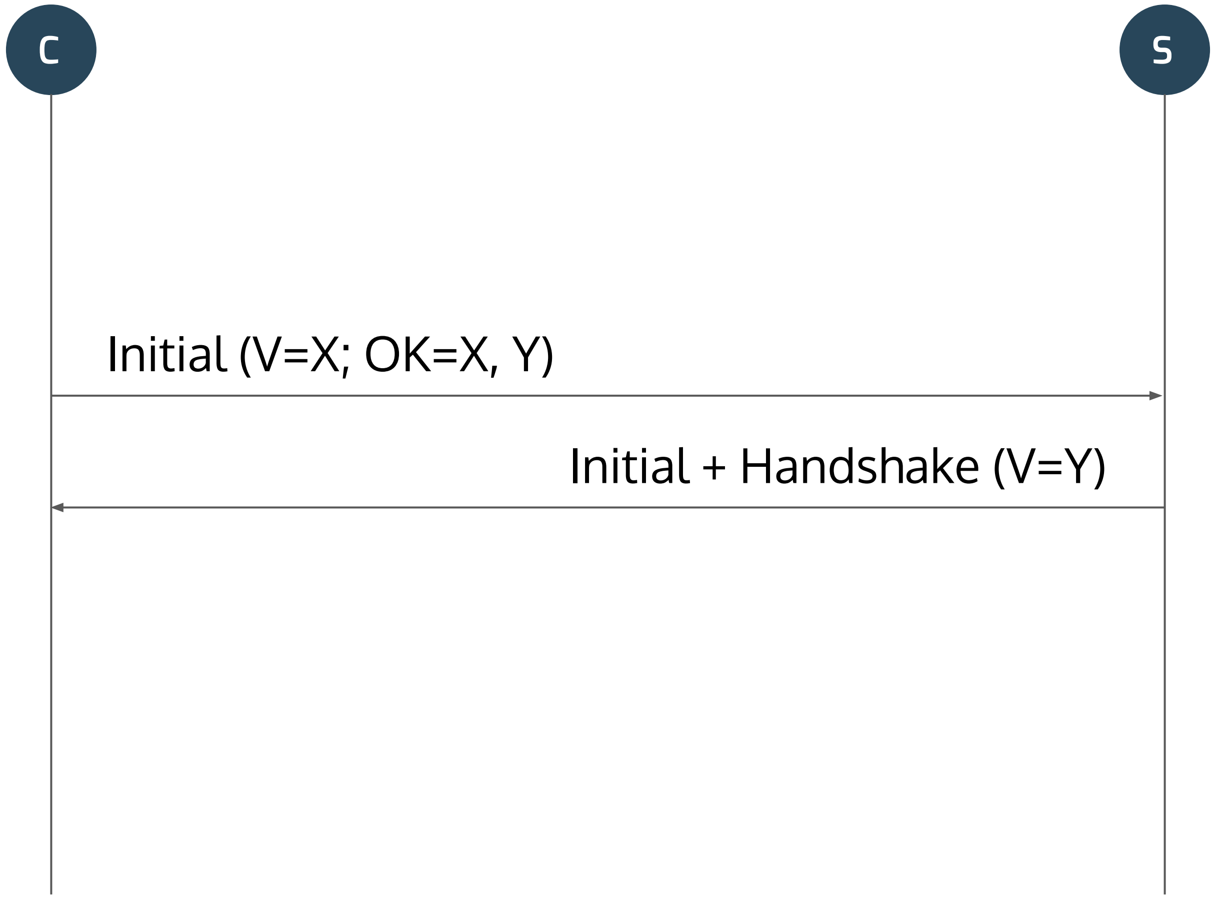
Client advertises all versions it supports

After version negotiation, client lists what it learned:

- Could be the versions from the VN packet
- …or the versions the client eliminated as a result of VN

Server can pick any “compatible” version to continue with

Compatible Upgrade



“Compatible”

Loosely, B is compatible with A if the first packet from A can be used to continue with B

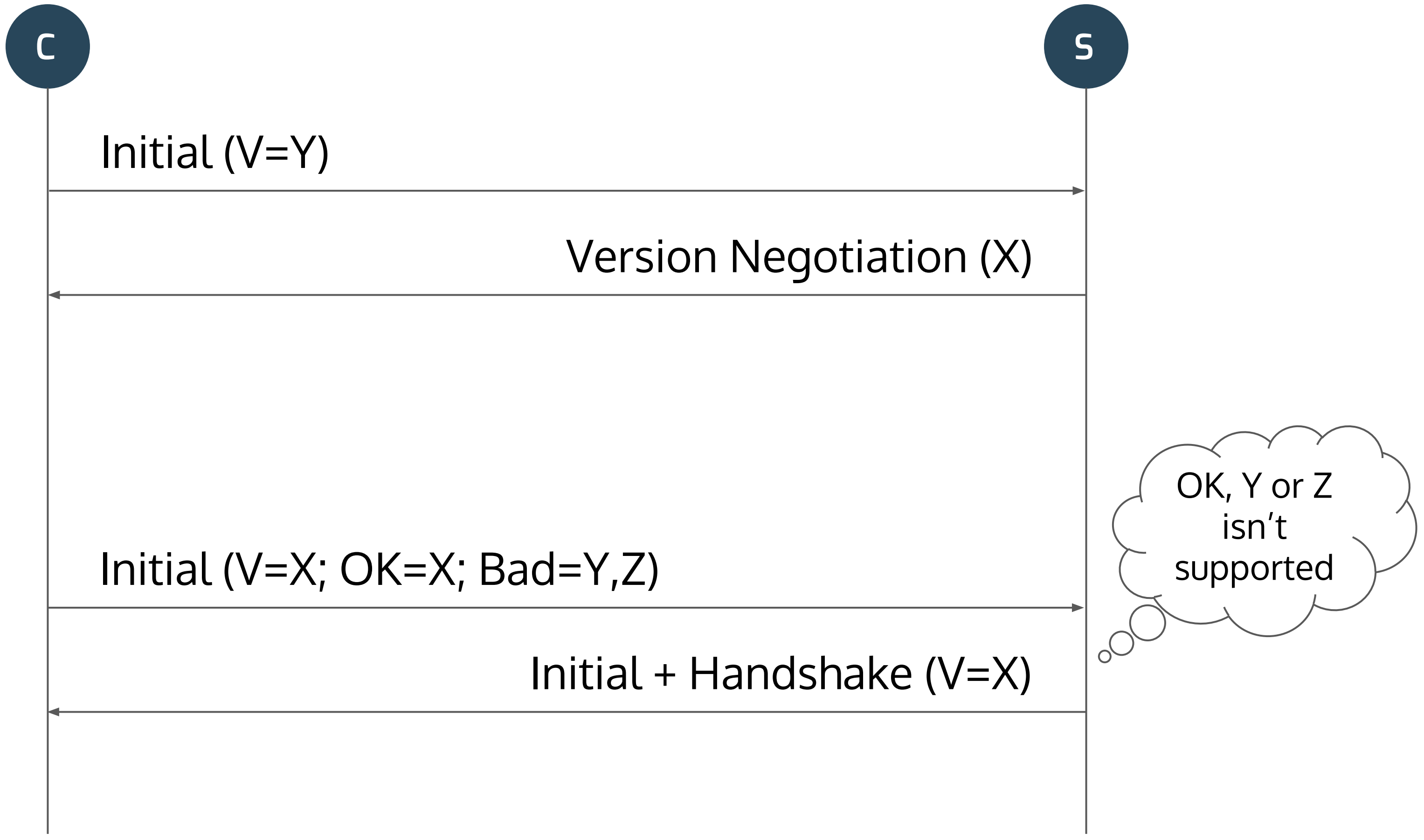
Generally, if a transform exists:

$$\text{first\_packet}_B = f(\text{first\_packet}_A)$$

… then B is compatible with A

The spec for B has to define any transforms to B

Validating Version Negotiation


C
S
Initial (V=Y)
Version Negotiation (X)
Initial (V=X; OK=X; Bad=Y,Z)
Initial + Handshake (V=X)
OK, Y or Z isn't supported

Why?

Deploying a final RFC version (0x1) will incur a latency penalty if the last draft (0xff000011) is widely deployed

Clients will offer (and get) the boring 0x1 version when we start deploying the fancy multipath-enabled 0x2 version

- Multipath might not be widely deployed
- And it might not be worth risking paying a round trip