Unidirectional Streams

QUIC Interim 2017-06, Paris

Why

The transport becomes more generic, simpler

A litany of small issues:

- #515 - the server can't speak first
- #281 - HOLB on server push
  - Client has to send empty STREAM+FIN for server push

Symmetrical Streams

Unidirectionality has an effect on protocols with 1-to-1 mappings between client and server messages

... which might seem like it applies to most protocols

... except HTTP server push, RTP, CoAP, maybe others

1-to-1 and streams-as-messages don't always fit

The cost is that 1-to-1 protocols need an explicit correlator

... stream ID no longer works as implicit correlator

#515

After a 1-RTT TLS 1.3 handshake, the server speaks first

In HTTP over QUIC, the server could speak first, but

- stream 1 is a **client**-initiated stream

We could use stream 2, but

- in 0-RTT the client speaks first

Using stream 1 for 0-RTT and stream 2 for 1-RTT is gross

#281

Server push operates in two stages:

1. the promise stage where new “requests” are created
2. the fulfilment state where responses are sent

In practice, promises are often generated opportunistically

… usually as new resources are “discovered”

Order of promises doesn’t matter

… but fulfilment order is critical to performance

#281 cont.

If a server promises more than the maximum stream number...

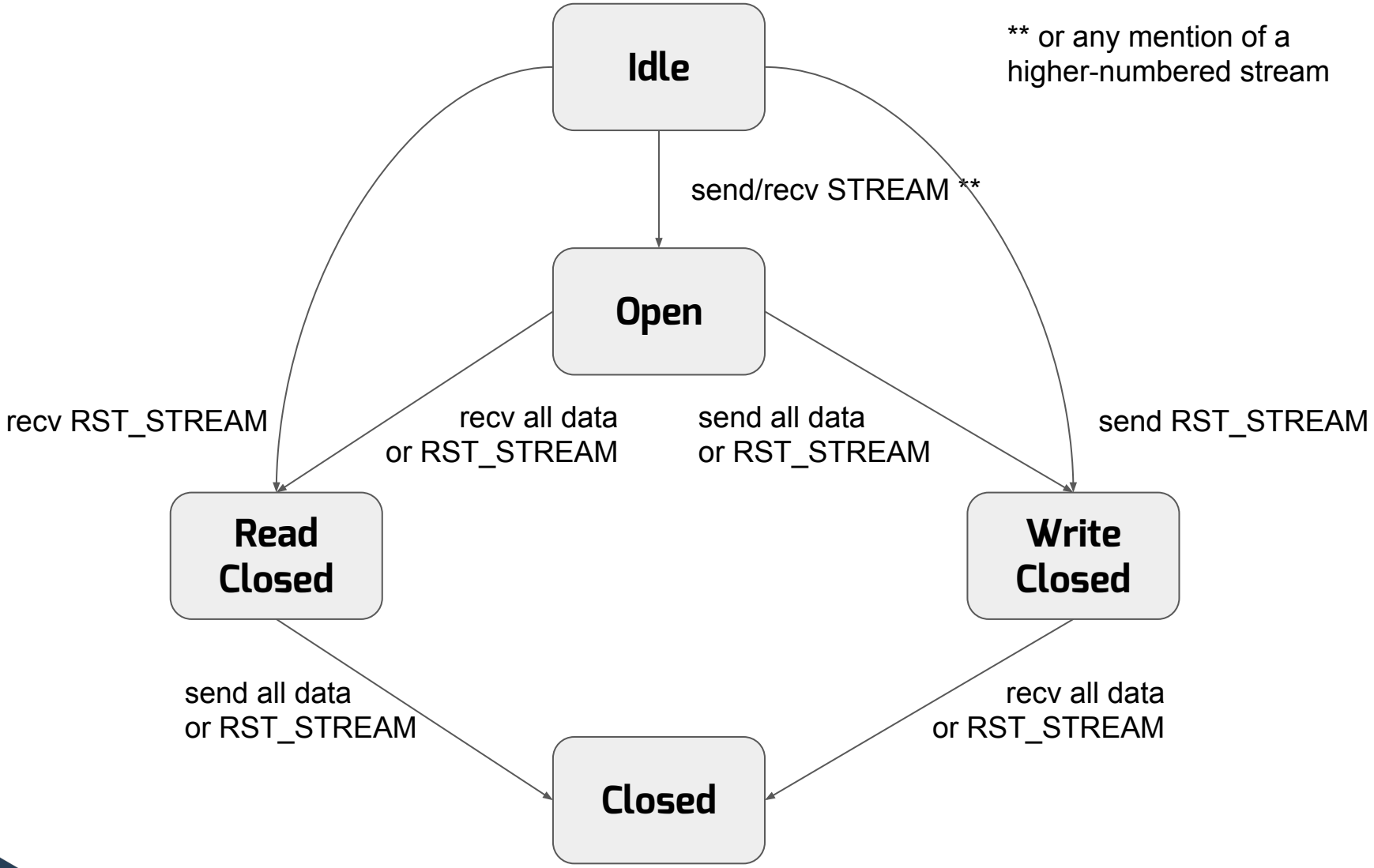
A resource that is “discovered” late can be stalled

If that resource is urgent, it might take an RTT to sort out

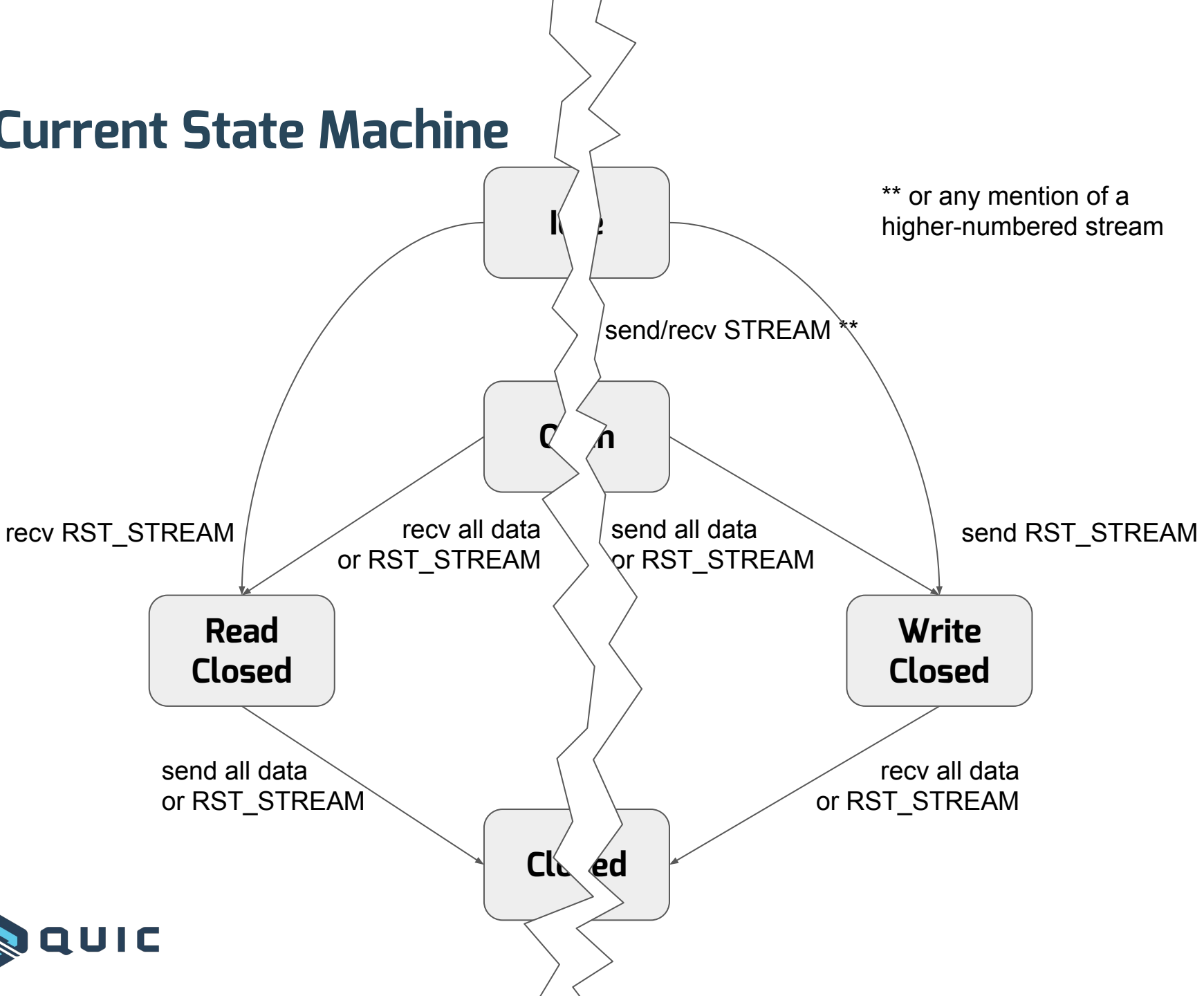
Ideally, promises could be fulfilled in any order

That requires a layer of indirection...

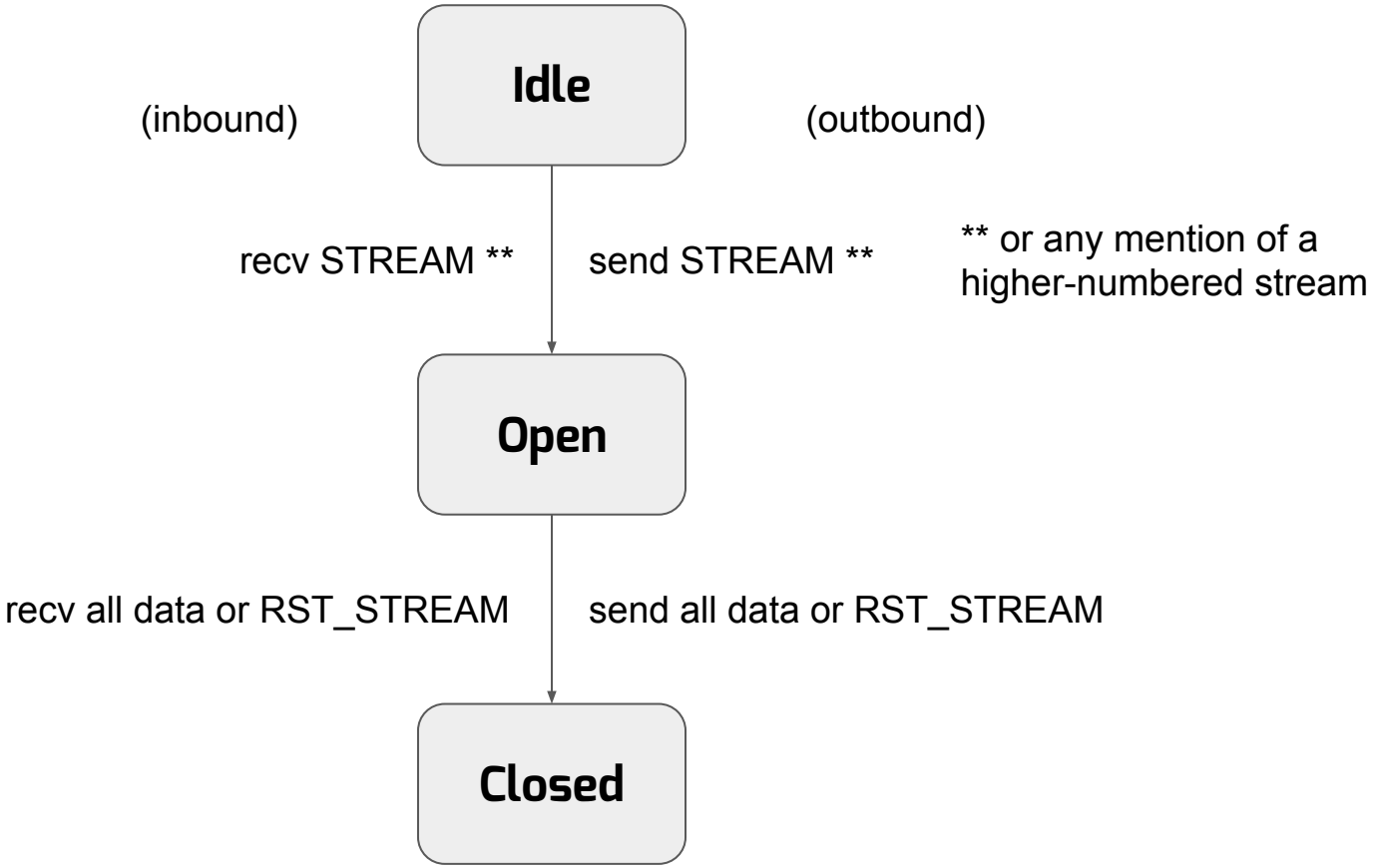
Current State Machine



Current State Machine



Simplification



HTTP Impact

HTTP Mapping Strawman

Changes:

- response streams include the stream ID of the request
- push streams include the stream ID of the response stream that contained the promise, and a promise index
- header streams have a flag if there is a body
- body streams include the stream ID of the headers

maybe see #557

- need a new control frame to cancel a push

Advantages

No empty streams

No odds/evens for streams

... twice as many requests possible 

... more when requests/responses have no body

Push fulfilment in any order

Endpoints can send SETTINGS immediately

Maybe-Negative Consequences

Server push is counted with responses against stream limit

- server now has to decide which to answer

Extra correlators on the start of server streams

- though probably less was used on empty streams
- most will be relative, so we can find an efficient encoding

Non-Obvious Consequences

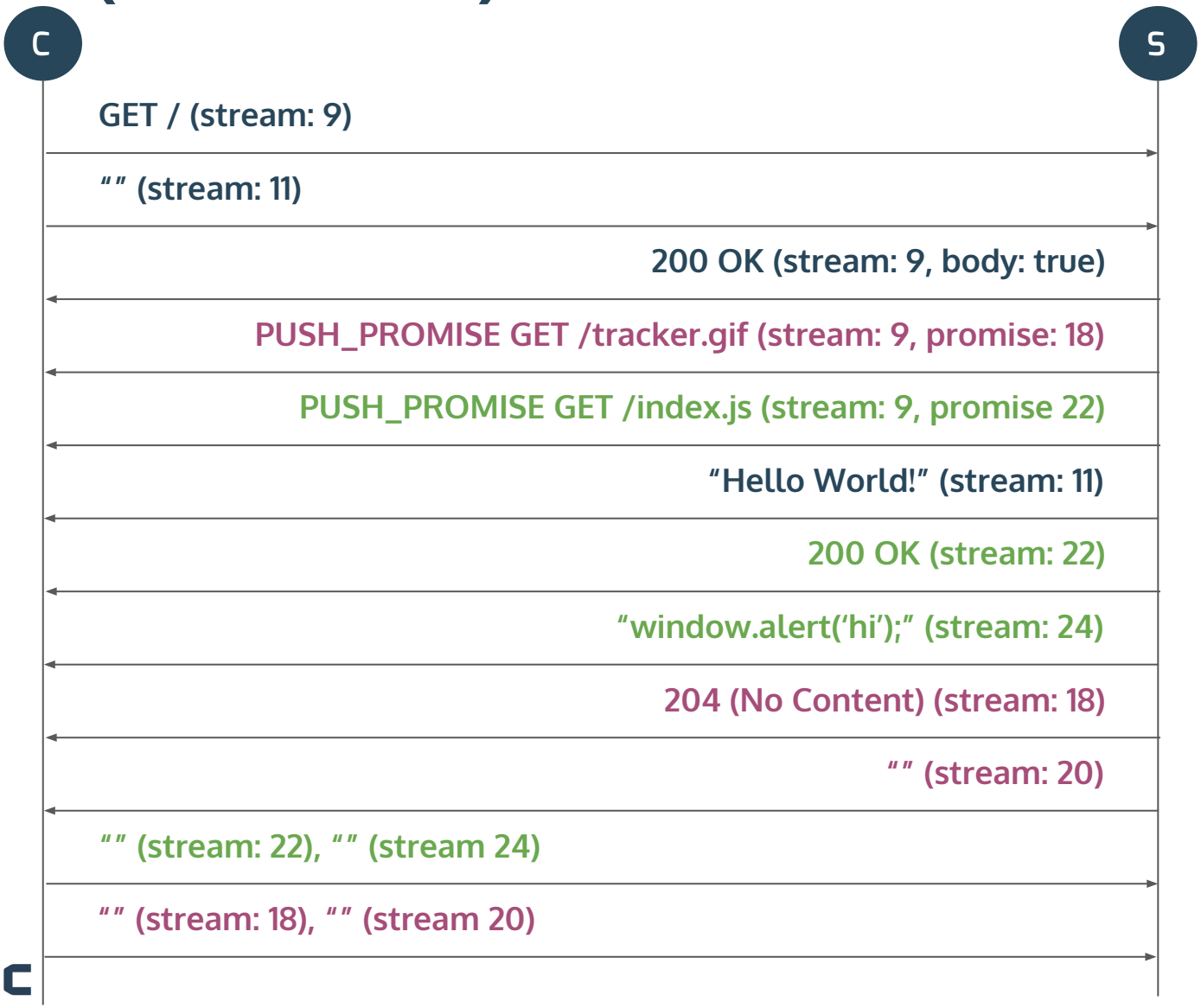
A client can make more requests than the server can answer

With bidirectional streams, client MAX_STREAM_ID governs pushes, which have this exact problem

Recommendation:

- Don’t worry about it, let the server reset requests
- If it hurts, stop: advertise a larger limit

Example (Bidirectional)



Example (Unidirectional)