



HTTP/QUIC

Interim 1806 - Kista

PRINCIPLE:

HTTP/QUIC is conceptually like HTTP/2, but structurally divergent when it makes sense.

HTTP/2 defines ten frame types

DATA

HEADERS

PRIORITY

RST_STREAM

SETTINGS

PUSH_
PROMISE

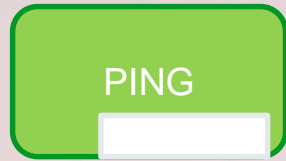
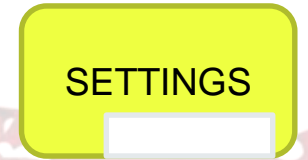
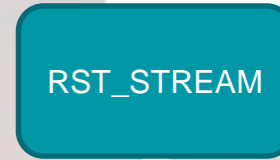
PING

GOAWAY

WINDOW_
UPDATE

CONTINUATION

Six HTTP/2 frames use the Flags field



...so it's defined as part of the base frame layout.

HTTP/QUIC eliminates three frame types...

DATA

HEADERS

PRIORITY

RST_STREAM
CANCEL_PUSH

SETTINGS

PUSH_PROMISE

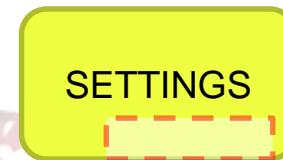
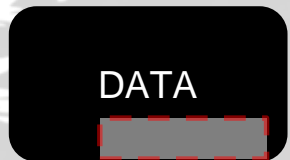


GOAWAY





...and changes which frames use Flags



PROPOSAL:
Define Flags inside PRIORITY,
not in every frame type.

PRINCIPLE:

Application layer shouldn't need to “grab” a particular stream by ID

Stream grabbing in prior versions of HTTP/QUIC

- PUSH_PROMISE includes Stream ID
- Control stream

Stream grabbing in HTTP/QUIC -12

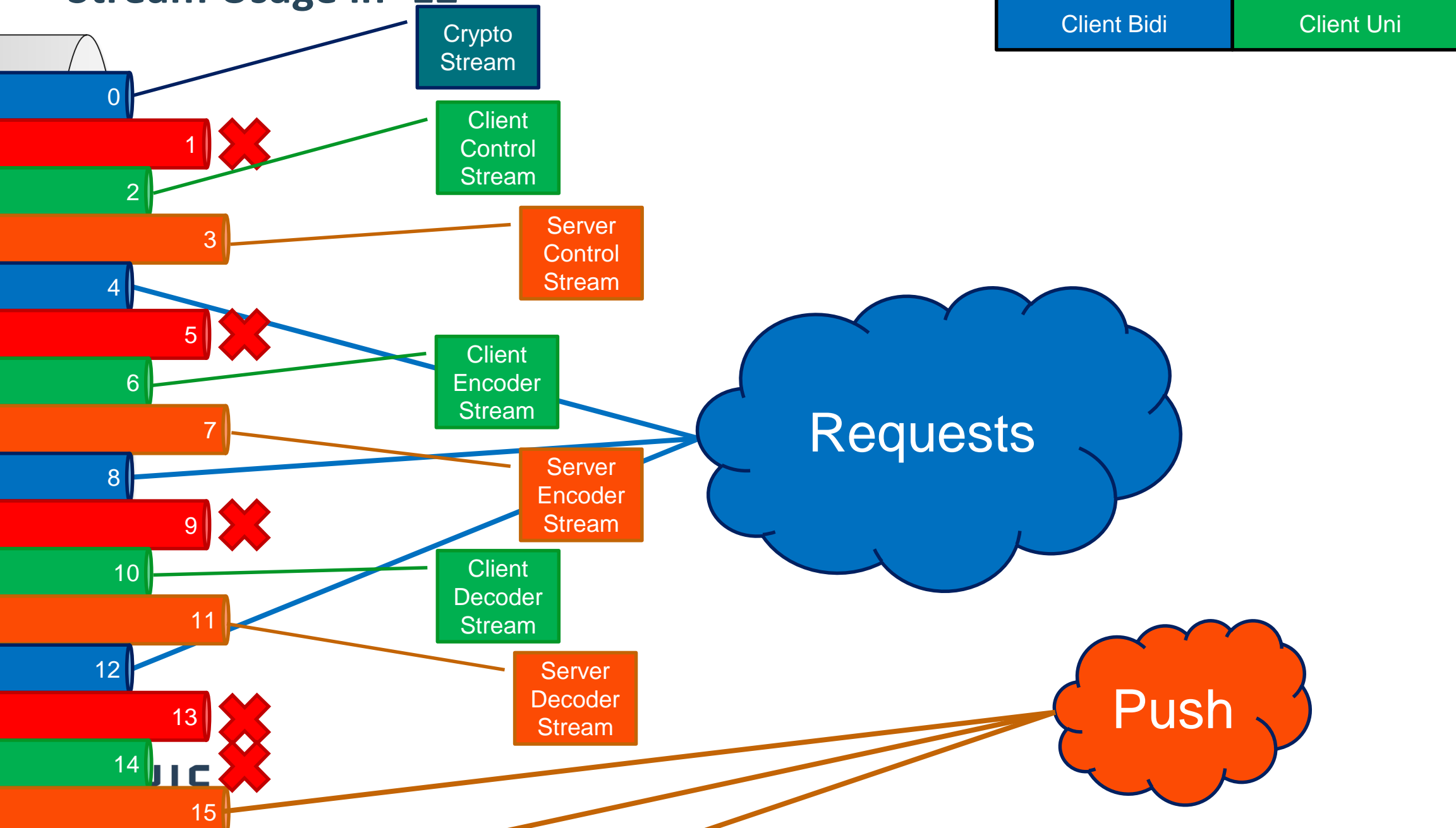
- ~~PUSH_PROMISE~~ includes
Stream ID Push ID, header
- Control stream
- QPACK encoder stream
- QPACK decoder stream

Stream Usage in -12

Server Bidi	Server Uni
Client Bidi	Client Uni

- Bidirectional streams
 - Client-initiated: Request/response (modulo crypto)
 - Server-initiated: Not used
- Unidirectional streams
 - First stream: Control stream
 - Second stream: QPACK encoder
 - Third stream: QPACK decoder
 - All subsequent: Push

Stream Usage in -12



A word cloud background featuring various adjectives and nouns in different colors and sizes. The words are scattered across the entire slide, with some appearing more prominently than others. The colors include shades of blue, green, yellow, orange, and red. The words are of various sizes, with some being significantly larger than others. The overall effect is a dense, colorful collection of text that fills the background of the slide.

PROPOSAL: Self-describing unidirectional streams

Self-Describing?

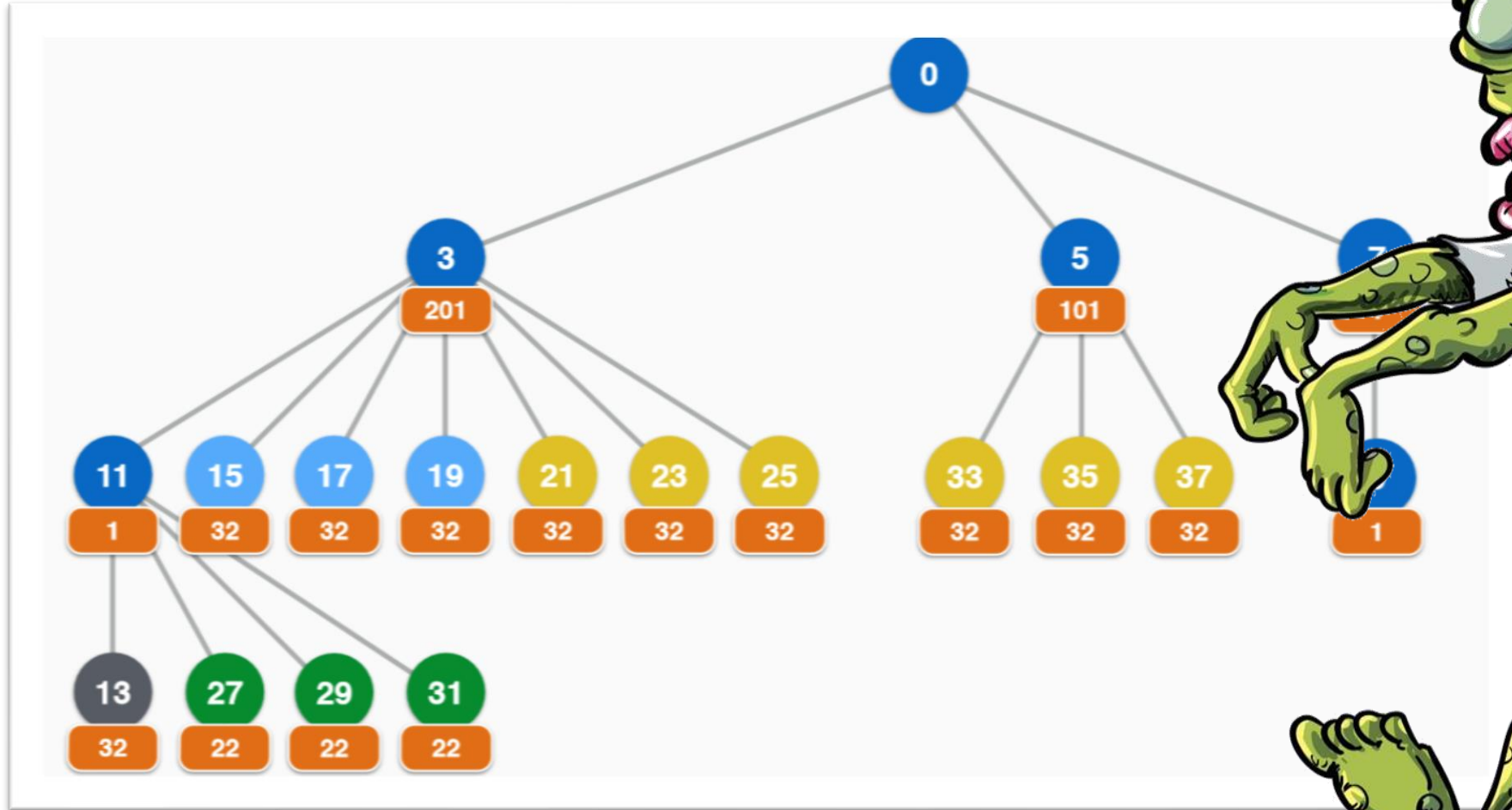
Server Bidi	Server Uni
Client Bidi	Client Uni

- Begin with a type byte
 - If you understand it, keep reading. Four types defined now:
 - Control
 - QPACK Encoder
 - QPACK Decoder
 - Push
 - If not, kill the stream (STOP_SENDING)
- Extensible, similar to frame types
 - Define frame if data is always a single unit
 - Define stream type if data can develop over time

PRINCIPLE:

HTTP/QUIC is conceptually like HTTP/2, but structurally divergent when it makes sense.

ATTACK OF THE ZOMBIE STREAMS



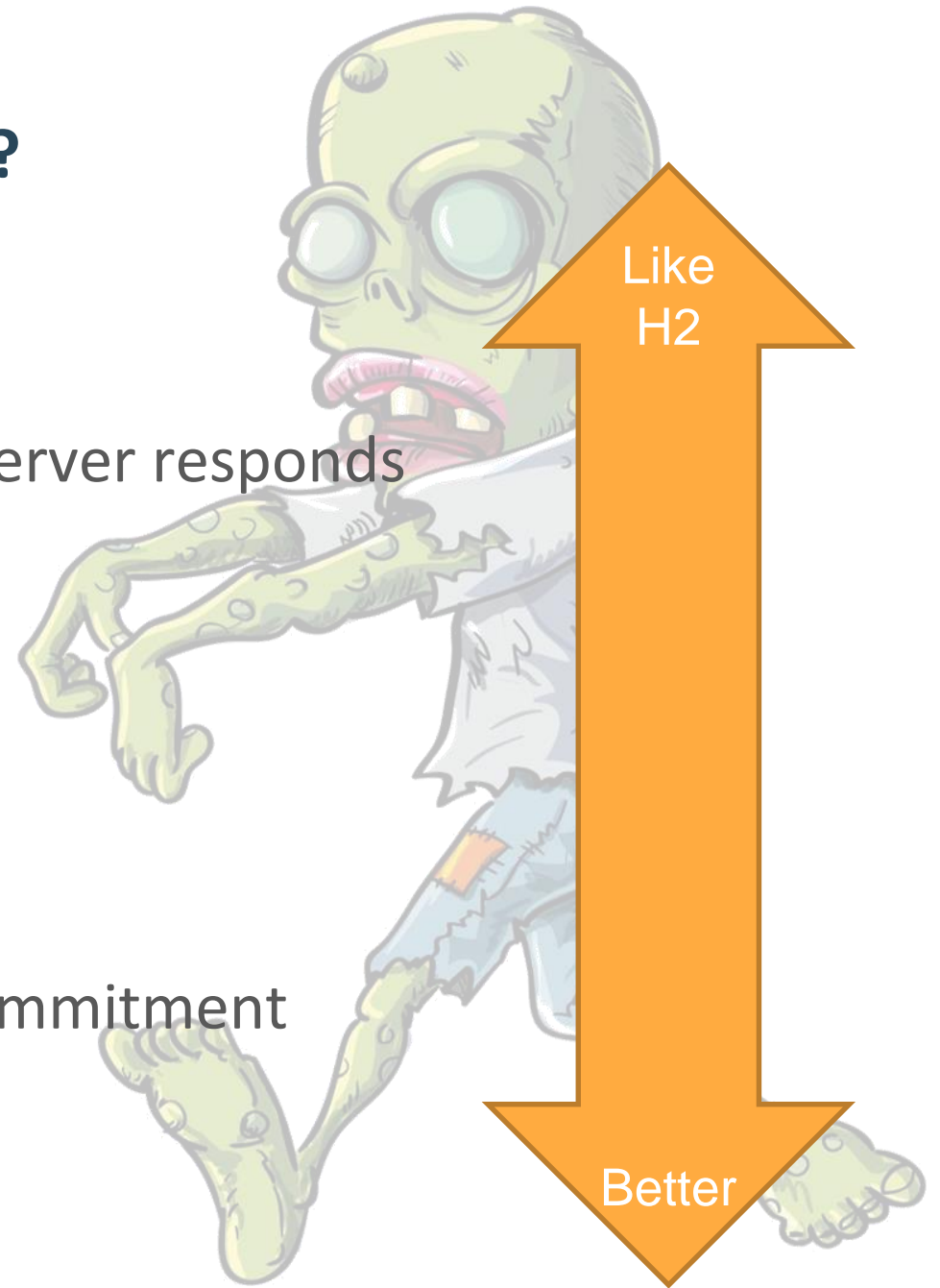
The HTTP/2 method has serious drawbacks...

- Inconsistent client/server views of priority tree if server prunes dead streams
- Unbounded server state commitment if it doesn't

...but none of them are really QUIC-specific, so fixing them might be outside our charter.

How should we represent this in HTTP/QUIC?

- Explicitly close request streams
 - Send QUIC STREAM w/ FIN, Offset=0; server responds
- Don't support priority in HTTP/QUIC
- Introduce placeholders
 - Would permit server to bound state commitment



PROPOSAL: Placeholders

