HTTP/QUIC

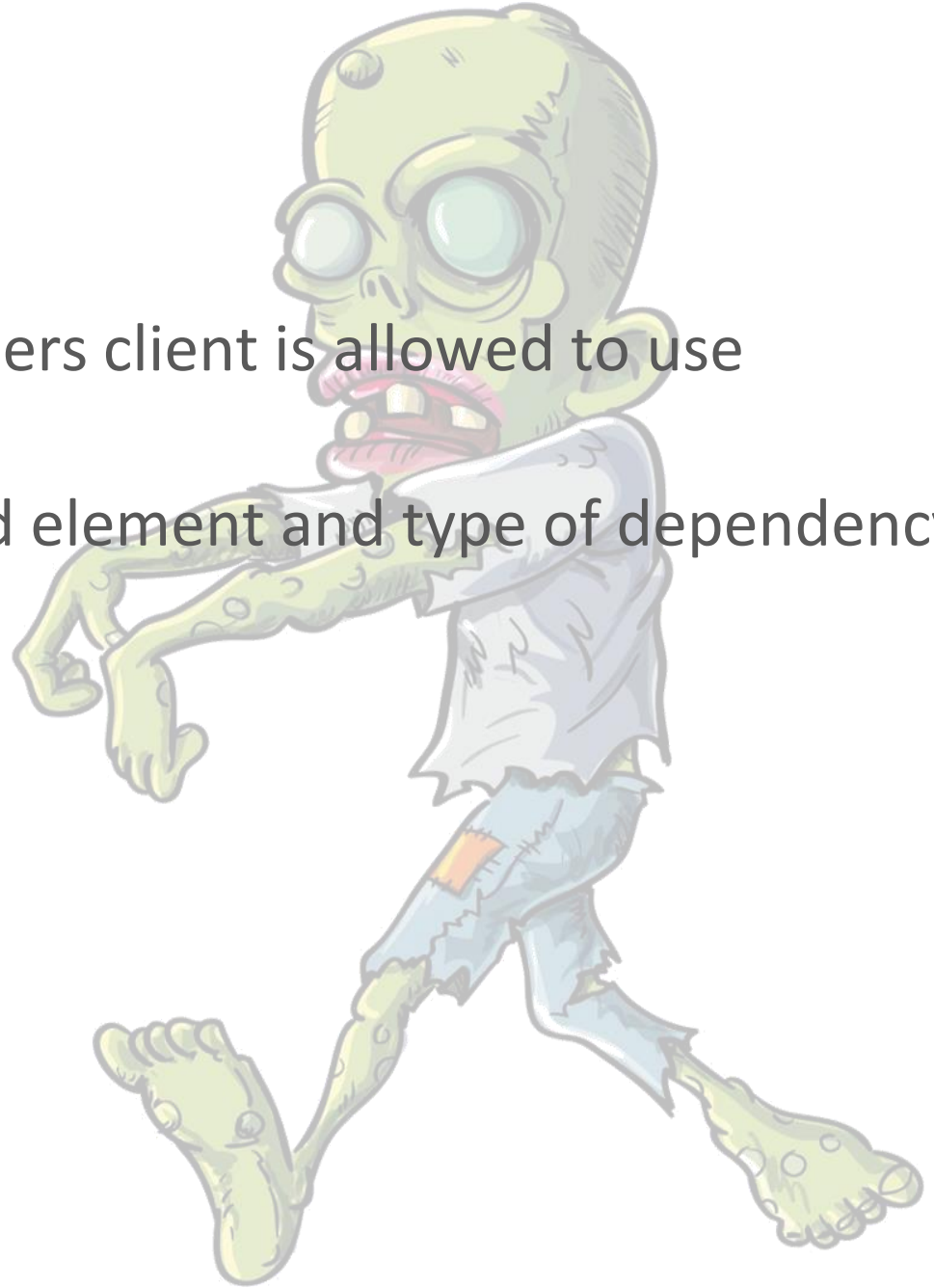
IETF 102

Notable Changes since London

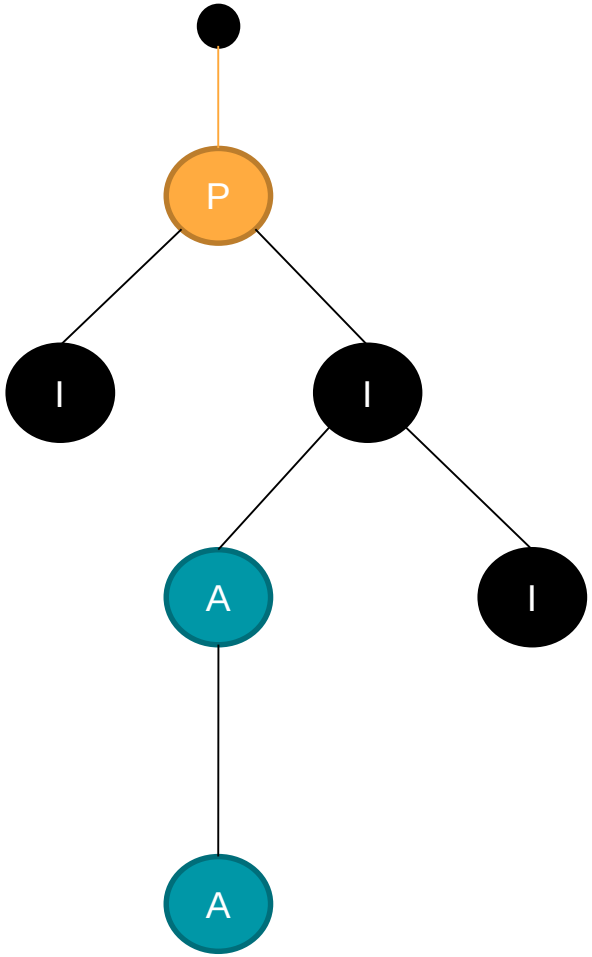

Flags defined inside PRIORITY,
not in every frame type.

Placeholders in the PRIORITY Tree

- Server setting decides how many placeholders client is allowed to use
- PRIORITY frame indicates type of prioritized element and type of dependency
  - Request
  - Push
  - Placeholder
  - Root of tree
    - (0 is a valid request stream now!)
- Permits more aggressive pruning



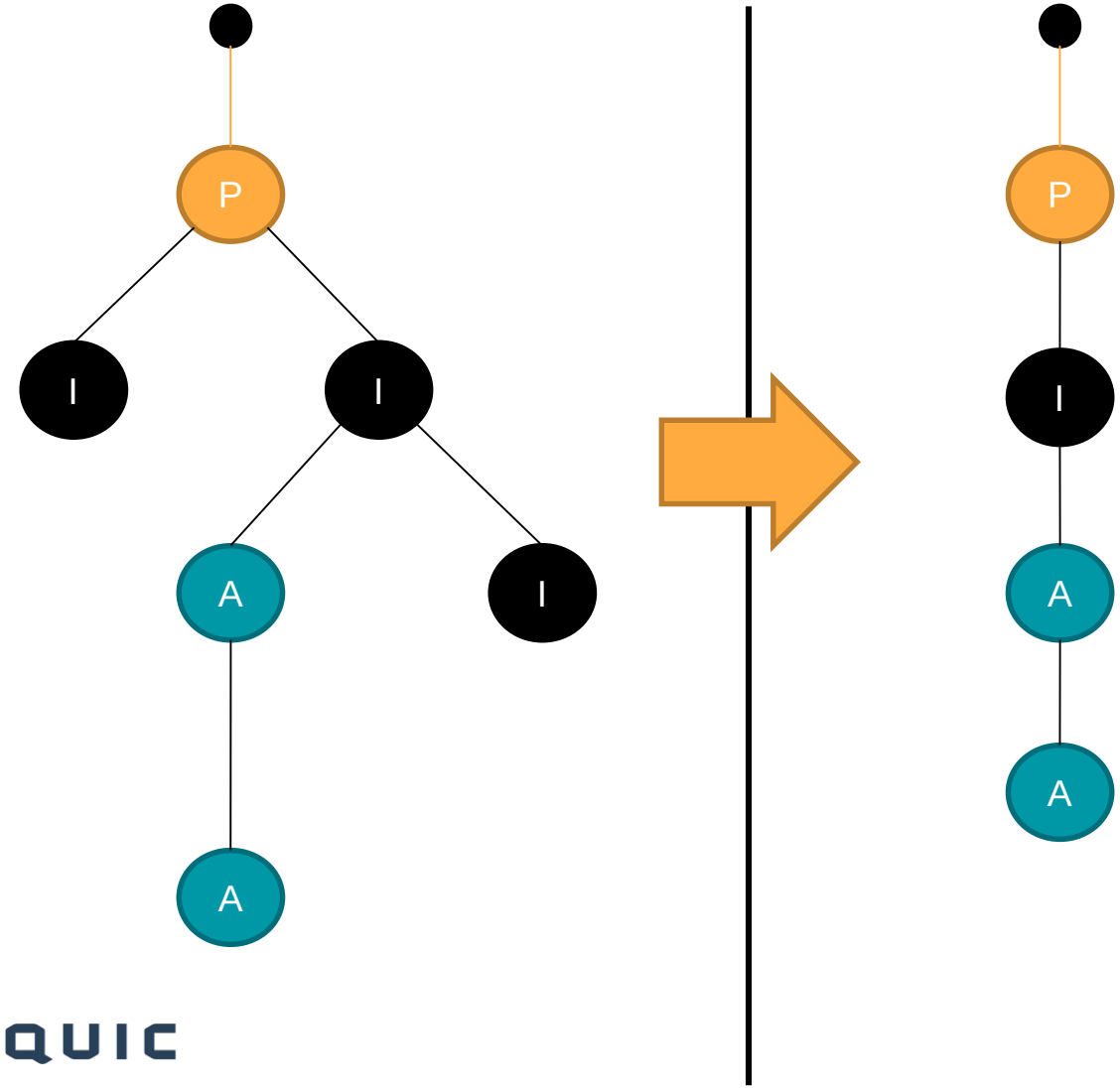
Aggressive Pruning



Active = open or recently closed
Inactive = closed >1 RTT ago

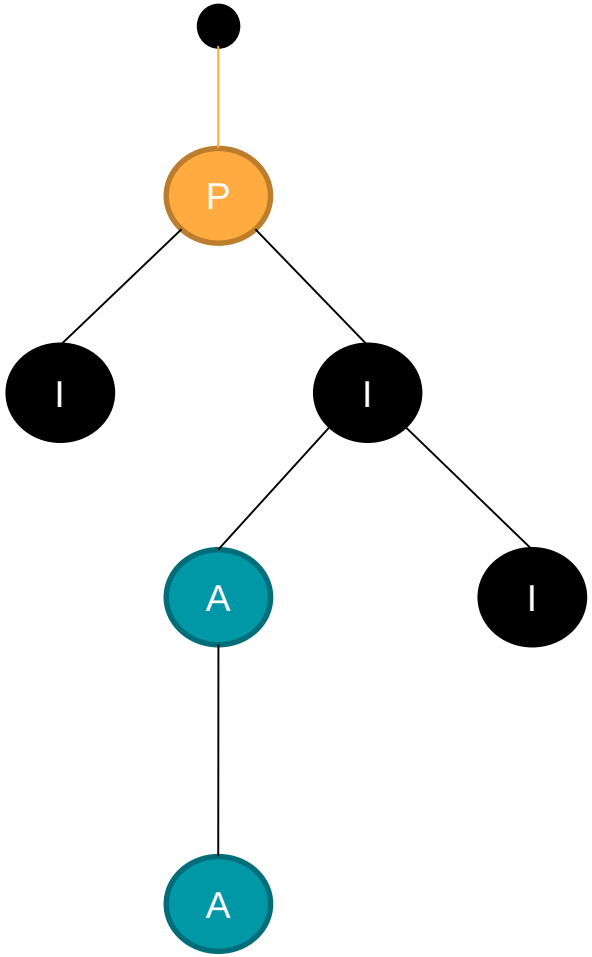
Aggressive Pruning

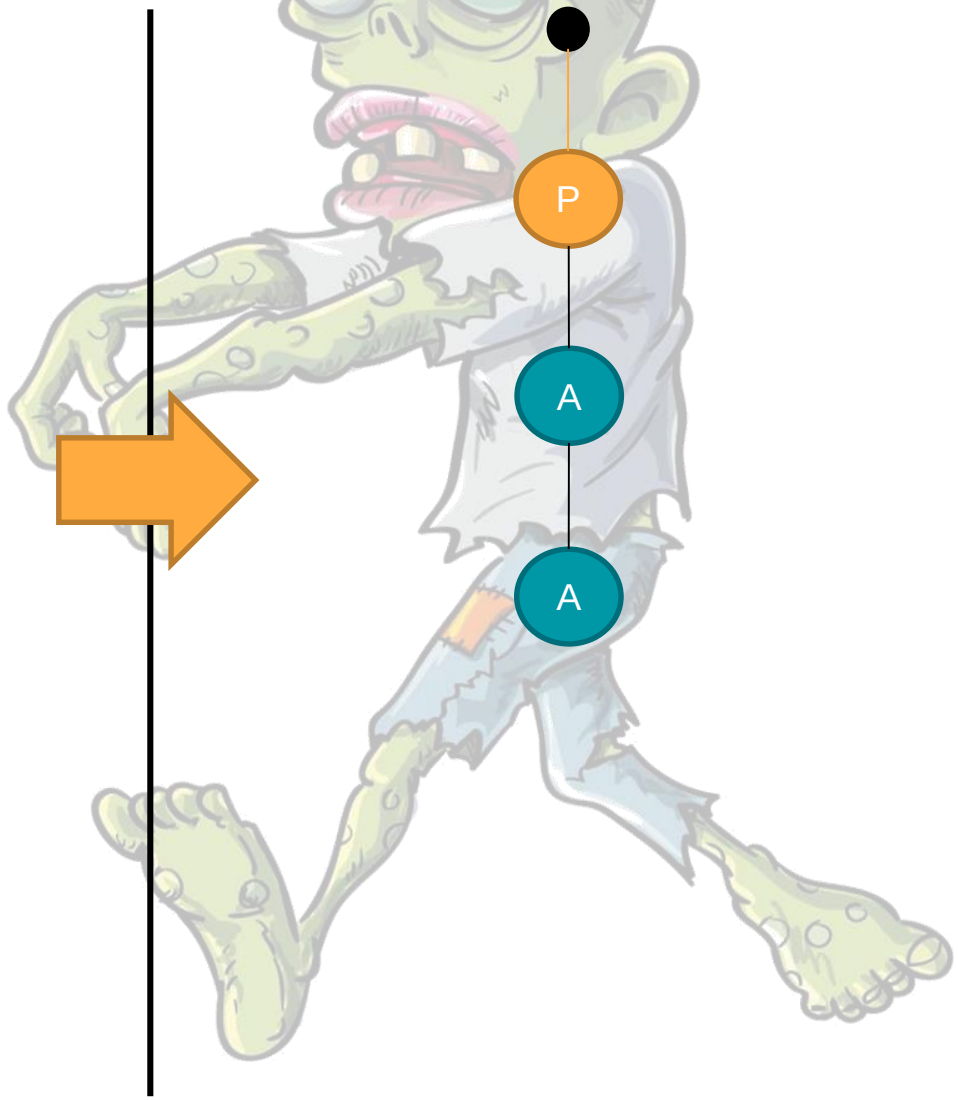
Active = open or recently closed
Inactive = closed >1 RTT ago



Aggressive Pruning

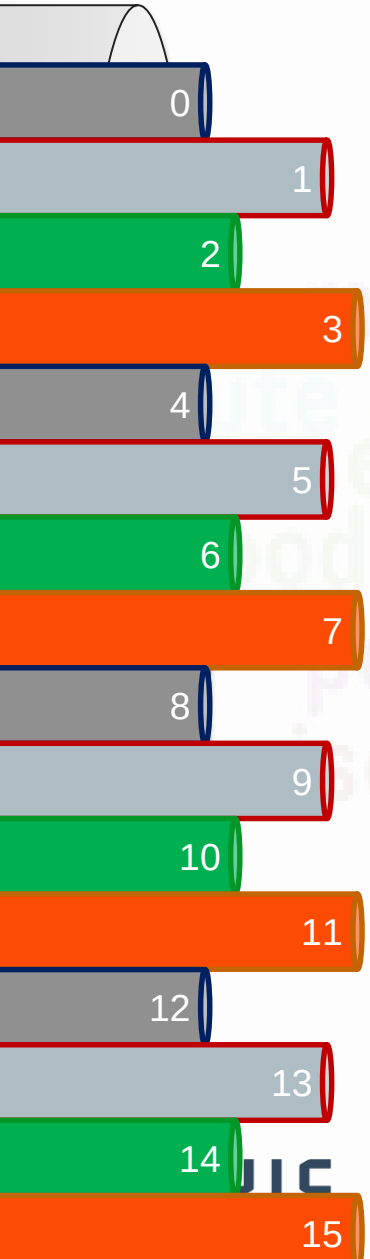
Active = open or recently closed
Inactive = closed >1 RTT ago





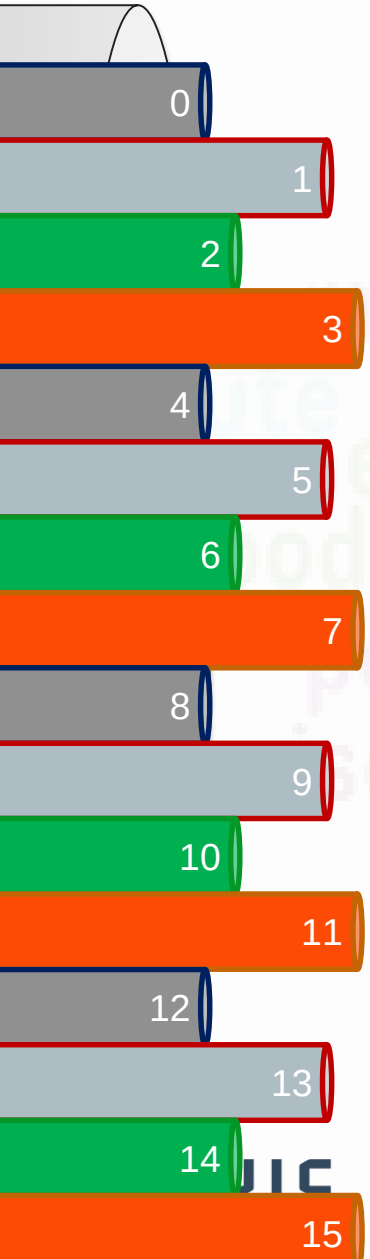
Self-Describing Unidirectional Streams

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, stop reading the stream (may trigger 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

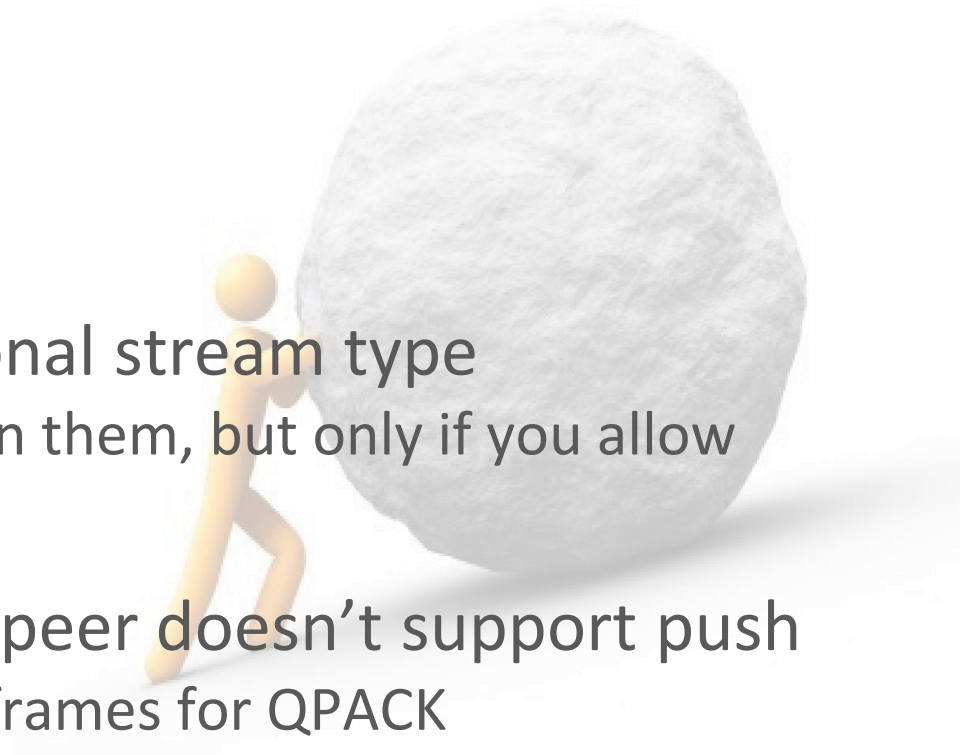
Self-Describing Unidirectional Streams



Server Bidi	Server Uni
Client Bidi	Client Uni

- In Kista, hum was roughly split between “do this” and “not sure yet”
- Follow-up discussion on list was largely positive, but acknowledged drawbacks:
  - Debugging without tools is somewhat harder
    - …in an encrypted protocol you can’t debug without tools anyway
  - If data arrives out of order, stream can be open with an unknown type
    - …which also makes the out-of-order data unusable, even if you support that

Philosophical Question: How Separate Is Push?



- Push streams are now just another unidirectional stream type
  - You still have to account for the QPACK frames on them, but only if you allow them to be created in the first place
- MAX_PUSH_ID frames aren't needed if either peer doesn't support push
  - If MAX_PUSH_ID remains 0, no PUSH_PROMISE frames for QPACK
- PRIORITY explicitly supports Push IDs as a prioritized/dependent object type
- SETTINGS_ENABLE_PUSH was removed in favor of MAX_PUSH_ID frames
  - Should we bring it back as a Server Push "master switch"?

All other HTTP/QUIC issues are editorial, parked, or post-v1!

...unless you opened more after I did these slides, of course.

0-RTT and SETTINGS

- QUIC:
  If 0-RTT data is accepted by the server, the server **MUST NOT reduce any limits or alter any values** that might be violated by the client with its 0-RTT data.
- HTTP/QUIC:
  Servers MAY continue processing data from clients which **exceed its current configuration** during the initial flight. In this case, the client MUST apply the new settings immediately upon receipt.

Previous connection

SETTING_HEADER_TABLE_SIZE: 64000

0-RTT

QPACK
Table size:56KB
Insert: (cookie,32KB blob)
Insert: *other stuff*

HEADERS
From table, using:
- cookie
- :authority
- user-agent

1-RTT

SETTING_HEADER_TABLE_SIZE: 4096

Proposal: Match Transport

Status quo: Tolerate client overruns

- Client has to deal with reduction of setting values after beginning to send data
  - …and there's no synchronization provided by the protocol
- Server has to recover old settings in order to differentiate between stale and malicious clients

0-RTT implies same or better

- Server has to involve HTTP in the decision of whether to accept 0-RTT
  - …which means recovering the old settings
- Each setting needs to define what constitutes "reduce or alter" if it's not obvious
- Settings can only increase, not decrease

Now implement and find the rest!