

HTTP Datagrams, UDP Proxying, and Extensible Prioritization

[draft-pardue-masque-dgram-priority](#)

IETF 115 – London – 2022-11

Lucas Pardue – lucaspardue.24.7@gmail.com

Background

RFC 9000 - stream multiplexing can have a significant effect on application performance

QUIC does not provide a mechanism for exchanging prioritization information.

RFC 9114 - HTTP/3 punts on stream prioritization.

RFC 9218 - Extensible Prioritization Scheme for HTTP(/2 and HTTP/3)

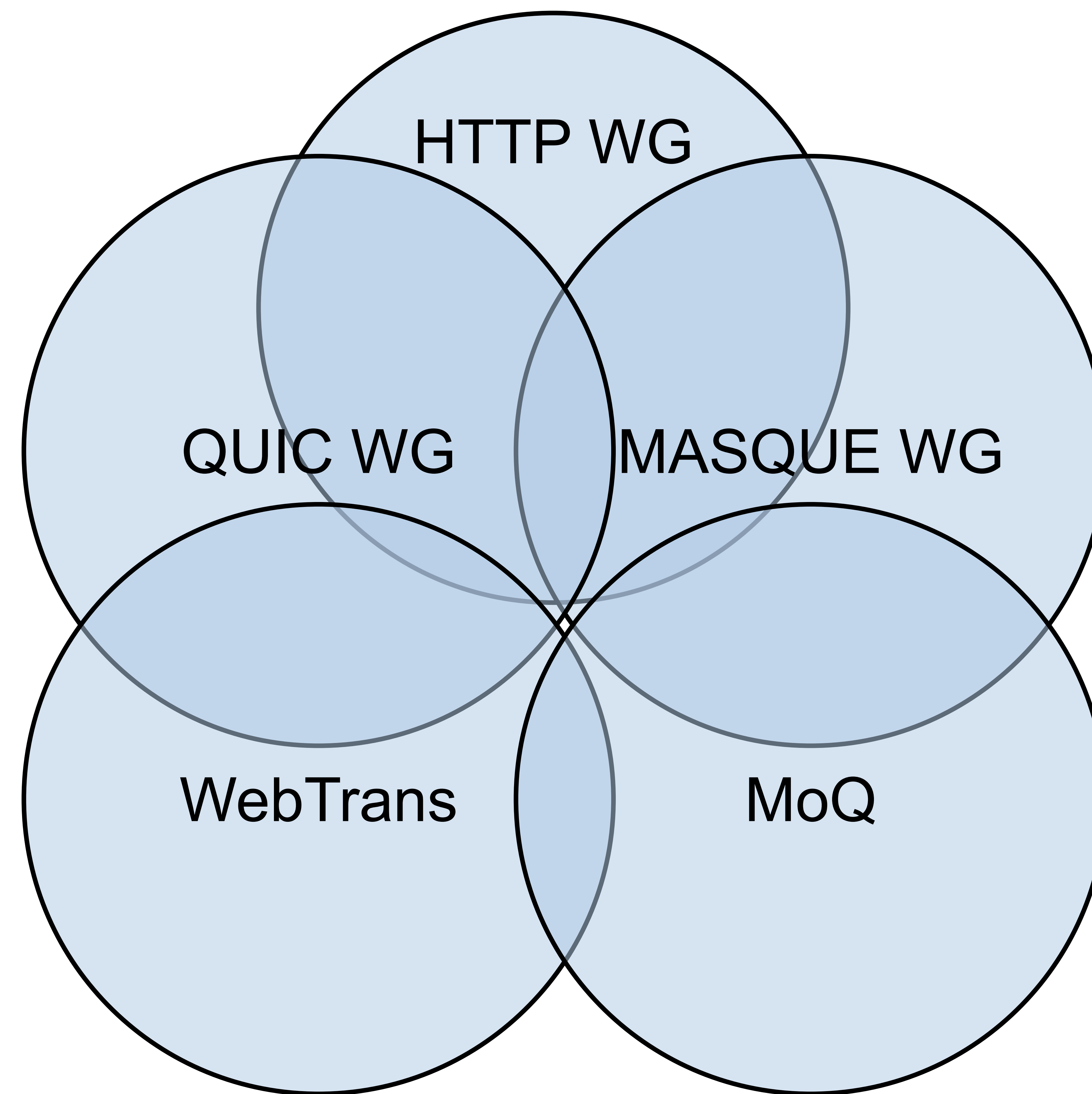
RFC 9221 - QUIC DATAGRAM frames. No transport multiplexing identifier.

RFC 9297 - HTTP DATAGRAMS and the Capsule Protocol

RFC 9298 - Proxying UDP in HTTP

MASQUE proxying and WebTransport definitely can exercise stream and datagram multiplexing

Venn and the art of protocol maintenance



Extensible HTTP Priorities recap

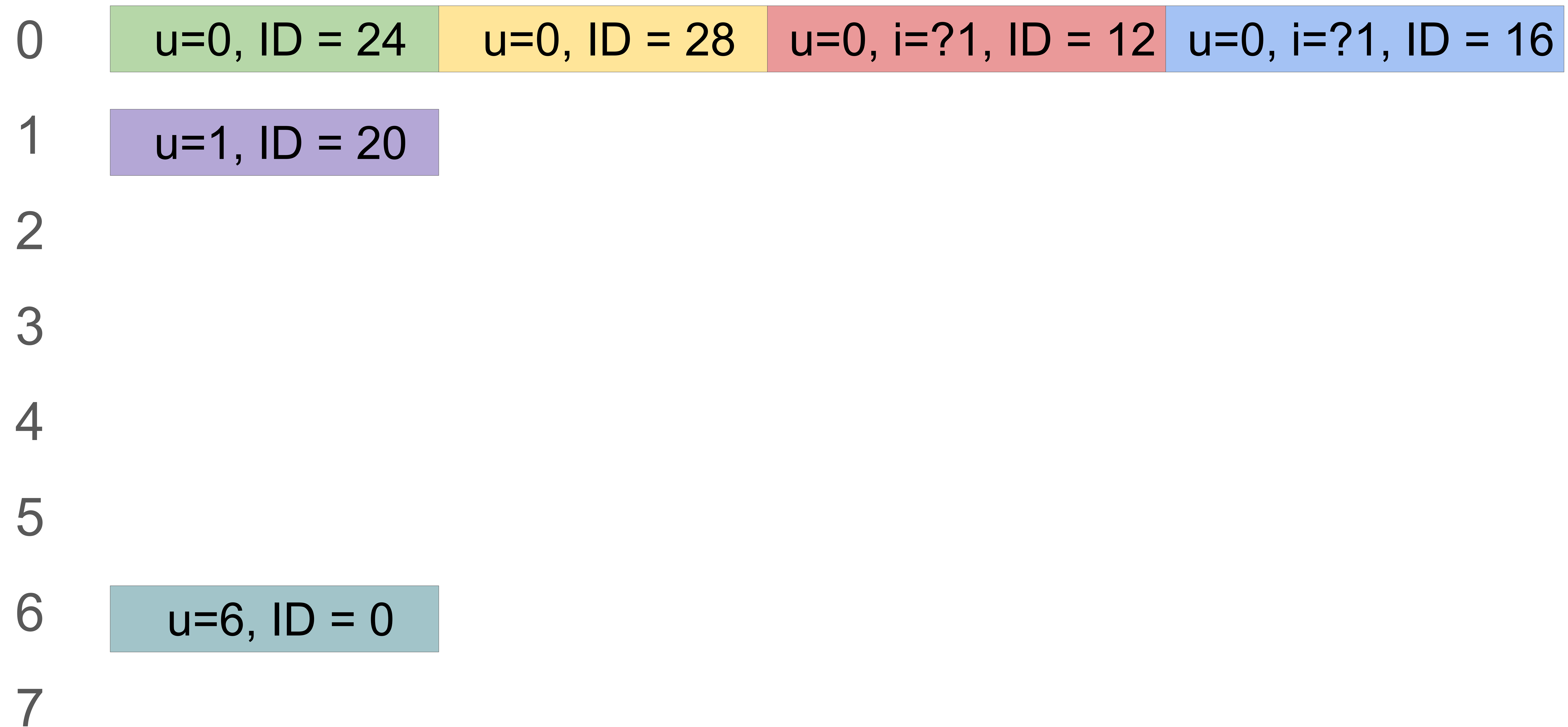
HTTP Extensible Prioritization for streams defines signals:

- *urgency* (“u”) - between 0 and 7. Smaller the value, higher the precedence
- *incremental* (“i”) - response can be processed incrementally (data as it arrives)

And some scheduling guidance:

- Expressing priority is only a suggestion.
- RECOMMENDED to respect urgency, serve in stream ID order.
- RECOMMENDED to respect incremental, fair bandwidth sharing between incremental at same urgency

Extensible priorities stream scheduling



HTTP datagram scheduling

FIFO??

Qrtr stream ID = 0

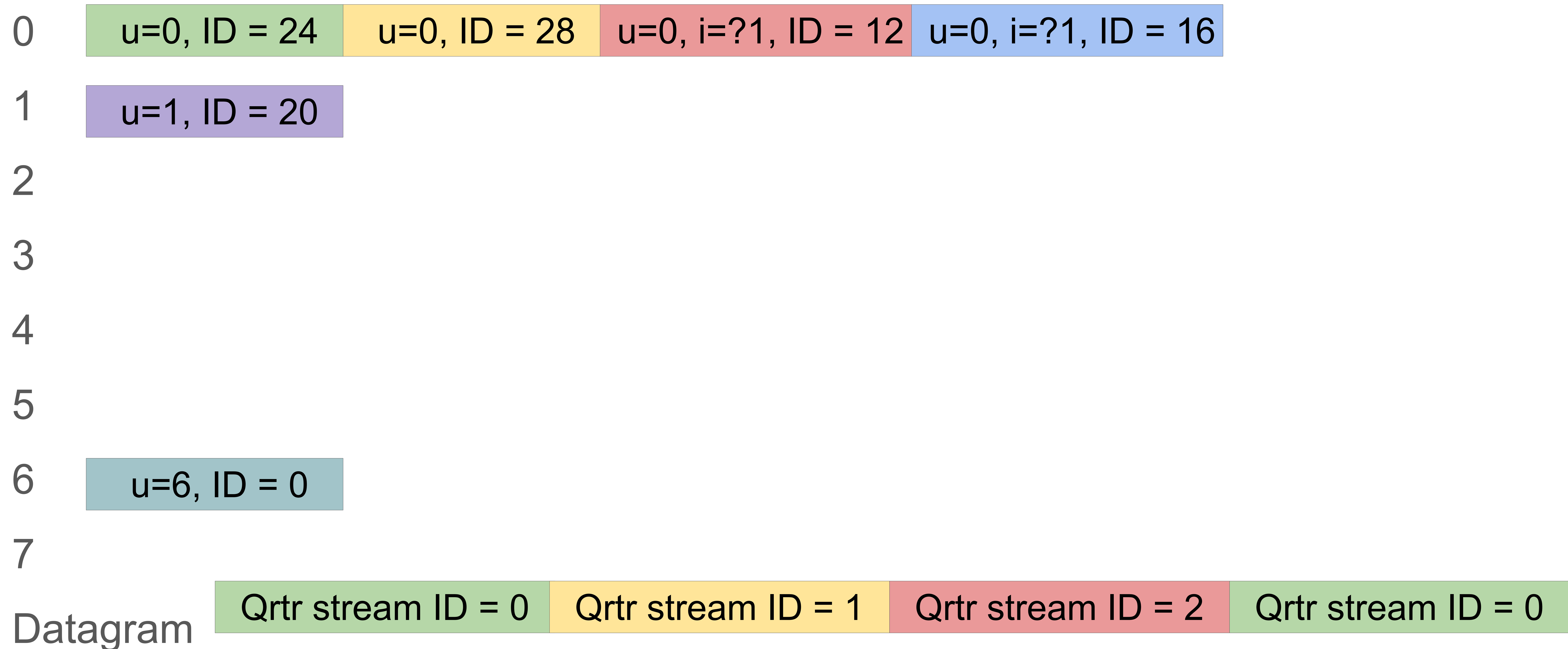
Qrtr stream ID = 1

Qrtr stream ID = 2

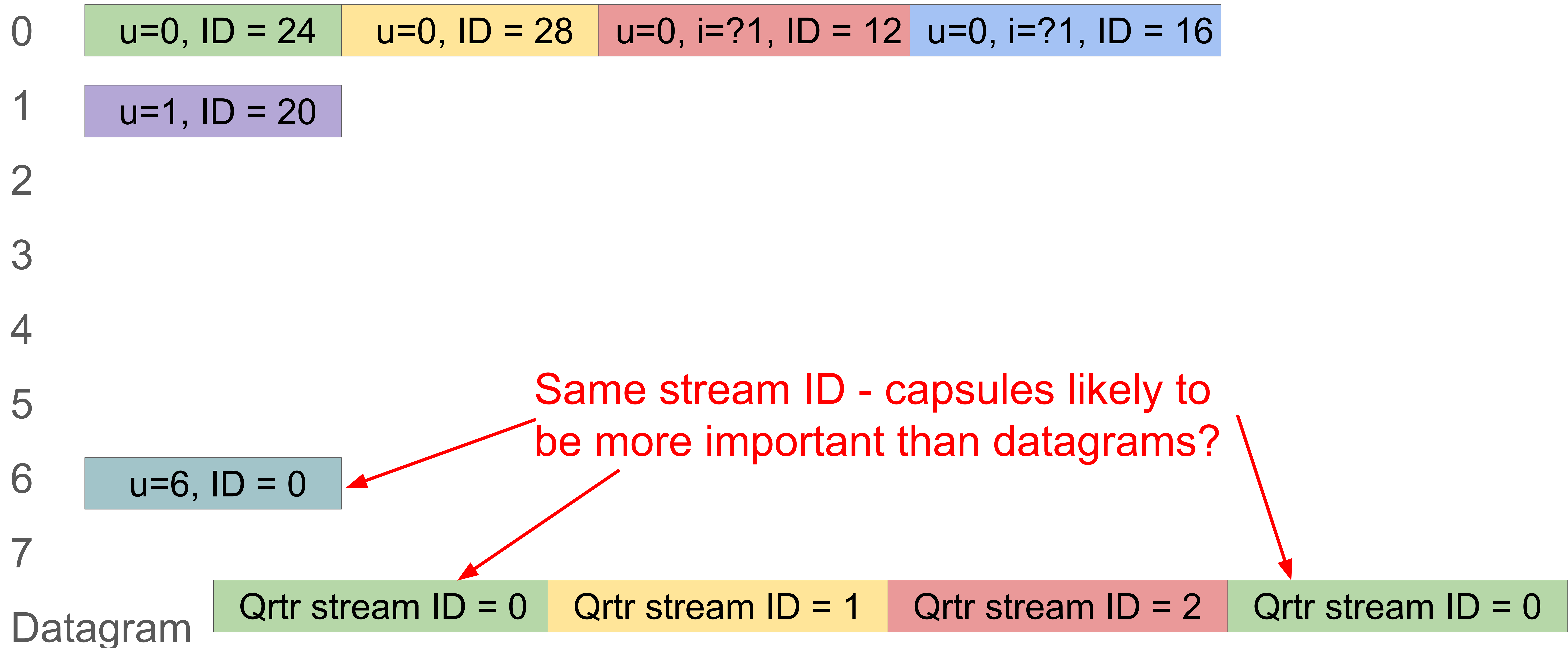
Qrtr stream ID = 0

Sticking stuff in the same bucket/queue is a bit basic

Stream and datagram scheduling?



Capsule and datagram scheduling?



Bouncing around

- HTTP Datagram Issue [#46](#) – The spec should discuss how h3-datagram works (or does not) with priority.
 - **Closed** with a PR that says:

Prioritization of HTTP/3 datagrams is not defined in this document. Future extensions MAY define how to prioritize datagrams, and MAY define signaling to allow endpoints to communicate their prioritization preferences.

- HTTP Priorities [#1550](#) – How are DATAGRAM frames prioritized?
 - **Closed** with a PR that says:

The priority scheme defined by this document considers only the prioritization of HTTP messages and tunnels ... Where HTTP extensions change stream behavior or define new data carriage mechanisms, they MAY also define how this priority scheme can be applied.

draft-pardue-masque-dgram-priority

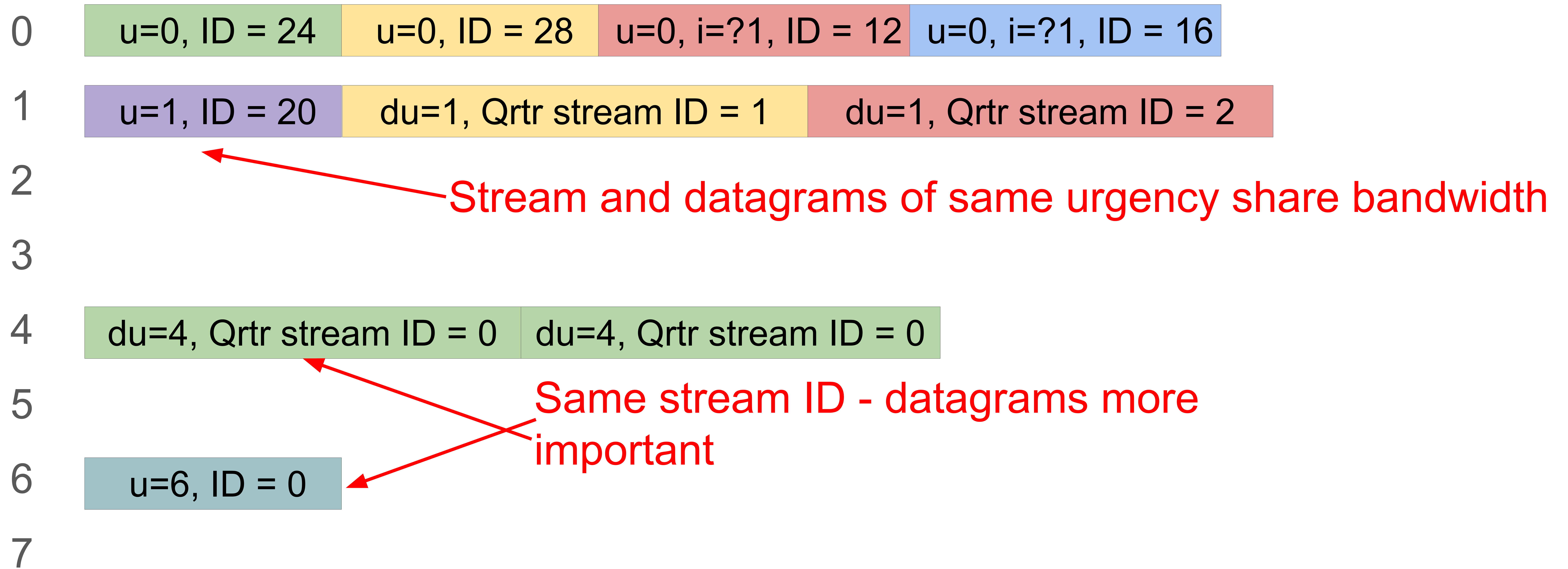
Extend extensible priorities with a compatible parameter: *datagram-urgency* (“du”).

Identical to *urgency*, except that it applies to datagrams.

Omission of *datagram-urgency* is a signal to use the default. But there is no default value. Instead the default is to use the stream’s *urgency*.

Where stream and datagrams have the same urgency, default recommendation is to **share bandwidth** between them when packetizing. E.g., 50/50 split between stream data and datagram data, or some other proportion

Capsule and datagram scheduling?



Datagram

Is the problem academic?

w3c / webtransport Public Unwatch 84

<> Code Issues 42 Pull requests 3 Actions Projects Wiki Security

Datagram vs stream & relative stream prioritization #62

Open vasilvv opened this issue on Sep 26, 2019 · 60 comments



vasilvv commented on Sep 26, 2019

Currently, we specify no particular order in which stream data and datagrams

The way the current scheme works in Google's implementation is roughly:

1. A timer or a received packet causes the connection to become writable.
2. The connection notifies the application so that it can send its datagrams.
3. The connection checks the send buffers for the open streams and sends

This works only if all operations are synchronous, and we can't make step 2

Weighted flows of datagrams and short-lived streams #419

Open jan-ivar opened this issue on Sep 22 · 18 comments



jan-ivar commented on Sep 22

TPAC TL;DR: fixing stream/datagram priority requires addressing a category problem: "datagrams take priority" means all datagrams over all streams, whereas "this stream takes priority" means over some other streams and sometimes datagrams.

#62 tries to solve two different things with the latter: 1) bandwidth allocation/starvation between long-lived streams (and datagrams), and 2) strict send-order within a media-flow of short-lived segment/frame streams. How can this work in concert?

Future apps may have *multiple* data flows (composed of datagrams or short-lived streams), and need a way to specify relative importance and speed (weight). Sending less works poorly, and setting the priority of every datagram

Proposal F from TPAC proposed new high-level objects to address this. With feedback from @wilaw, we'

```
const streamFlow = await wt.createStreamFlow({weight: 2.0});
for await (const segment of mediaSegments) {
  const sIs = await streamFlow.createShortLivedStream();
```

Support minimal prioritization necessary to enable IETF MoQ base streaming protocol #431

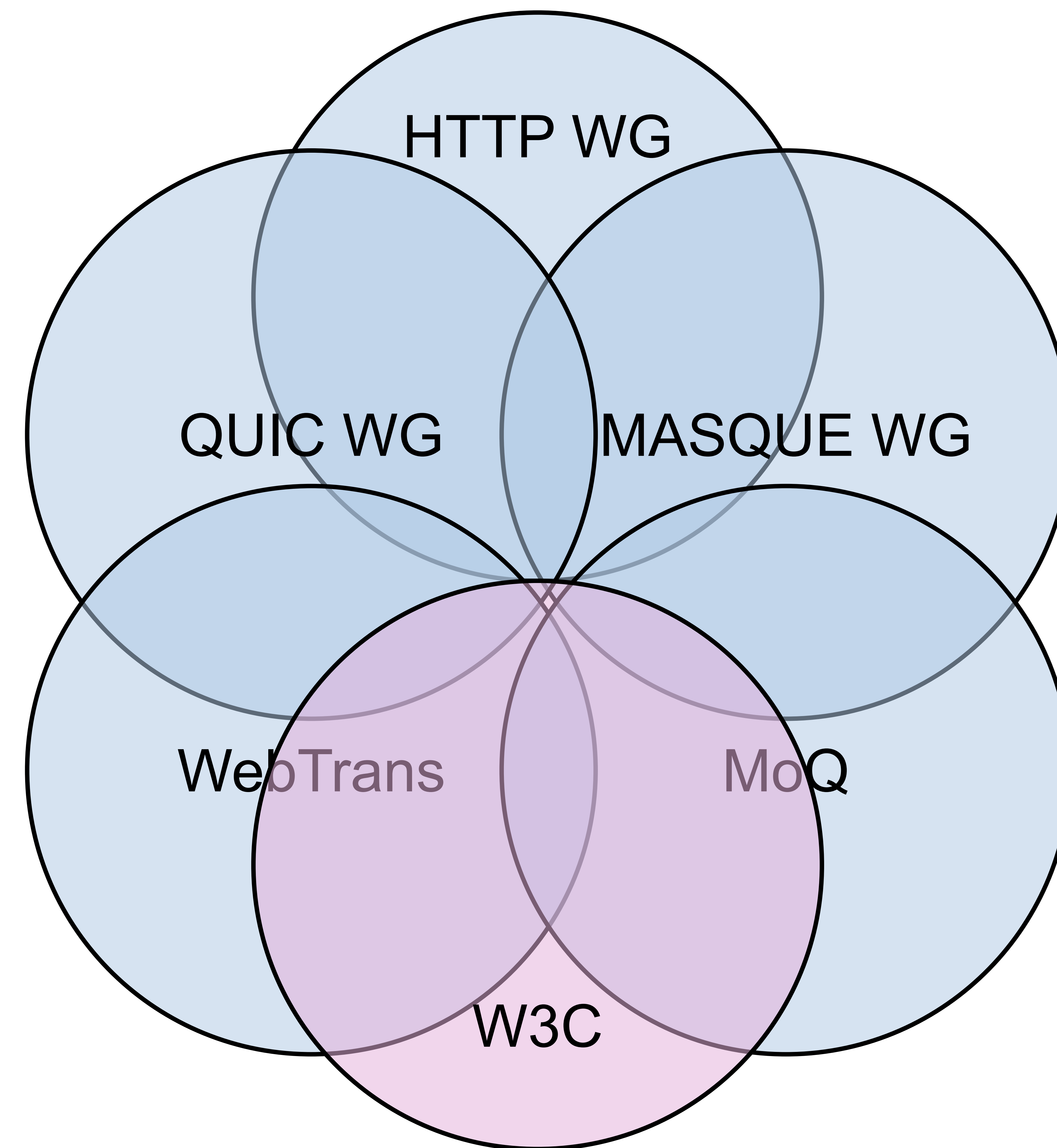
Open wilaw opened this issue 5 days ago · 0 comments



wilaw commented 5 days ago

There is an ongoing effort within the recently chartered Media Over Quic (MoQ) work group to standardize a simple low-latency media delivery solution for ingest and distribution of media. The use-cases include live streaming, gaming, and media conferencing and these correlate highly with the target use-cases for WebTransport within the W3C. As such, it seems reasonable that the initial version of the W3C Webtransport API should at a minimum provide an API surface that enables MoQ base protocol to be implemented in a browser user-agent.

Liaisons venn-gereuses



HTTP Datagrams, UDP Proxying, WebTransport, MoQ, and Extensible Prioritization

draft-pardue-masque-dgram-priority

IETF 115 – London – 2022-11

