

MOQT: What Gets Sent Next

June 18, 2024 - IETF MoQ Interim

Cullen Jennings (fluffy)

What is more important to send first

1. Audio more important than video
2. Low bitrate video more important than high bitrate video
3. Video of Bob more important to subscriber A but less important to subscriber B
4. Layered codec 15 fps layer more important than 30 fps enhancement layer
5. Low res thumbnails less important than main video screen
6. Video older than 250 ms not useful
7. Video older than 30 mins not allowed
8. Video that needs to be played soon is more important than audio far in the future
9. Fetching older media is less important than subscribing to live edge
10. The networks should order what to send next with network priority: EF or AF42
(first obj in group) / AF43
11. The WiFi should order this media with priority AC_VO (voice) or AC_VI
(video interactive)
12. In a GOP, I frames more important than P frames
13. In a video, long term ref frames are more important
14. This media for metrics should be less than best effort
15. This media is non queue building and should have priority over bloated stuff (L4S)

Prefetch

Video player is:

- 1) Playing video A (most important)
- 2) Prefetching the first 5 seconds of video B, and C so that if the user switches to play one of them, the new video can instantly start playing
- 3) When getting the beginning of B or C is done, will get start of D.

Req: Subscriber can say A is more important than other videos it is getting, then later change one of the other videos to more important.

Prefetch Fairness

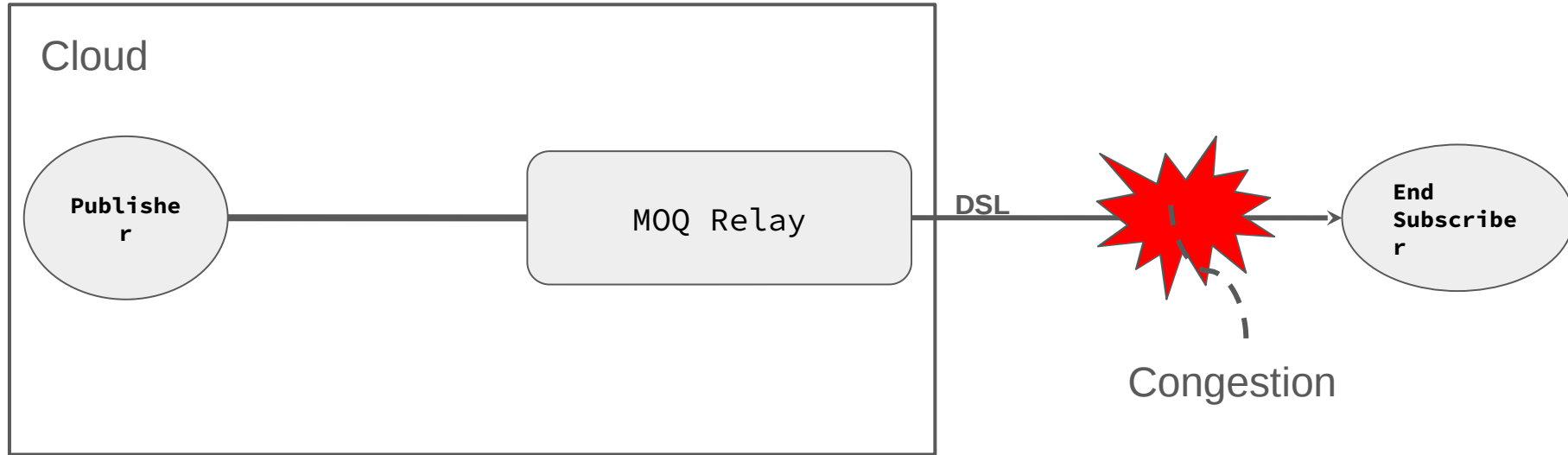
Video player is:

- 1) Playing video A which has 4k and 1080P
- 2) Prefetching the first 5 seconds of video B for fast channel switch

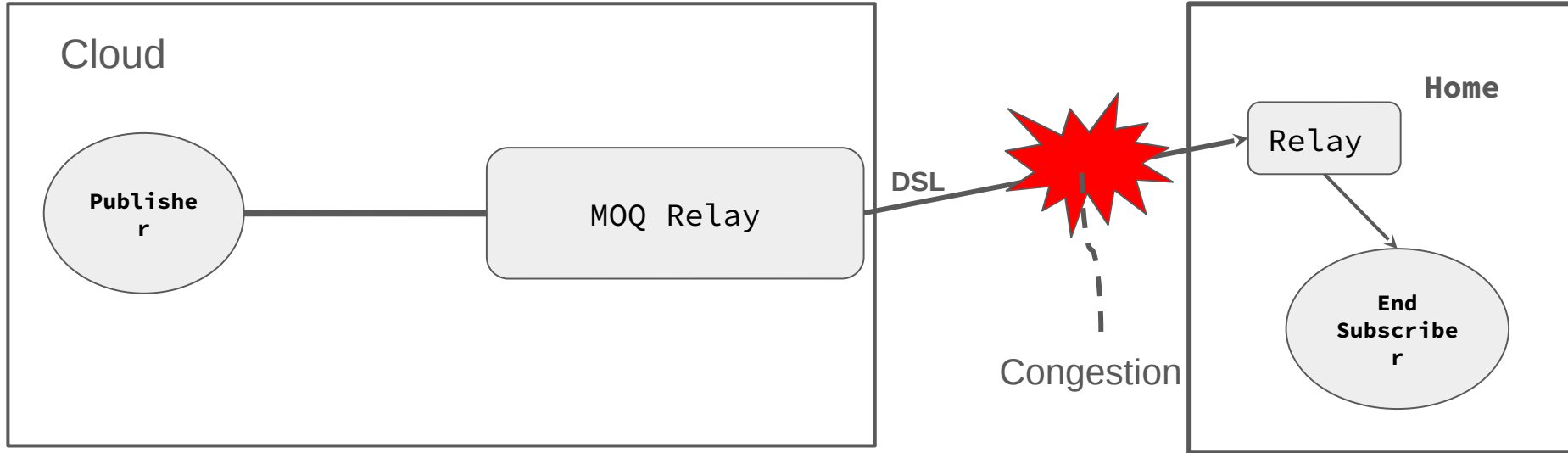
Better to have 1080p with fast channel switch than to have 4K with slow channel switching.

Req: be able to say that prefetch should get some percentage of the bandwidth

Where can things go wrong : “Last hop between End Subscriber & the Relay”

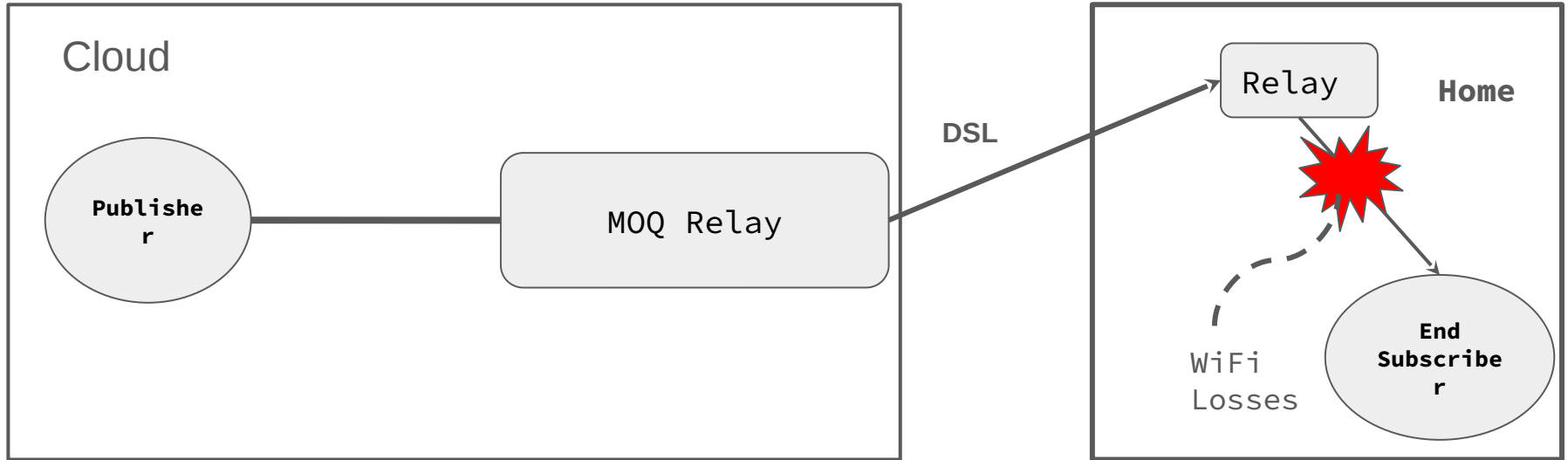


Where can things go wrong - “Between home relay and cloud relays”



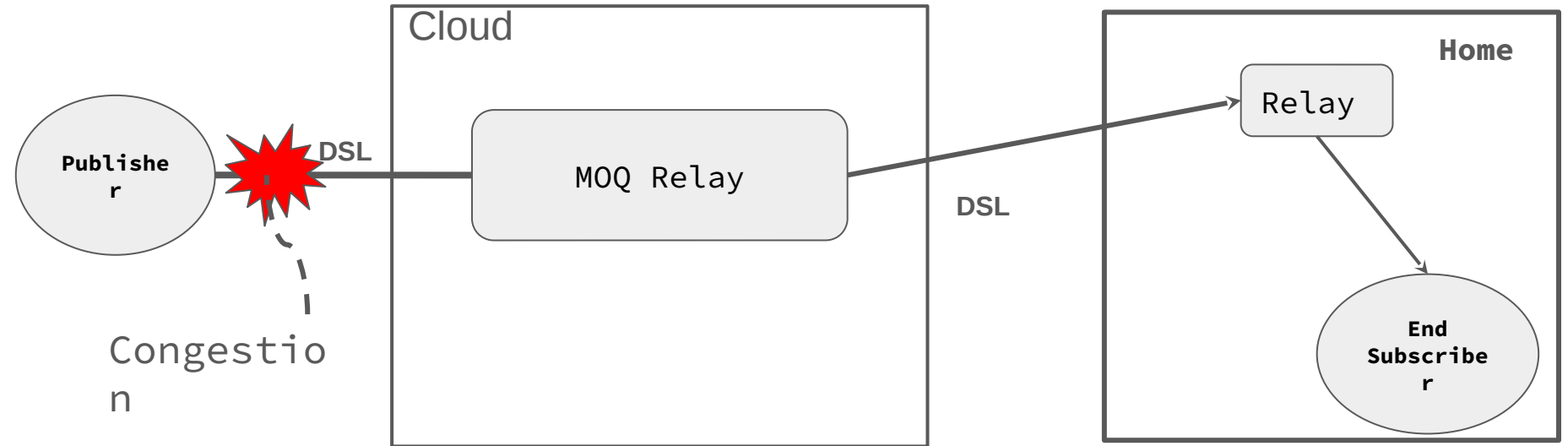
- In common case of WAN congestion, congested link is between two relays, not end client and relay

Where can things go wrong - “Between Local relay and the End Subscriber”



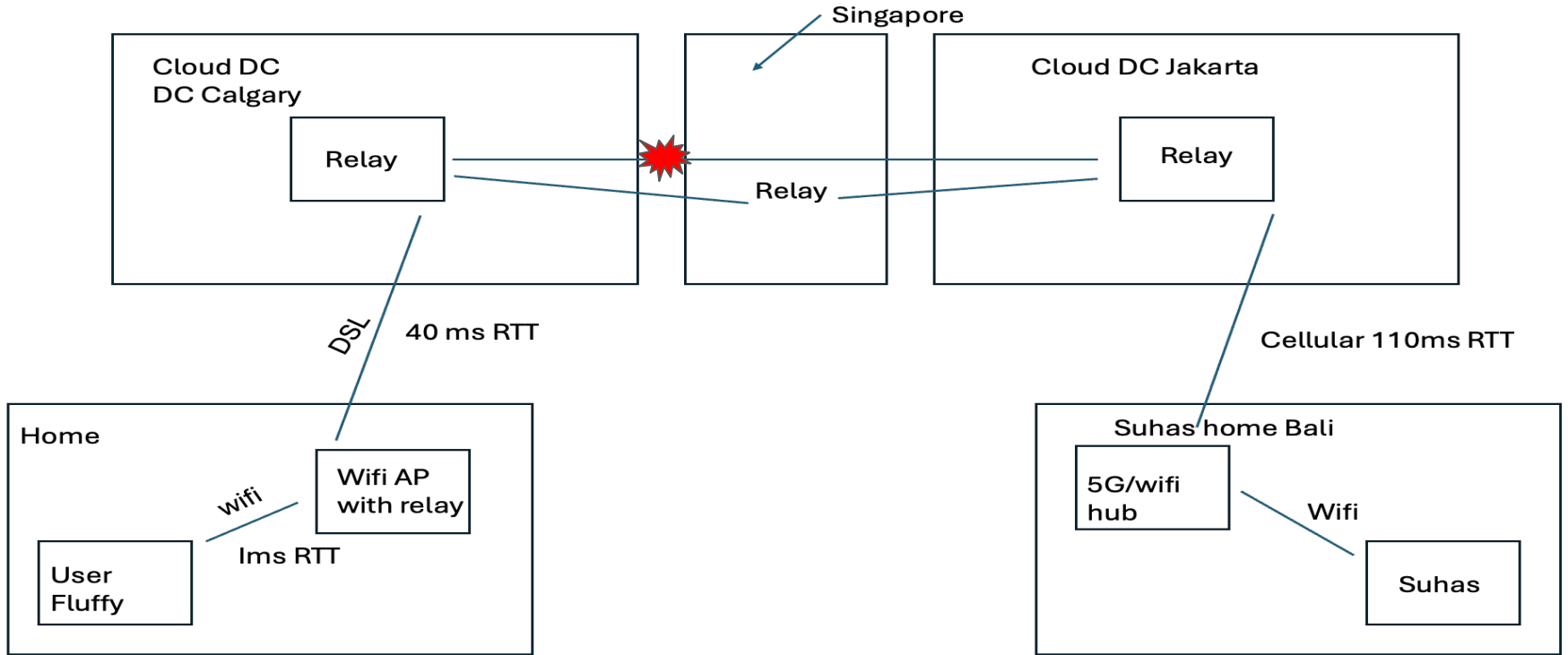
- Local relay in home allows fast retransmission/recover of packets lost on wifi
- Wifi usually supports priorities of AC_V0 & AC_V1

Where can things go wrong - “First hop from original publisher to the first relay”



- Upstream broadband often supports priorities like AF

MoQ Deployment



Moving to slides that help talk about solutions

Strawman Day 1: What gets sent next

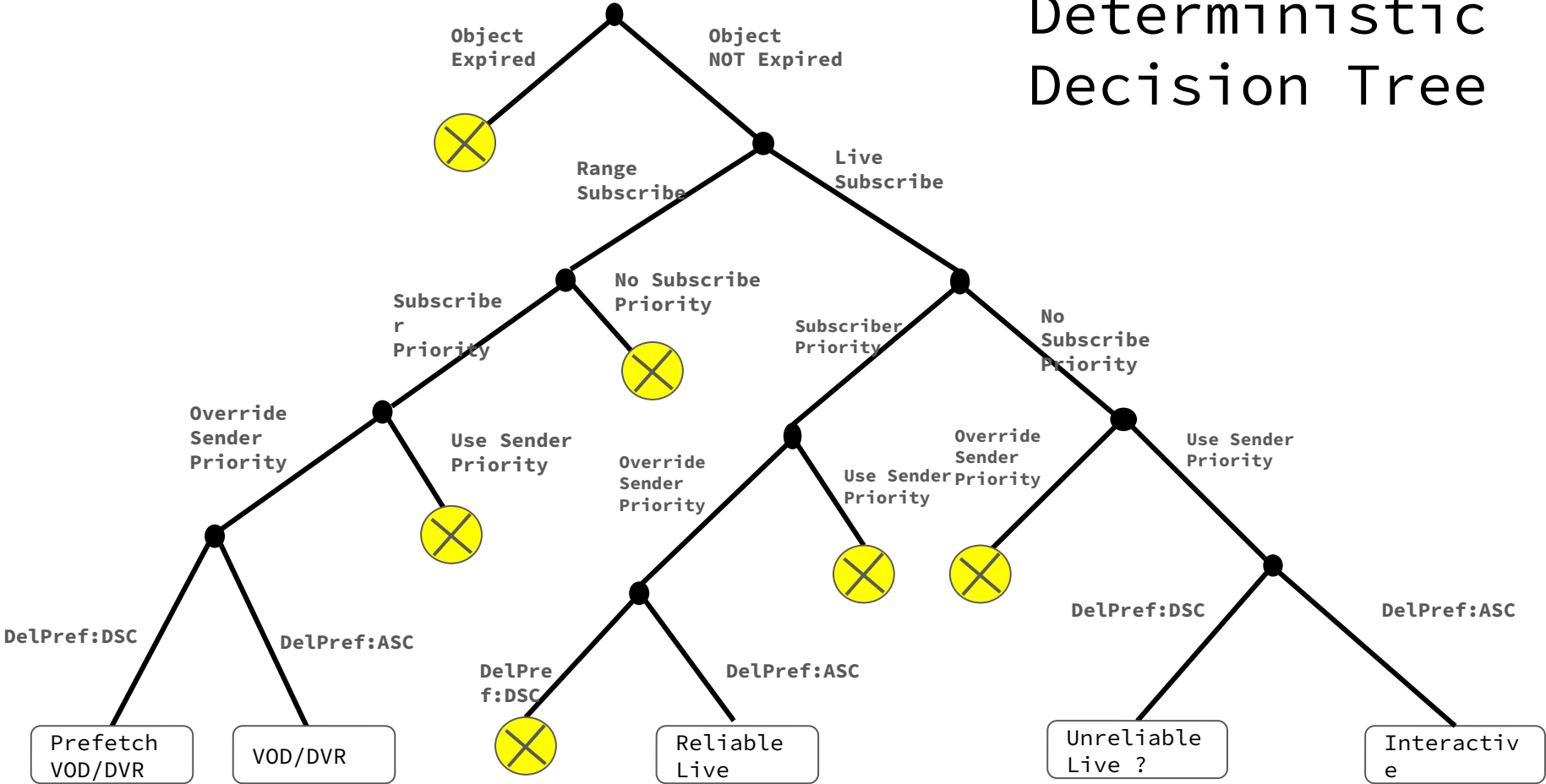
Chooses object to send next out of the objects available when there is space to send something in that connection:

- Pick unexpired objects
 - Pick track with lowest subscriber-order
 - Pick the track with lowest sender-priority
 - Pick oldest or newest object based on subscriber-delivery-pref of FIFO or LIFO

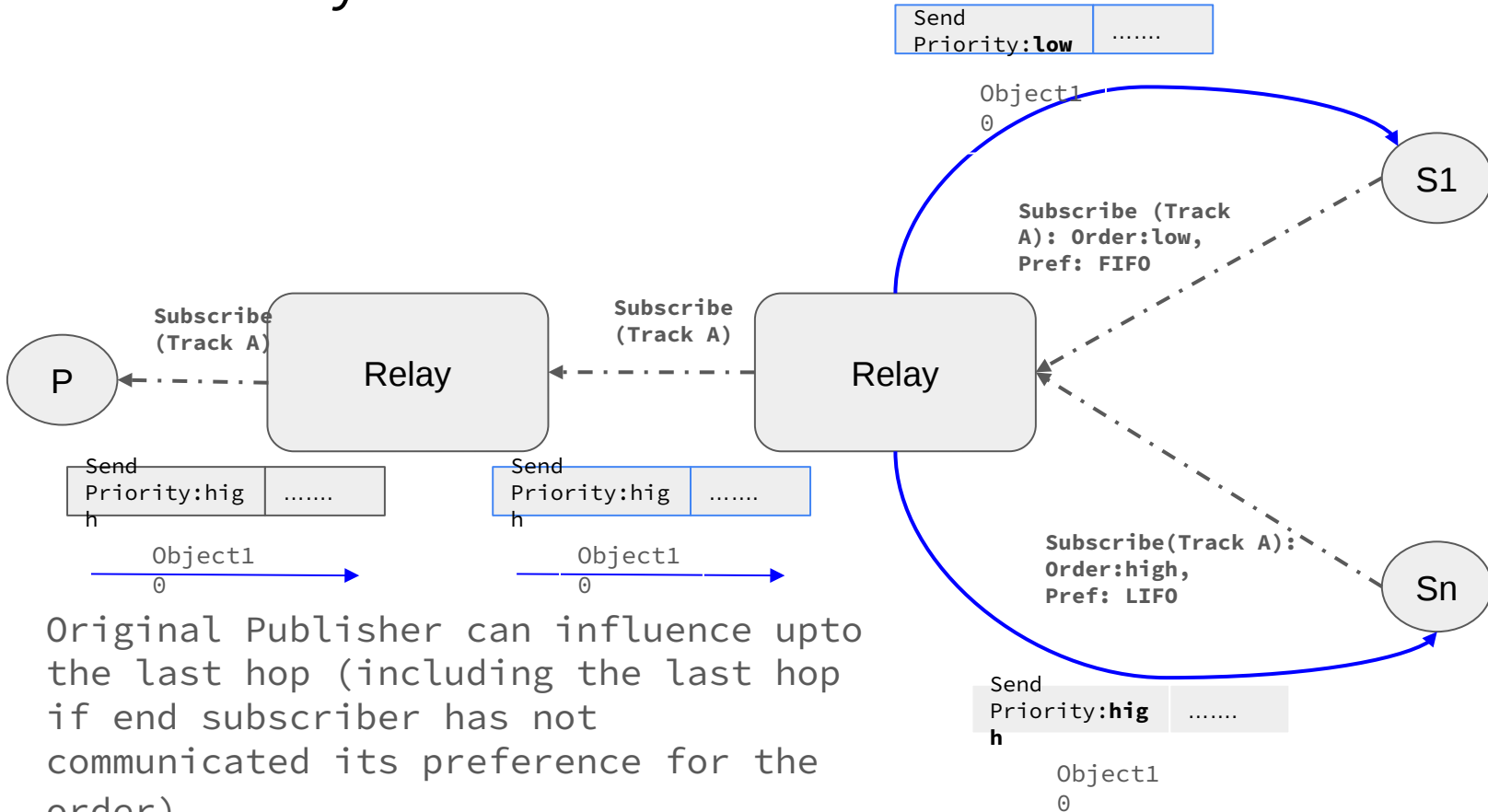
* Default subscriber-delivery-pref is FIFO

* subscriber-order and subscriber-delivery-pref only go from end subscriber to first relay/publisher and are not aggregated upstream

Deterministic Decision Tree



End Subscriber “can” influence “Last Hop Delivery”



Original Publisher can influence upto the last hop (including the last hop if end subscriber has not communicated its preference for the order)

Day 2, Take 2

Strawman Day 2: What gets sent next

When there is space to send something on outbound connection, chooses next object to send next out of the objects available in that connection:

- Ignore expired objects
- Ignore tracks with no objects to send
- Select a track to send next by :
 - Rank by **“subscriber-priority”**, if equal rank by **“original-publisher-priority”** and if equal then implementation picks out of this the equal ones.
- Within the selected track, select group by:
 - Prioritize highest or lowest **group-id** based on **“subscriber-sending-order”** of Ascending-Group-ID or Descending-Group-ID
- Within the selected group, select object by:
 - Prioritize the **smallest object-id** in group

* Default end-subscriber-delivery-pref is Ascending

For subscriber-priority and subscriber-delivery-pref, relays SHOULD indicate “no preference” when making an upstream subscription for the track. Explain why this makes sense, and acknowledge Relay may have it own view of priorities.

Expiration (this is not Max Cache Duration)

Expiration times are relative specified by original publisher and are relative times.

- This only makes sense for data delivered on live edge
- Question to resolve: individual subscribers might have different constraints, but expiration time applies to the content at the relay. Something here to ensure that expectations of the publisher and the subscriber are both met. Likely through a note that indicates to the publisher to do the right thing.

The publisher may set (or not)

The subscriber can override for track.

The relay takes min of publisher and subscriber time for this subscription.

Relay do not propagate subscriber expiry time up (proxy might)

Timestamp recorded when received start of object.

All objects in the group have the same expiry timeout.

Have to do FIN or RST. When ? Jana will give us his take on this ...

Note that the receiver can always stop a stream

Scope for the Priorities

Is original-publisher-priority per track or group or object ?

- Priorities of object cannot change inside a group
- If doing stream per track, end-publisher cannot change the priority mid track. If you want to do this, use stream per group or stream per object.
- If doing stream per group, end-publisher can change on each group
- If doing stream per object, end-publisher can change on each object
- end-subscriber can change subscriber priorities at any point by issuing new subscription or updating existing subscription.

How to encode on wire ?

Original-Publisher puts the “original-publisher-priority” in a header at the start of stream.

- Subscribe OK will not work for this because the priority can change inside a track when doing stream per group or stream per object.
- We also have issue the a relay can get an object from publisher before the subscribe OK arrives.

Subscriber and Subscribe Update indicates the “subscriber-priority” they want in the subscribe message.

Setting QUIC Stream priorities

- Set the stream priority based on the first object to be sent on the stream
- If there is a subscriber-priority use that otherwise use original-publisher-priority
- QUIC stream priority does not change after the first object.

Setting DSCP

- Catalog can provide a Pref-DSCP for each track.
- Subscriber puts Pref-DSCP in subscribe.
- Any QUIC connection sets the DSCP for all traffic in that connection based on the PrefDSCP of first object sent on that connection.
- Subscriber that uses tracks with two different Pref-DSCP, forms two connections to relay and does subscribes for all the EF over one and subscriber for all the BE over the other.
- Publish that uses the two different PrefDSCP, forms two connections to relay and does announce for all the EF over one and subscriber for all the BE over the other. If the the two track are in the same namespace, then relay needs to accept the Subscribe received on the connection with the correct DSCP for that track.
- Relays probably don't use DSCP on relay to relay traffic, but if they want to, they would see the subscribes and form new connection if it was a new DSCP.

Range of Priority Values

- Proposal: Range is 0 to 255.
- Same range for original-publisher-priority and subscriber-priority
- RFC 9218 is not enough priorities for typical video conference
- Proposal A (0 to 63): (this is RFC 9218 x10)
 - 0 is most important
 - 30 is default
 - 63 is least important
- Proposal B (0 to $2^{62}-1$)
 - 0 is the most important
 - ? is the default
 - $2^{62}-1$ is the least important
- Mapping to QUIC Priority ?
 - What is the minimum thing QUIC stacks guarantee to do today ?
 - Why does QUIC WG think priorities are usable without this ?
 - Want to make progress ⇒ start with Proposal A
- Should we have 1 byte so we can concatenate sender-priority / publisher priority

Do we need a original-publisher-group-order?

This would be way for original-publisher to set default ascending or descending order of the groups, if the end-subscriber did not specify one.

Names: lowest-group-id and highest-group-id

Proposal: Yes, add it in subscribe OK and in track status.

- Default is lowest-group-id (from earlier slide)
- no use case, can add later
- end-publisher can assign group-ID the way they want into groups so already have implicit control of this

Tracks from different Namespaces

Issue is a end-subscriber subscribing to tracks with different namespaces over the same connection with same subscriber-priority values and uncoordinated original-publisher-priority values between the namespaces.

- Lacking a use case for this.

If the end-subscriber uses a different subscriber-priority for all the tracks, this will override any use of the uncoordinated values.

If the end-subscriber uses different connection for the two namespaces, this will result in fair sharing between the namespaces.

- probably not what application wants but again lacking use case

Proposal: Do nothing and write these considerations up in the draft.

Equal Priority Fairness

When the end-subscriber-priorities of two tracks are equal, and the original-publisher-priorities are also equal, then what...

- Proposal A: tie breaker is FIFO
- Proposal B: “do round robin” based fair bytes on wire

This can be left up to relay as application can assume both tracks will get some bandwidth.

In most uses cases, it seems getting half of both tracks is worse than getting all of one.

- Proposal C: relay consistently chooses one
- Proposal D: signal consistent or round robin