Request Ordering

Constraints

MOQT messages for different tracks can be received and processed out of order

Unsubscribe is no longer a MOQT message, it’s a QUIC control message

- Cannot be ordered with respect to MOQT operations anymore

Use Cases

1) Swap Tracks
   E.g. in a video conference, replace Alice with Bob
   Want to pause Alice's track before resuming Bob to reduce congestion

2) Client Side ABR
   Replace Rendition A with Rendition B without overlap or gaps
   Sounds familiar…

3) Repeated updates to the same Track
   Need to be ordered
   → Already covered in draft-18: REQUEST_UPDATE serialized on Track's bidi

Are there more?

Seems to come down to REQUEST_UPDATE

Forward → Yes

Priority, Delivery Timeouts → changes likely don’t need to be synchronized

Range Filters → is atomic ordering updates between tracks necessary?

Solution Space

1) **Do nothing – defer to QUIC packetization**
   Control messages are small and high priority → will go in one packet
   Live the the races

2) **Reconstruct the ordering of the control stream**
   2a) Required Request ID
      People didn't like. Additional changes required to bound receiver state
   2b) WAIT_FOR
      More generic mechanism, state management still an issue

   Neither guarantees ordered/atomic application processing

Solution Space

3) **An atomic mechanism for swapping tracks**

4) **Something else?**

The SWITCH_FROM Parameter

```
SWITCH_FROM {
  Switch From Request ID (vi64),
  Mode (1),      // Hard or Soft
  Publish Done (1),
  Reserved (6),
}
```

Can appear in SUBSCRIBE, REQUEST_UPDATE

Sent on the “resume” subscription bidi stream

→ No race, the Switch From (suspend) subscription must exist

→ No need to pre-subscribe either – include SWITCH_FROM in SUBSCRIBE

Location Filter of resume subscription governs behavior

SWITCH_FROM

The publisher determines the Start Group of the resume subscription as normal

1. Set Forward=1 and Apply the Subscription *Filter*
2. Continue delivery on suspend until there's an object to publish in the Start Group
3. Make the Switch

   Mode = Hard

   Set Forward=0 on the suspend subscription

   Mode = Soft:

   Set End Group = Start Group - 1 on the suspend subscription

   Cancel delivery of any Groups >= Start Group

   Send PUBLISH_DONE if requested
4. Send SUBSCRIBE/REQUEST_OK

SWITCH_FROM

Publish Done = 1

- Send PUBLISH_DONE on suspend subscription after executing
- For “soft” mode, requires End Of Group signal, or timer

Publish Done = 0

- Suspend subscription remains open

Switch Alice for Bob

Use SWITCH_FROM Hard

These tracks are not group aligned → soft mode will do the wrong things

Also, groups may be very long

ABR Cases to Consider

Ready to upswitch:

- Conservative: minimize overlap
  - → Use Soft with some unreceived absolute group
- Aggressive: accept overlap for quality upgrade of received groups
  - → Use Hard with some already received/partially received group

Needs downswitch

- Conservative: high-res continues while relay readies low-res
  - → Use soft with careful group choice
- Aggressive: stop high-res NOW
  - Don’t use SWITCH_FROM → No coordination required

Does it cover all the cases?

Folks need to work it out.

The mechanisms – building on 1642, using a param – feel right

Room to add functionality – unused bits in SWITCH_FROM flags

Inclined to keep it as simple as possible and let data drive additions