



Joining Fetch Dissent Potential Solutions

Boulder, Feb, 2026
Ian Swett

Focus on HoL Blocking

WG Indicated this was the most concerning problem

Multiple changes to current draft are needed to achieve this with the current approach (Alan's example/detail)

At least 3 implementations 'Lie' and start at the current Group today ([#1386](#))

So that must be a promising direction

Detailed Alternative: Join Group 'Location Filter'

Specifies a number of Groups to go back (like [#1362](#))

- 0 is equivalent to 'Join Immediately' aka NGR

Publishers can provide an alternate starting Group
'*StartGroup*' Param = First Group to be delivered
Otherwise start at Largest Group like today

Using SUBSCRIBE because it has the optimal delivery encoding for beginning playback quickly

Considering NGR use case as a first-class concept

Client

Join Immediately (NGR)

SUBSCRIBE (Join.Group=0)

Join at Current Group

SUBSCRIBE (Join.Group=1,

Join at N Groups Prior to Current Group

SUBSCRIBE (Join.Group=N+2)

Rejoin on Existing Subscription (Forward=1)

REQUEST_UPDATE (Join.Group=0 or 1, Forward=1)

Relay - Caching or Not

Has all Groups in Cache (and is willing to deliver)

Immediately respond with 'StartGroup'

Has some recent Groups in Cache

Immediately respond with 'StartGroup'

Has No Existing Subscription

Send SUBSCRIBE upstream

Has no cached Objects, but Active Subscription

Either go upstream or return no 'StartGroup'

New Group Request

To know when to deliver from cache vs upstream, add a MAX_JOIN_OBJECT_ID Track Extension Header

Relay:

If Largest Object Id for Latest Group is larger, go up
If not, start Subscription at the Current Group

Original Publisher:

Decides whether to start a new Group if it receives a SUBSCRIBE (Join.Group=0)

Does this allow us to remove Joining Fetch?

Joining Fetch is a latency optimization

What's the worst case here?

Publisher returning without StartGroup and mid-group

Costs < 1 RTT, because the Publisher can still immediately start delivery Objects

Flow Control Concerns?

Publishers can respond with `TOO_FAR_BACK`

Today, publishers can limit the number of streams they use for a single Subscription to avoid it starving others

“What if there are 1000 Groups”

Not new: No limit on number of open Subgroups or Groups today

We’re not optimizing for pathological cases, we’re just need to ensure they don’t cause other problems