

Fetch

Unified Proposal

Overview

New Control Message for managing fetch transactions
FETCH, FETCH_OK, FETCH_ERROR, FETCH_CANCEL

New data stream for fetch data with fetch header message,
bunch of objects, then fetch complete messages.

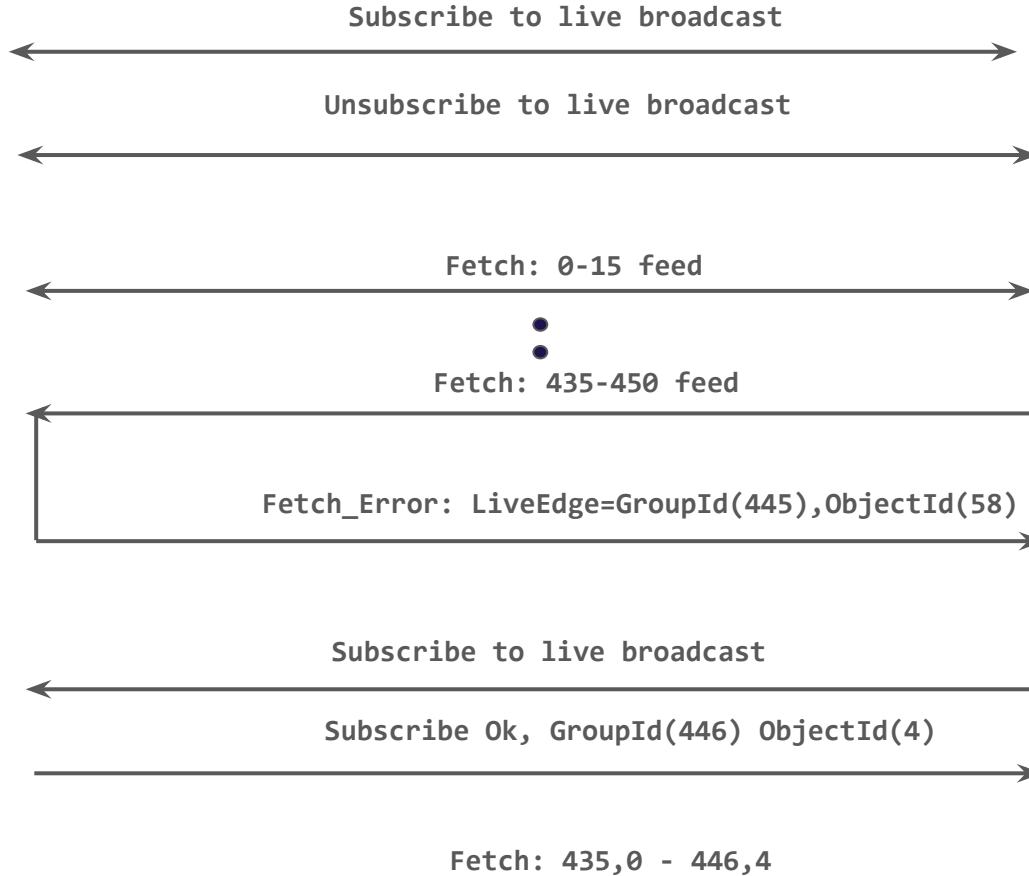
Examples

Transition
to Live

User scrubs 10 mins back in the playhead and plays at 2x

Source

Client



1. On Subscription, User discovers that current timestamp is 10 mins into the feed. The user decides to scrub the player back to the beginning of the feed. Player then unsubscribes to the live feed.

2. Player starts fetching 30s worth of feed every 15s, plays it at 2X speed, until it receives Fetch_Error.

3. Player subscribes to live feed and issues fetch to fill in remaining gaps from the live edge.

4. Player is now caught up to

User filling 5 seconds buffer before Live Edge (2)

Source

Client

Subscribe to live broadcast

Subscribe Ok, LiveEdge=GroupId(3011),ObjectId(5)

Fetch: 3009,0 - 3011,5

1. Player subscribes to live feed and also issues fetch to fill in 5 seconds buffer up to the live edge. It also buffers the live edge feed off the subscription, while the fetch is in progress.

2. Player is now receiving live edge and has 5 second buffer.

Proposal : Fetch Message

```
FETCH {  
  Fetch ID (i),  
  Track Namespace (b),  
  Track Name (b),  
  Fetch Priority (8),  
  Group Order(i),  
  StartGroup (i),  
  StartObject (i),  
  EndGroup (i),  
  EndObject (i),  
}
```

```
FETCH_OK {  
  Fetch ID (i),  
}
```

```
FETCH_ERROR {  
  Fetch ID (i),  
  ErrorCode(i),  
  [Latest Group ID(i)],  
  [Latest Object ID(i)]  
}
```

```
FETCH_CANCEL {  
  Fetch ID (i),  
}
```

1. QUIC Data Stream per request and response
 - a. Request sent on control stream
 - b. Data is sent over Unidirectional Stream
2. Absolute Range identifies the start and stop points for the requested objects
3. Fetching for data ahead of live edge is treated as error and returns live edge in Fetch Error message on the control stream.
4. Group Order specifies client's preference for ordering the groups. Default is ASC.
5. Fetch Priority is compared against subscriber priority within the session up to the first relay.

Proposal : Fetch Header and Object Message

```
FETCH_HEADER MESSAGE {  
  Fetch ID (i),  
  Fetch Priority (8),  
}
```

```
{  
  Group Id(i),  
  Object Id(i),  
  Object Payload Length(i),  
  [Object Status(i)],  
  Object Payload(..)  
}
```

```
FETCH_COMPLETE MESSAGE {  
}
```

Examples

Cache &
Relay

1A

ASC Order Fetch

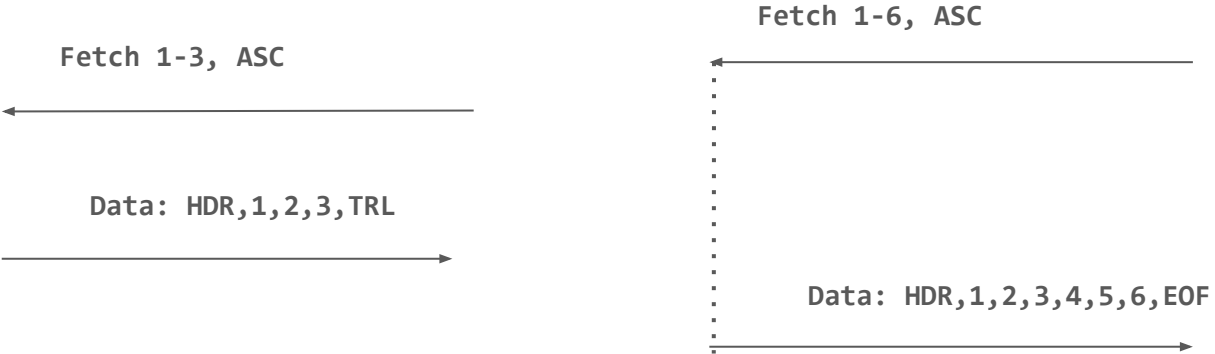
Original Publisher



Relay



Client



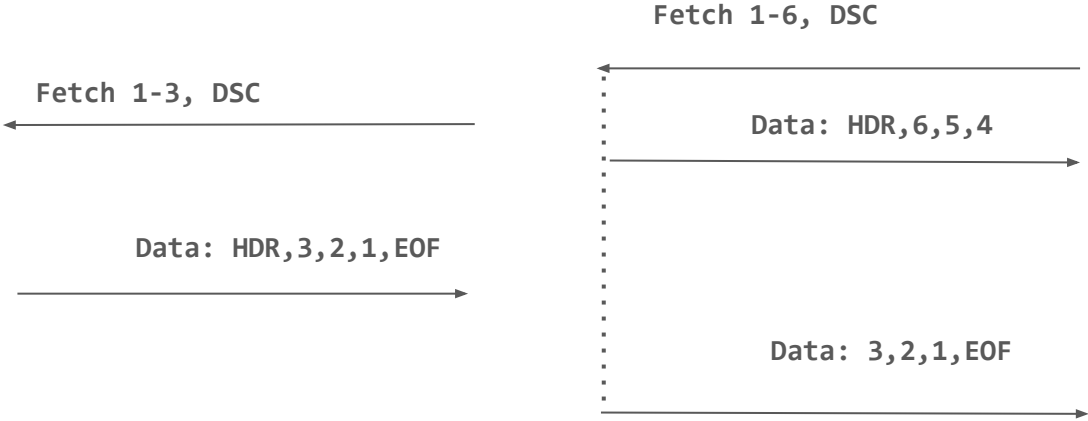
1B

DSC Order Fetch

Original Publisher

Relay

Client



2

Fetch of non existing data

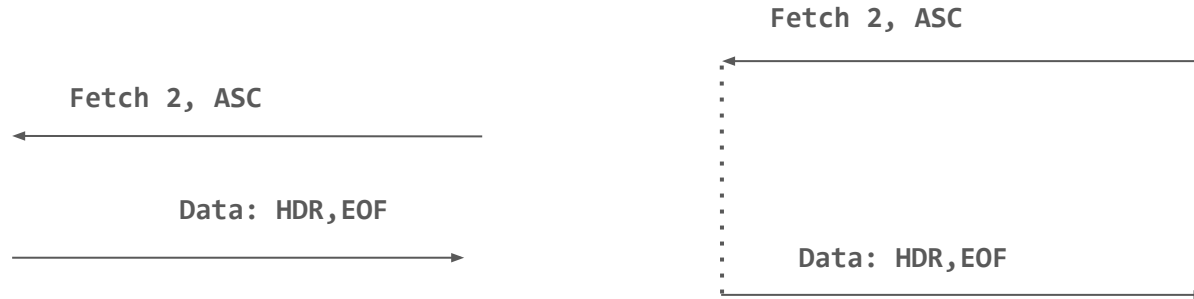
Original Publisher



Relay



Client



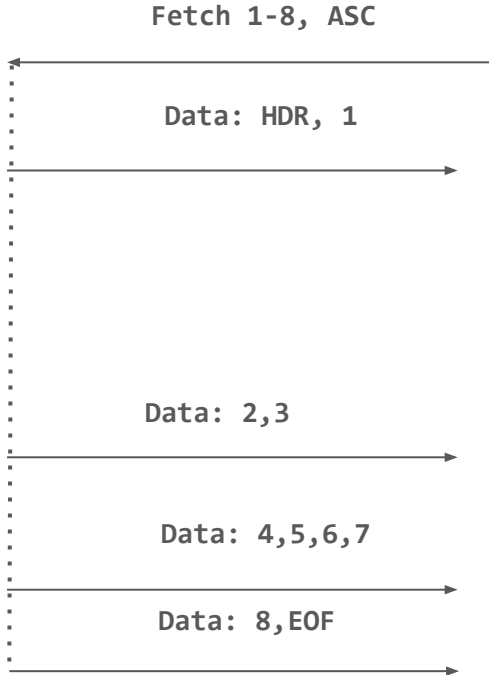
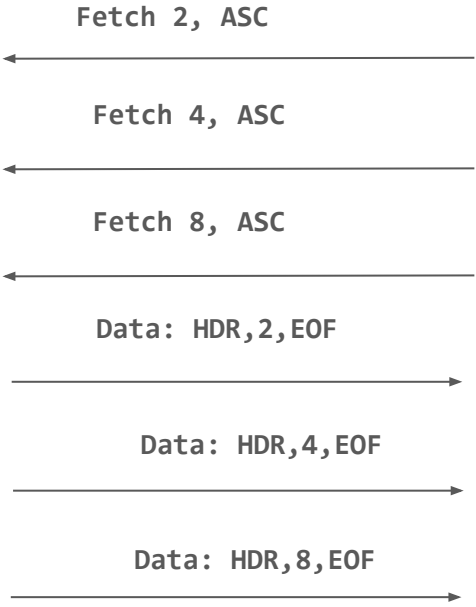
3

Successful Fetch Filling Gaps

Original Publisher

Relay

Client



4

Range Fetch with missing objects

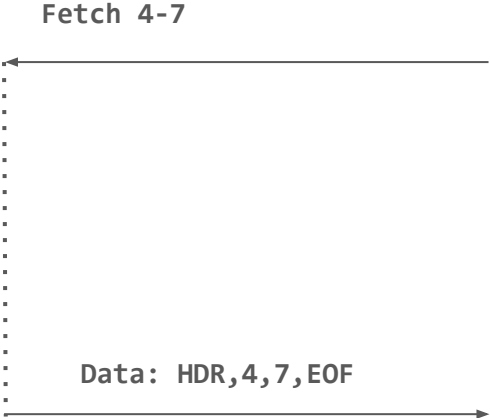
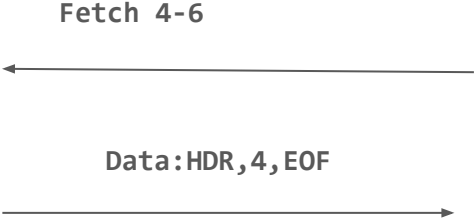
Original Publisher



Relay



Client



Improvement to select multiple ranges

```
FETCH {  
  Fetch ID (i),  
  Track Namespace (b),  
  Track Name (b),  
  Fetch Priority (8),  
  Group Order(i),  
  Range range(...)  
}
```

```
RANGE {  
  StartGroup (i),  
  StartObject (i),  
  EndGroup (i),  
  EndObject (i)  
}
```

1. Allows a client to retrieve multiple ranges with a single request.
2. Pipelined response in a single stream.
3. Guarantees order of responses
4. Ranges are processed independently (no deduping)

On to future ...

If we agree this is heading in the right direction:

- **PR on Fetch**
- **Then PR on Subscribe clarifications / simplification**

Fetch Properties

Solve for the past

Resolution in terms of ambiguities

- Communicates state of all objects within the range

Controlled delivery via QUIC stream-level flow control

Opportunistically Cached Content

Enables for overlapping and concurrent queries

Subscribe Properties

Solve for the future

Tolerant to losses, ambiguities

- State ambiguity due to loss/reordering

Data arrives at fixed rate, typically set by the Original Publisher. Consume it or lose it.

Subscriptions cannot be fulfilled by caches (since everything is in future) but can be saved in caches for later fetches.

Overlapping queries are never useful, since the content should be always the same

Smooth Transition

Victor

Add subscribe-id to fetch

When you hit the live edge,

Subscribe activates

Will

Explicit error on hitting live edge

Do Subscribe for live

Another fetch for any missing things

Luke

Error when you fetch to the future

Mo

Don't do blind requests

App subscription has no idea where it is in the group.

So subscribe first.

Use track status for identifying the live edge

Plan fetch to not pass the live edge

Cullen

SubscribeOk → use the info to find the current live edge
and request via Fetch

Alan - Similar to Victor ??

Atomic subscribe & fetch

Fetch use cases *[Slide copied from MOQ Denver Interim - Feb 7 2024]*

Chat :

Alice is offline for a bit and comes online. She will fetch the history by providing the right ranges known.

VOD Movie Playback

A client wanting to consume a pre-encoded movie would make a series of FETCH requests for ever increasing ranges.

Advertising insertion

The catalog instructs the client to retrieve a specific range of content which will be used for advertising display.

DVR:

Bob wants to watch a recording that is older than any Relay caches can tolerate. Bob issues fetch request with range and is willing to wait for the objects to arrive before playing out.

Quick Catch Up Side Channel:

Carl joins a meeting few mins late and wants to quickly catch up to the lost meeting minutes, independently off the live track. Carl fetches with the right range request to retrieve the objects and his player can play it at 1.5X speed.

DVR Clip

Alice wants to watch the last goal scored in the live soccer game