



Publisher initiated server-side ABR using a defined Extension

Will Law

IETF moq workgroup interim, Boulder Feb 2026

Problem Statement

Publisher initiated server-side ABR functions when the original publisher & an edge server decide, on recurring basis, which of a subset of tracks should be sent to the client.

1. There is a finite set of tracks (usually 2-6)
2. Throughput is used as the basis for deciding which tracks to forward. Other metrics such as packet loss, RTT may be important.
3. The client does not control this behavior and it happens automatically.
4. The change occurs at group boundaries.
5. We want to use existing MOQT constructs as much as possible.

General approach

The approach is:

1. Define a new Extension which an original publisher can use to describe track-specific ABR groups and throughput thresholds in the content that it produces.
2. An edge relay recognizes this extension and activates the ABR behavior.
3. The client must still subscribe to content via `SUBSCRIBE_NAMESPACE` in order to receive it.

Define a new ABR Extension

This is normatively defined with MOQT, or in a separate RFC if you prefer.

```
ABR extension {  
  Type (i) = 0xXX, // must be odd since we carry more than a single integer.  
  Length (i),  
  ABR Group Identifier (i), // defines the ABR group you want to activate or deactivate  
  ABR Group Size (i), // defines the # of tracks in the group.  
  Bitrate (i) // defines the switching throughput in kbps.  
}
```

Example

The original publisher produces content, decorated with our new extension

Namespace	Name	Extension payload
example.com/broadcast/123	video-high	1,3,3600
example.com/broadcast/123	video-medium	1,3,2400
example.com/broadcast/123	video-low	1,3,1200
example.com/broadcast/123	audio-high	2,3,4000
example.com/broadcast/123	audio-medium	2,3,3000
example.com/broadcast/123	audio-low	2,3, 76

Flow:

1. Original publisher PUBLISHES 6 tracks, each decorated with the new ABR extension
2. Client issues a SUBSCRIBE_NAMESPACE (example.com/broadcast/123)
3. As new groups arrive, the edge relay examines the object extension.
4. For each ABR group, it holds back delivery until it has objects from 3 tracks.
5. Based on its throughput estimate to the client, it forwards the object with the highest indicated bitrate that is lower than its throughput estimate, and discards the other objects.

ABR group

There are 3 tracks in this group

Select this track if throughput \geq 76kbps

Comments:

1. How does the edge relay know it's an edge relay? Do we leave this up to the network operator? If intermediate relays implement ABR in a delivery network, then the last mile doesn't have access to the full stack.
2. This proposal limits switching to group boundaries. If we want to enable it at Object boundaries, then we could add a bit flag to dictate group||object switching.
3. Implicit in the design is that the relay must wait until it has N objects available across the ABR group, before it releases one of them. To avoid ABR failing because one track is delayed, do we want to set a max time limit the relay is willing to wait for jitter alignment before making a selection?
4. Do we need to extend the design to enable more complex ABR algorithms, that depend on other metrics such as loss, RTT etc, to be implemented at the relay?

Thank you for your time.