Media QUIC

- Watch
- Publish
- Explain
- Source
- Discord

published 2/26/2026

# On a Boat

**Saronic** posted a **blog** about how they've been using MoQ for years. Go read it.

I'm stoked because I can *finally* talk about nautical nonsense.

## Bandwidth Constrained

Imagine this:

- You're on a boat, man.
- You're going fast, man.
- You've got 9 cameras and StarLink is expensive, man.


**obligatory**

It’s not sustainable to constantly transmit a full HD stream from every camera. Even if you can afford it, ship happens. Your bandwidth will fluctuate and camera feeds will drop out.

Existing contribution protocols (ex. WebRTC, SRT, RTSP, RTMP) are push-based. A publisher, like a security camera on a boat, constantly transmits *everything* to a server. Not good when you’re on a boat.

But **MoQ is pull-based***. We are the ugly duckling of the media publishing world.

* Well technically, the IETF nerds added an (optional) `PUBLISH` message to **moq-transport**, so it can do both. But I *strongly* discourage you from using it, and here’s why:

## LIKE, COMMENT, SUBSCRIBE

But first the basics.

A MoQ broadcast is split into live streams called “tracks”. It’s completely up to your application to decide how to split stuff into tracks. For example, **hang.live** uses:

- `1080p`
- `360p`
- `audio`
- `captions`

- `chat`
- etc

Every MoQ viewer has to explicitly `SUBSCRIBE` to each track, otherwise it won't be transmitted.

This is fantastic because now tracks are optional. Window minimized? Codec unsupported? Screen too small? Don't `SUBSCRIBE 1080p`.

What if 100 viewers `SUBSCRIBE 1080p`? **moq-relay** is the champ that merges them into a single *upstream* `SUBSCRIBE 1080p`. Your boat is safe from the flood and serves *at most* one copy of each track.

What if 100,000 viewers `SUBSCRIBE 1080p`? Run a **cluster** of relays. Sprinkle them around the world and suddenly you have a global CDN.

It's the exact same concept as HTTP CDNs, but for live streams. MoQ is being standardized specifically so you can use a **CDN like Cloudflare** to scale world-wide. Or you could self-host **moq-relay** (like Saronic) and keep everything in your network.

## Enter, Renditions

**Mr Skeptic Strawman** says:

> *"I have a single viewer, why should I care about mass fanout?"*

Well **Mr Skeptic Strawman**, let's talk about renditions.

In new reality land, you don't want a *human* viewer. You instead throw an AI model at that shit. The AI will subscribe to the 360p stream from every camera and try to detect anomalies. Like a kraken is attacking or something.

idk I don't sail.

By default, only the 360p stream for each camera is being transmitted. If AI detects a kraken on the starboard bow, then it subscribes to the 1080p stream.


Arr, there be a kraken on the starboard bow.

But what happens during congestion if we can't transmit everything? **Answer**: it's your choice buddy.

All MoQ subscriptions contain a priority, so they can cooperate while sharing a connection.

- **Do you want the kraken cam?** Probably, set the priority to 10.
- **Do you want to crash the boat?** Probably not, set the front-facing cameras to 5.
- **Do you want audio?** Idk I don't sail, maybe set it to 15.

The QUIC library will transmit packets based on the specified priorities. In this example: audio, then the kraken, then front cameras, then anything else. This is all best-effort, so your sloppy-seconds might get queued/dropped if the bandwidth is too low.

**Something** has to be queued/dropped, but we do our best to make sure the most important stuff gets transmitted. Hopefully it's the kraken.

## Backfill that Boat

Another MoQ difference is that we don't drop frames, we prioritize/queue.

**Mr Skeptic Strawman** says:

> *"huh?"*

So lets say the kraken is consuming our crew, and our available bandwidth. We get an alert that the kraken is attacking.

A human wants to see that shit in real-time and configures a maximum latency of 100ms. If a frame

takes longer than that, then we skip ahead. We don't want a spinny boye getting in the way of the action.

But let's say there's also a VOD worker recording the footage for the post-mortem. It can tolerate higher delays, configuring a maximum latency of 30 seconds.

Normally, a WebRTC publisher would have dropped that frame after 100ms, but MoQ doesn't. We instead deprioritize it, sending newer frames instead but keeping old frames in RAM.

**We know the kraken will eventually be satiated**. Once we recover, we might have enough spare bandwidth to backfill the VOD before the 30s deadline.

The end result is a lossy live stream AND a pristine VOD. Just because one viewer wants real-time latency doesn't mean the rest of the world has to suffer.

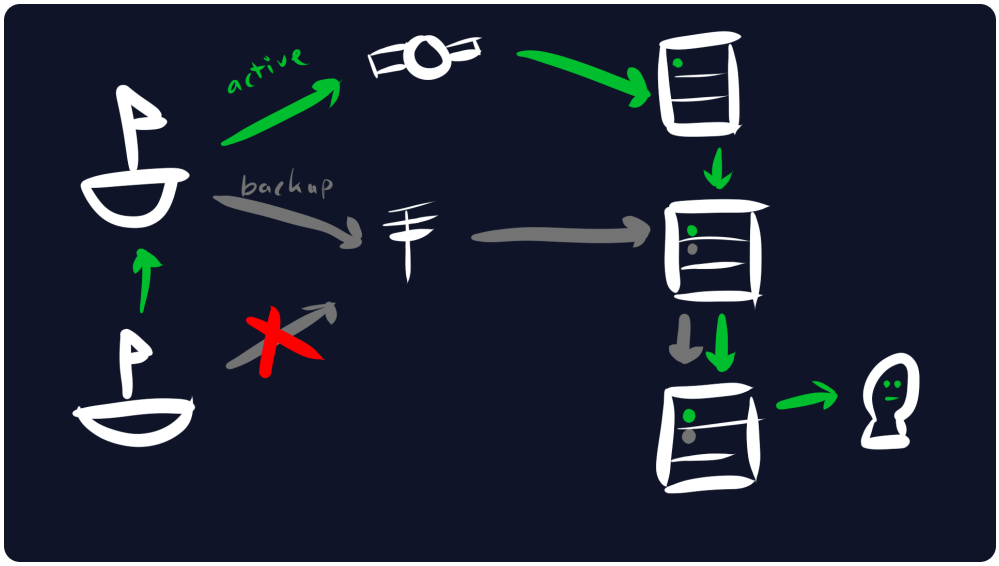
Literally huge.

## Bonded Pairs

What if you're near the coast? Can we use a cellular connection in addition to the satellite connection?

Why of course, rhetorical question. QUIC supports path migration, seamlessly switching between multiple paths (active-backup). You can switch from Wifi to cellular to satellite without severing the QUIC connection (unlike TCP).

There's also a **multi-path extension** that can use multiple paths at the same time (active-active). If you want to give it a spin, the **iroh** folks just released their **Quinn fork**. It's gated **behind a feature flag** if you want to try it with MoQ.




Like kittens, these pairs are bonded.

And that's only at the QUIC level. Another option is to establish multiple MoQ connections, advertising the broadcast as available on all of them. The viewer will automatically route subscriptions to the "best" connection. For example, via a P2P connection (via **iroh**) otherwise via CDN connection.

And remember, MoQ won't transmit a track until there's a `SUBSCRIBE` for it. You can establish multiple connections, possibly to different CDNs, and nothing will flow. The moment a viewer (on that CDN?) needs a track, then it will be transmitted.

Push-based protocols can't do this. **Pull-based is based.**

## The Wake not Taken

You need pretty huge buoys to be an early adopter. Saronic has been using my **moq.dev** libraries for maybe ~2 years now... even before I was a professional vagrant.

I want to thank them for their patience and support. They put up with bugs, overzealous renaming, and other shenanigans. They've got a pretty nice solution now and **are hiring**.

And there's a few other cant-be-named companies evaluating MoQ for *security purposes*. I can't say that was the plan... but it makes perfect sense. Hit me up if you want to chat about your use-case and how I can help. Even if I have to sign an NDA.

I'm just happy to see MoQ being used in the wild. **Dave Gullo** earns a shoutout for doing just that with **ooda.video**. He's using the dope **MoQ OBS plugin** we built in **Montevideo** to do all of the kraken detection stuff.

# Conclusion

MoQ is great for boats. And other vehicles. And stationary objects.

Want to learn more? I'm working on **more documentation**, including other non-boat **use-cases**.

Until next time sailor.