Good evening, everyone. Thank you for joining me to talk about Media over QUIC and its deployment into Content Delivery Networks.

I put resources for this meetup up at Vivoh.Earth, including a transcript of this presentation, in case you want to follow along, skip ahead, or leave early.

Before we dive into the technical details, allow me to spend one minute on my background with streaming media and content delivery, as it will explain why I am very passionate about the subject.

I created an Internet Cafe in Seattle in 1995, where I learned how to program the web via view-source for the front end and the O'reilly book on Learning Perl backend development. All open.

In 1997, I joined RealNetworks as an Application Engineer in their CDN group. I built streaming multimedia applications for broadcasters, including CNN and ABCNews, using RTSP and the Synchronized Multimedia Integration Language, which was an XML-based markup language for synchronizing multiple media sources for playback that included a Wall Clock feature for audience synchronization.

These apps were replaced by Adobe Flash in 2002 and then by HTML5 in 2010. Now they are scaled by CDNs with hundreds of thousands of caching servers, many of them are idle most of the day.

YouTube, Netflix, and Apple are deploying thousands of dedicated servers within ISPs. It’s depressing to think about the wasted energy consumption of this approach.

I didn’t learn how to author Flash and I found HLS to be pretty limited, so I focused instead on streaming within the enterprise, primarily via IGMP multicast, but, with QUIC, I am finally excited again about developing streaming applications for internet delivery. There is a lot of work to be done, which creates opportunities for innovation and disruption. Amazon Sye has a huge advantage today but this could change quickly!

To understand the significance of Media over QUIC, it's essential to look at what we gave up when we switched from true streaming media to the glorified progressive downloading of HLS. We gave up UDP.

Flash had a streaming technology called "Fusion" that supported unicast, IGMP multicast, and peer-to-peer delivery with the same connection. I appreciated their innovation but did not want to be locked into one vendor.

To scale low latency streaming video, we need to switch back to UDP. Amazon Sye already has UDP streaming with audience synchronization, which gives them their advantage.

In 2011, Google released WebRTC, which was finally standardized ten years later. Developers built UDP-based streaming applications with it and even used it to scale HLS via peer-to-peer.

However, it was not designed for one-to-many streaming or peer-to-peer sharing of HLS data, and so the vendors that have tried to make this work suffered when Google made changes to their libWebRTC browser library. libWebRTC is essentially a black box, locked up by Google, focused on the Google Meet use case.

I know that WebRTC is open source but I can only count on one hand the number of developers that I know who can actually compile libWebRTC. Only Google provides what goes into all browsers.

Enter QUIC and HTTP/3.

QUIC is a UDP-based transport protocol designed by Google and Cloudflare. It forms the foundation of HTTP/3, the latest version of the HTTP protocol. Unlike libWebRTC, browser vendors have implemented all of the essential components needed for Media over QUIC on their own, including: WebCodecs and WebTransport (WebTransport is QUIC in the browser).

QUIC offers a reliable and ordered option per stream or just raw unreliable datagrams. QUIC supports multiple streams per connection but does not specify how these are used for video delivery. This is why we need Media over QUIC (or something similar, like Kyber, which I can talk about later).

Media over QUIC enables the independent implementation of full-featured, interoperable, and use-case optimized media applications without needing to wait for Google.

The key part about MoQ, which I will talk about in detail later, is to use multiple, reliable and prioritized streams organized for video.

The adoption of QUIC and HTTP/3 has been rapid. According to data from Cloudflare, approximately 30% of internet traffic now uses HTTP/3. For Meta 75% of their traffic is QUIC, YouTube is not far behind, and Netflix delivers most of their HD content via QUIC today.

So, why is QUIC gaining such rapid traction? There are several key advantages:

1. Reduced latency: QUIC eliminates the need for multiple round trips during connection establishment.

2. Improved congestion control: QUIC has better mechanisms for detecting and responding to network congestion than TCP's “slow start” and its aggressive backoff schemes, leading to more efficient use of available bandwidth.

3. Connection migration: QUIC connections can survive network changes, such as when a mobile device switches from Wi-Fi to cellular data.

4. Built-in security: QUIC incorporates TLS 1.3 by default, ensuring that all connections are encrypted without the additional overhead of a TCP handshake to set up TLS.

Given these advantages, one might wonder, "Why not just use HLS via QUIC?" While this approach would indeed leverage some of QUIC's benefits, it doesn't fully solve one of the key issues in streaming: head-of-line blocking.

Head-of-line blocking occurs when a single lost packet can hold up the delivery of subsequent packets. This is particularly problematic in video streaming, where it can lead to buffering and playback interruptions. To mitigate this, video players setup and tear down many connections, which add latency. This occurs even with QUIC alone since data is flowing through the same connection stream, which is marked as reliable and ordered, versus raw datagrams, which would require the application to implement its own data recovery scheme.

You can see a screencast of QUIC for HLS on Vivoh.Earth by clicking on: “Inspecting YouTube”

To truly leverage the power of QUIC for media streaming, we need a protocol designed specifically for this purpose.

Media over QUIC, or MoQ, is a new protocol designed to optimize media streaming over QUIC connections. At its core, MoQ uses a Subscribe and Fetch model, which allows for more efficient delivery of media content.

You can see a screencast of what this protocol looks like by clicking on: “Inspecting MoQ” on Vivoh.Earth.

Let's break down some key concepts of MoQ:

1. Tracks, Groups, and Objects:
   - Tracks are collections of media organized for consumption as a whole, for example video and audio at one resolution
   - Groups are join-points in the content, like keyframes in a Group of Pictures, which are needed to start playback.
   - Objects contain either the metadata needed for relays or data, which can be encrypted and is inaccessible to the relay.

2. Sessions and Priorities:
   - Sessions are client-initiated bidirectional control streams with setup messages.
   - The protocol supports system-wide priorities, including for both publishers and subscribers.

3. Migration and Graceful Switchovers:
   - Migration allows relays to restart during active sessions by issuing a GOAWAY message, after which their clients reinitialize streams.
   - Publishers can announce network switches, allowing for graceful transitions. For example, encoders can switch between contribution networks as needed.

The advantages of MoQ are significant:

1. Reduced Latency

2. Universality: MoQ serves as a universal protocol for ingest, distribution, and playout … supporting end-to-end encryption and

avoiding repackaging overhead. RTMP is finally going to be replaced!

3. Relay Capabilities: MoQ allows for independent relays that can operate without needing to decrypt or transcode the content, enhancing privacy and efficiency. They function as a mesh without differentiating between the subscriber connections of clients or other relays.

Now let's explore how we can implement MoQ into a Content Delivery Network.

1. Global Pub/Sub Relay Network:
MoQ has a publish-subscribe design where content publishers push their streams to our network, and subscribers receive the content from the nearest relay.

2. Centralized Database of Tracks:
To manage our content effectively, you can use a centralized database of MoQ tracks and their origin servers. One option is to use Redis with global replication. We developed a global anycast enabled directory with CloudFlare Workers and Global Object Storage and published the source for this on Github.

3. Ingest and egress routing via global DNS or AnyCast

Regarding load testing, we ramped up thousands of connections to small rust-based MoQ relay nodes and saw very manageable CPU and RAM consumption.

In summary, what I love about Media over QUIC is that it is an open standard that is not controlled by any one vendor. This will spur innovation and create an opportunity for new start-ups in our industry.

At Vivoh.Earth, I published links to some great resources on Media over QUIC, including Luke Curley’s highly opinionated blog posts on the subject and great talks from just last month by Lorenzo Miniera and Ali Begen.

You can get the source from quic.video and join Luke’s Discord server to get help building the MoQ applications or just send me a message and I will give you my build steps.

Thank you for your time.