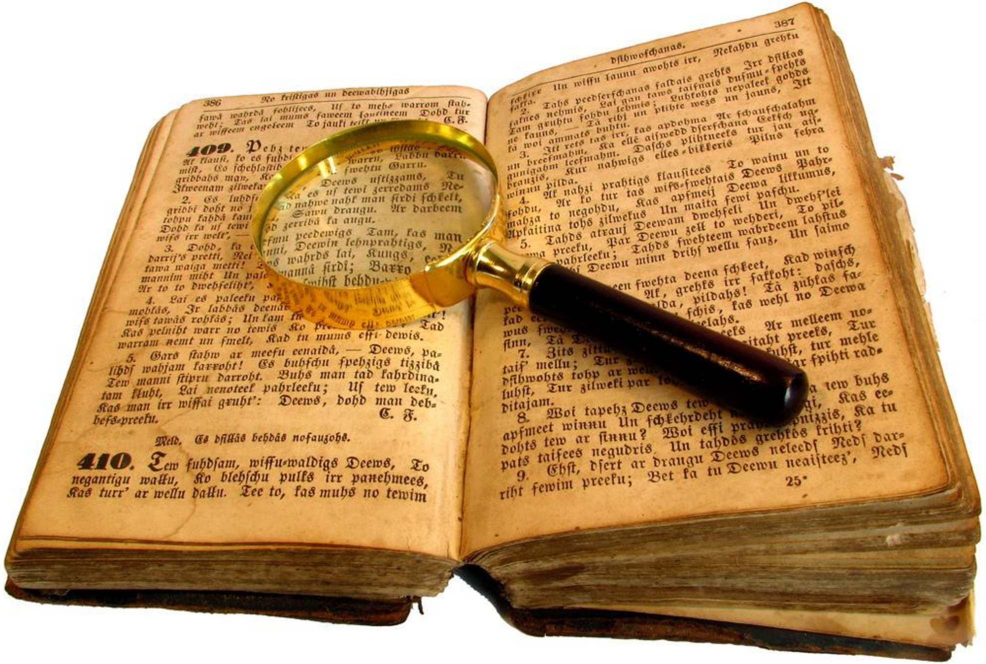
WebRTC QUIC

Status and future as of 10/2018

History

Original goals:

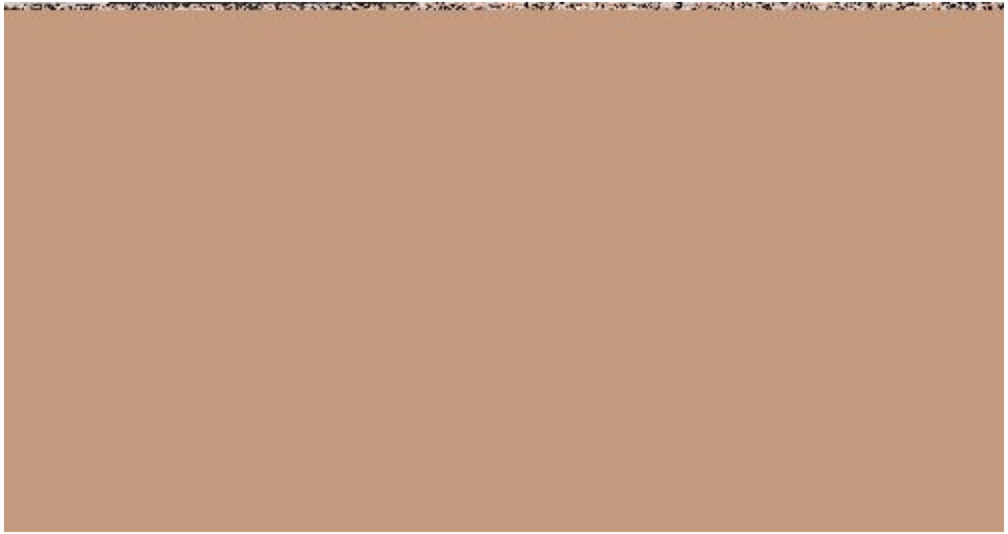
- RTCDataChannel, but with QUIC (and without SDP)
- Media over QUIC (maybe, eventually)



Basically help people like this

"... anyone's first reaction when looking at WebRTC is that it is unnecessarily complex for what it does—it's true that there is some unnecessary boilerplate in SDP and ... 4 handshakes!?"

"... I would be very interested in using QUIC instead of SCTP ... the weakest part of our implementation is SCTP"



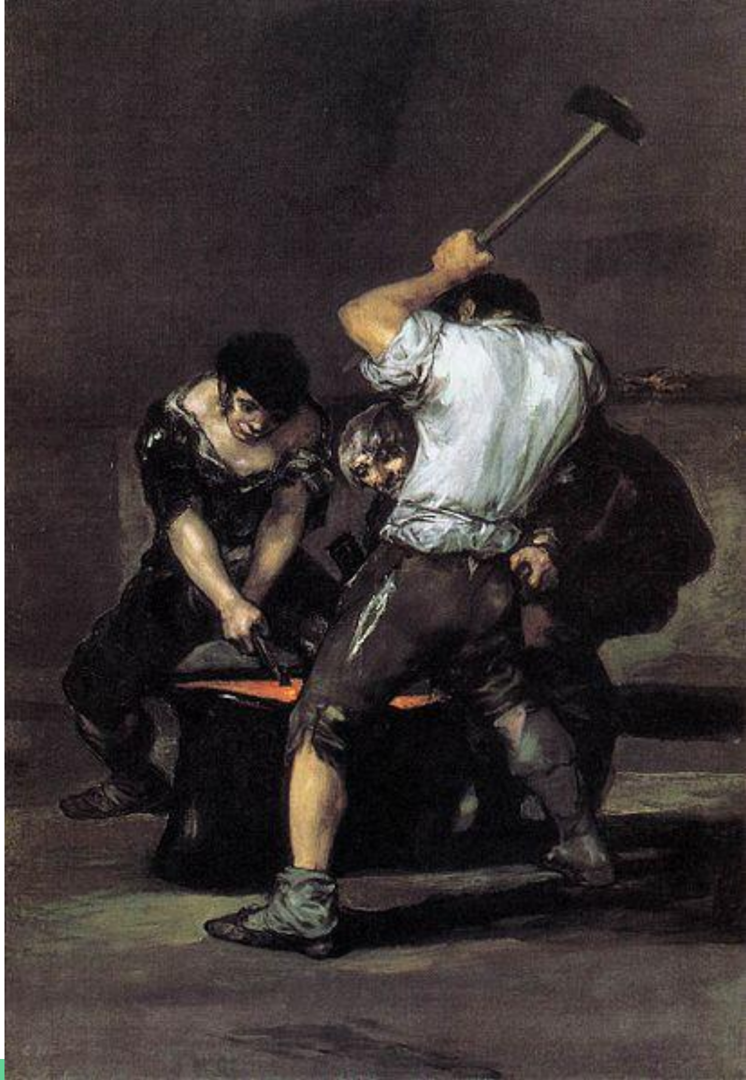
https://blog.parsecgaming.com/game-streaming-tech-in-the-browser-with-parsec-5b70d0f359bc

Some roadblocks encountered (and routed around)

- QUIC transport draft does not yet have functionality needed to support all RTCDataChannel features
    - Example: maxRetransmits > 0
- No IETF WG available to work on data channel protocol for QUIC.
    - RTCWEB WG shutting down, QUIC WG focussed on HTTP only (and behind schedule)
- RTCDataChannel API not optimal for some applications.
    - Where stream abstraction is preferred, should we emulate streams on top of RTCDataChannel on top of a QuicStream?
    - Why not provide native access to QuicStreams?
- Revised goal: support native stream abstraction enabling implementation of RTCDataChannel API on top of it.
    - Similar requirement: ORTC ability to emulate RTCPeerConnection

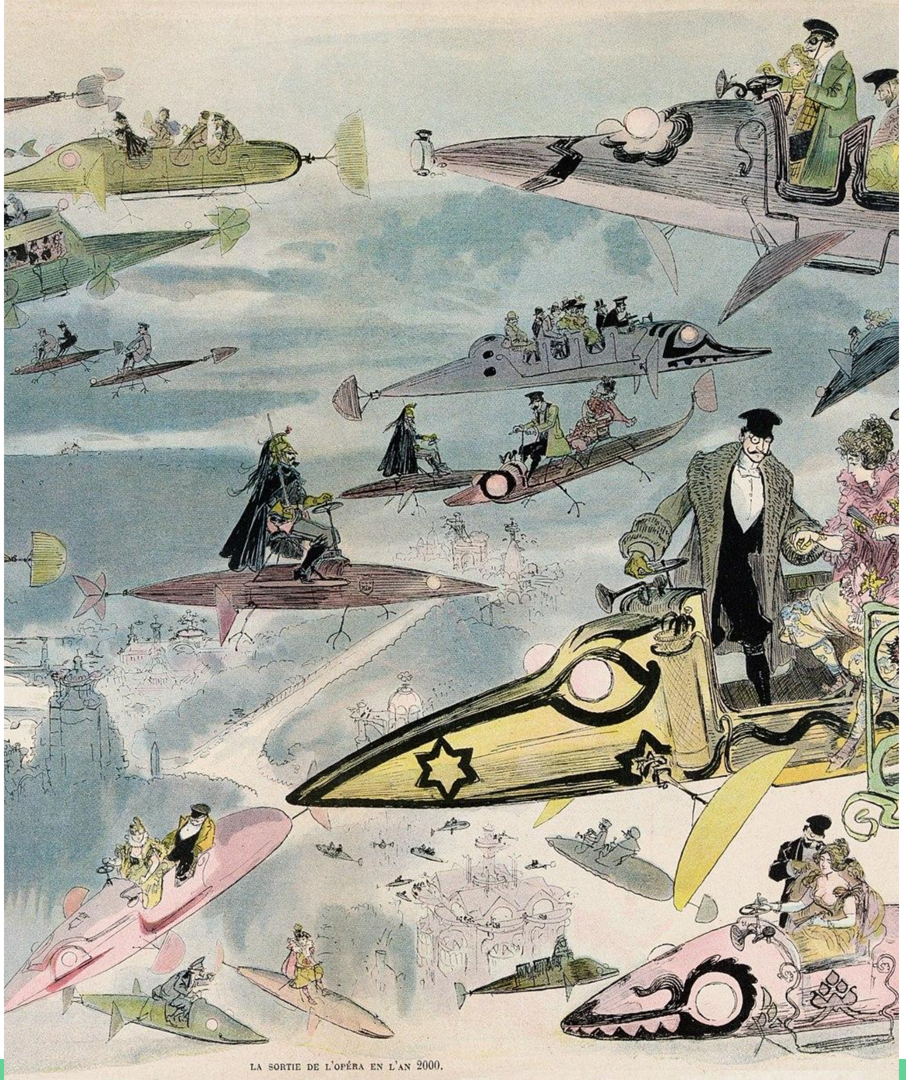
Where we're at

- A spec that meets the original goals
- An implementation in Chromium coming in an original trial soon
- A spec matured a lot from implementation experience
- Compatibility with recent QUIC drafts, and support for the recently added QUIC features (e.g. unidirectional streams)
- Proposal for multiplexing QUIC with RTP/RTCP/DTLS/STUN/TURN
- Adoption in ORTC CG
- No adoption (yet?) in WebRTC WG



Where we're headed

- [Considering WHATWG streams]
- [Figuring out how to get off the main JS thread]
- Considering client/server use case (like websockets, without ICE)

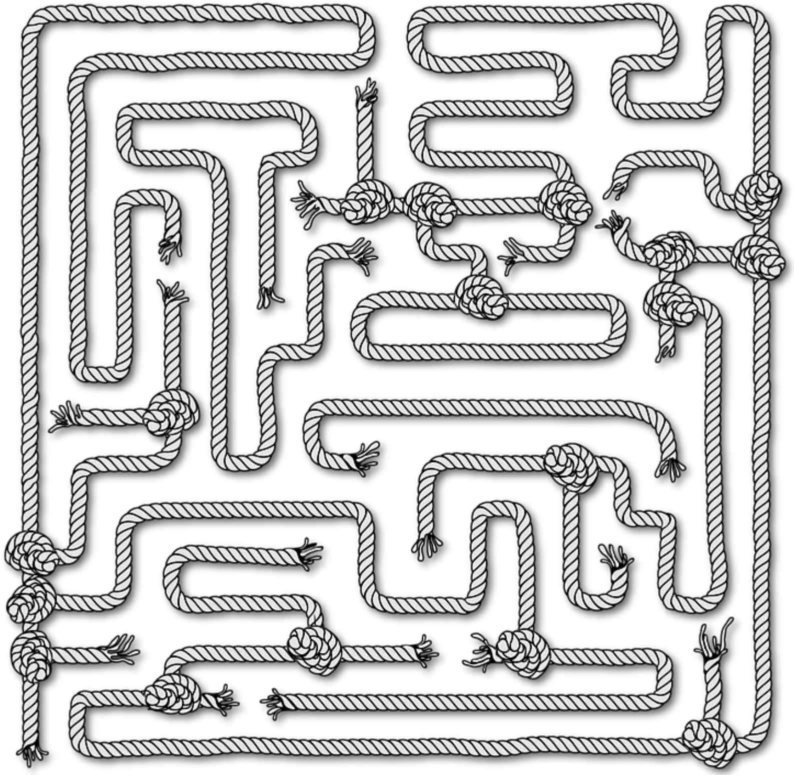


Interesting things we've learned

- Lots of people want a server->client QUIC (either as "unreliable websocket" or "replacement for Server Push" or "please no SCTP on my server")
  - [Media and Entertainment Interest Group]
- SCTP + MSE is a thing
  - https://blog.rainway.io/building-a-cross-browser-cross-platform-real-time-game-streaming-protocol-b00d000fe788
  - https://blog.parsecgaming.com/game-streaming-tech-in-the-browser-with-parsec-5b70d0f359bc
- Lots of people are very afraid of WebRTC
  - When we say "RTCQuicTransport does what you need", they say something like "ewww... RTC.... "
  - Lots of blog entries like "I fought this beast and survived, but it's complicated"
  - They think WebRTC == PeerConnection (which is bad)

Questions for the WebRTC WG

1. Does WebRTC WG wish to adopt QUIC APIs? (or let the work continue in the ORTC CG)
2. Does WebRTC WG want to support client/server use cases or not? (or let another WG take that on)
3. Does WebRTC WG wish to adopt WHATWG streams across the board? (DataChannel, RtpSender, RtpReceiver, QuicTransport)



Repeated Discussions That We'll Repeatedly Repeat

"Let's copy the websocket API"
It has major faults, like no receive back pressure
The TAG specifically says not to copy it

"Let's copy the RTCDataChannel API"
It has the same problems as websockets

"We just want a QUIC websocket; we don't want RTC"
That's easy to do with ICE-lite
But we can also add a constructor w/ a URL (like ws)

"But how will QUIC do unreliability?"
1. Message-per-stream + time-based cancel (Already there)
2. Message-per-stream + "don't retransmit" flag (coming soon?)