WebTransport + WebCodecs

at W3C Games Workshop 6/19
Problem 1: WebRTC not great for cloud gaming
Problem 1: WebRTC not great for cloud gaming
Problem 2: MSE not great for cloud gaming
Problem 1: WebRTC not great for cloud gaming
Problem 2: MSE not great for cloud gaming
Problem 3: WebSocket not great for gaming
Problem 1: WebRTC not great for cloud gaming
Problem 2: MSE not great for cloud gaming
Problem 3: WebSocket not great for gaming
Solution: WebTransport + WebCodecs
HTTP + MSE

WebTransport + WebCodecs!

WebRTC
I'm the guy that made Agar.io, Diep.io and a few smaller games. I analyzed the possibility of using WebRTC in my games several times so far, but it seems that right now, it's still hard to use in a server-client architecture. You need to bring this [1] behemoth and all of its dependencies to your project dependencies on the server side, even though you only care about a tiny bit of it (unreliable data channels). It's unlikely that people will start using it until there is an easy stripped-down version that only deals with data channels.
Why can't I send UDP packets from a browser?

A solution for enabling UDP in the web

Posted by Glenn Fiedler on Sunday, February 26, 2017

Premise

In 2017 the most popular web games like agar.io are networked via WebSockets over TCP. If a UDP equivalent of WebSockets could be incorporated into browsers, it would greatly improve the networking of these games.
All user agents delay the start of playback until some minimum number of frames are decoded and ready for rendering. This buffer provides an important cushion against playback stalls that might otherwise be caused by intermittent decoder slowness.

It also adds a small amount of latency to start playback. For example, in Chromium this adds roughly 200 milliseconds for common video frame rates.
DatagramSocket

public class DatagramSocket
extends Object
implements Closeable

MediaCodec

public final class MediaCodec
extends Object
WebTransport

WebCodecs

DatagramSocket

public class DatagramSocket
extends Object implements Closeable

MediaCodec

public final class MediaCodec
extends Object
UDP isn't secure!

- encryption
- congestion control
- CORS/consent

If only there were a way to add that to UDP but keep all the good parts of UDP...
HTTP/3
What is QUIC?

- Fast connection setup (1-RTT, or sometimes 0-RTT)
- Secure
- Low-latency congestion control (pluggable)
- Many reliable streams (like TCP/TLS * N)
- Unreliable/unordered datagrams (like UDP)
- Can be p2p (with ICE)
- Basis of HTTP/3
- Widely deployed
- Many implementations coming
Benefits for games

- Faster game loading
  (particularly with many components).
- More network resilience: making bad networks usable
  (particularly for mobile network with high RTT/loss)

But what about cloud gaming?
Streaming with WebRTC stack

Server

ICE, DTLS, SRTP

Browser
(Webrtc stack)

Encode/Forward, Packetize

Depacketize, Buffer, Decode, Render
Streaming with WebRTC stack

Server: Encode/Forward, Packetize

ICE, DTLS, SRTP

Browser (WebRTC stack): Depacketize, Buffer, Decode, Render

"Hard to use in a client-server architecture"

Not a lot of control in buffering, decoding, rendering. All controlled by browser.

Limited by RTP (no generic data)
Streaming with WebRTC Data Channel + MSE

Server
- Encode/Forward, Serialize

ICE, DTLS, SCTP

Browser
- Decontainerize, Buffer, Decode, Render

JS
- Containerized Media
- data
Streaming with WebRTC Data Channel + MSE

"Hard to use in a client-server architecture"

Low-latency mode is implicit magic

Have to containerize media just to get it in

Server

Encode/Forward, Serialize

ICE, DTLS, SCTP

Browser

(RTCDataChannel + MSE)

Decontainerize, Buffer, Decode, Render

JS

data

Containerized Media
Streaming with WebSocket + WebAssembly

Server

WebSocket, TCP

Encode/Forward, Serialize

Browser (WebSocket)

wasm

Deserialize, Buffer, Decode

data

Raw Media

Render
Streaming with WebSocket + WebAssembly

Head-of-line blocking
Performance/latency
Quality
Battery life

Server

WebSocket, TCP

Encode/Forward, Serialize

Browser (WebSocket)

Render

wasm

Deserialize, Buffer, Decode

data

Raw Media
We can do better
We can do better transport
WebTransport is a better RTCDataChannel and a better WebSocket

- Reliable/ordered and unreliable/unordered
- Easy to use in a client/server architecture
- Client/server and p2p
- Provides datagram support
- Same security properties as RTCDataChannel and WebSockets (encryption, congestion control, CORS)
- Faster!
- Better API (support for back pressure)
We can do better
We can do better codecs
MediaRecorder → RtpSender → MSE → RtpReceiver

Encoders and Decoders

Like mobile APIs, but hidden

"built-in encoder makes sense so we don't have to ship WebAssembly modules to do the same thing"
Streamlining with WebTransport + WebCodec

- Easier to deploy server
- Low latency
- More web developer control

Diagram:
- Server -> QUIC -> Browser
  - Server: Encode/Forward, Serialize
  - Browser: WebTransport + WebCodec
    - JS: Decode, Render, Deserialize, Buffer
    - Raw Media

...
Other use cases/benefits

- Push game state (with low latency)
- Stream game assets (with low latency)
- Communication during a game (with same APIs)
- Upload media to server for ML (with low latency)
- Transcoding (faster than real-time)
- Get new codec support faster (less for a browser to implement)
- Get new/wider container support (from JS library)
- Better support for spatial/temporal scalability
- e2e encrypted group communication
WebTransport Status:
- has origin trial (p2p version)
- looking for customers

WebCodecs Status:
- has proposal/explainer
- looking for interest
More Info

• RTCQuicTransport Origin trial
  • Announcement: https://developers.google.com/web/updates/2019/01/rtcquictransport-api
  • Documentation: https://github.com/shampson/RTCQuicTransport-Origin-Trial-Documentation
  • Sample code: https://webrtchacks.com/first-steps-with-quic-datachannel/

• WebTransport
  • Proposal/Explainer: https://discourse.wicg.io/t/webtransport-proposal/3508
  • Spec: https://wicg.github.io/web-transport/

• WebCodecs
  • Proposal/Explainer: https://discourse.wicg.io/t/webcodecs-proposal/3662
  • Repository for future spec: https://github.com/pthatcherg/web-codecs
Topics for discussion

We are **looking for customers**

- What use cases do you have for secure datagrams or low-level codecs?
- What would you like to see in these APIs?
- **Use the RTCQuicTransport origin trial!** (If you run into issues with client/server, let me know... I've got a demo server almost working at [https://github.com/pthatcherg/quic-go/tree/gquic/example/webtransport](https://github.com/pthatcherg/quic-go/tree/gquic/example/webtransport))
Other things I couldn't fit into the slides

- HouseParty/EPIC acquisition
- Data with audio/video at the same time
- Spatial audio
- Temporal/spatial scalability
- Study of origin trial vs SCTP: https://docs.google.com/document/d/1F0lfp6z2bLitqDrbvcRewkKBr20a2R snoFOwceKUWk/edit#