WebTransport

Will Law

W3C WebTransport WG Co-Chair
Akamai - Chief Architect

September 7, 2021

Experience the Edge

What real-time data applications would we like to build on the web?

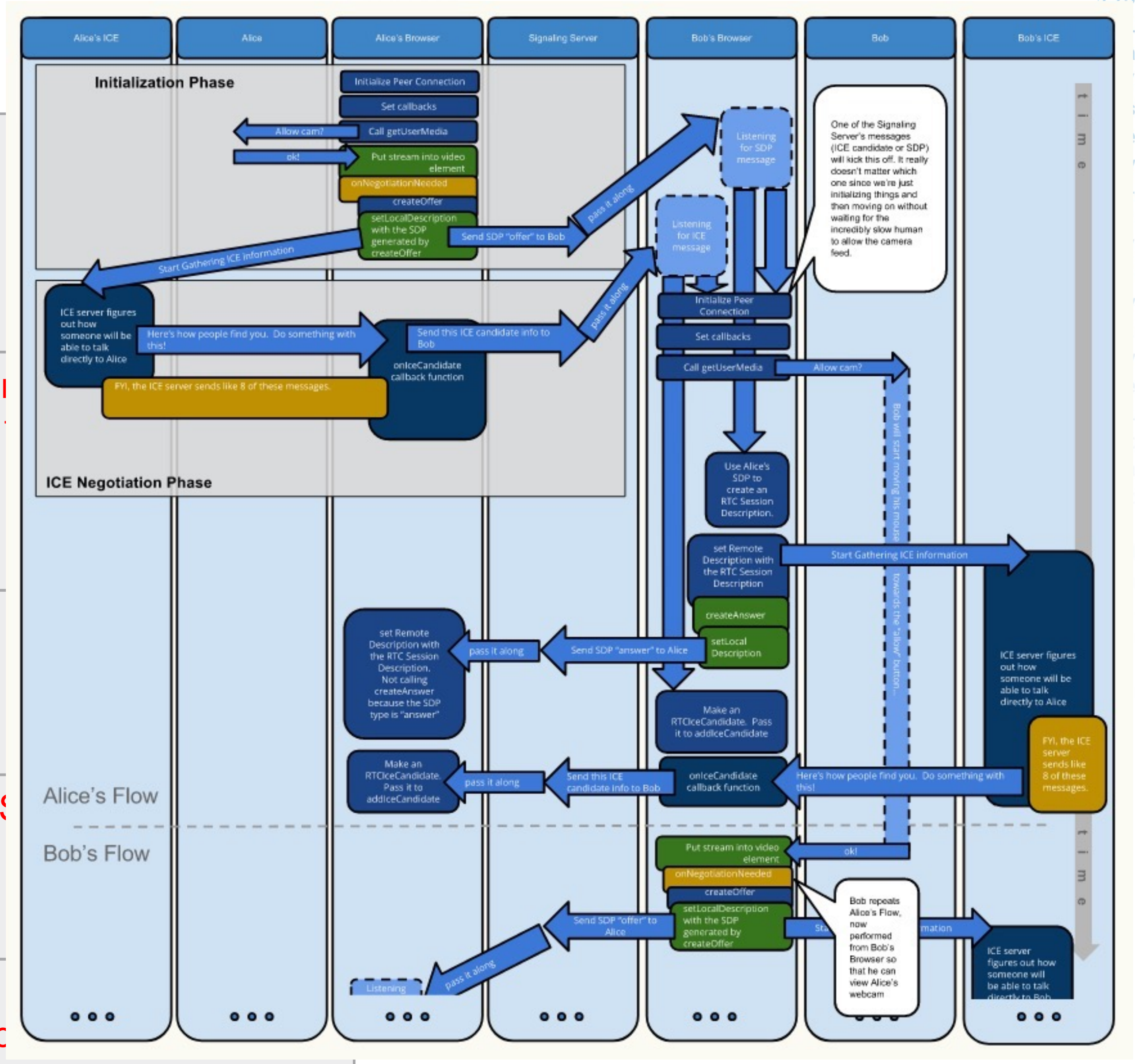
- Real time audio and video communications apps with improved privacy, performance & simplicity.
- Multiplayer game play communication & orchestration.
- Cloud Game Streaming.
- Low latency video delivery, for sports, news and industrial camera analysis.
- IOT sensor and analytics data transfer, such as vehicle location.
- Pub/Sub messaging platforms.
- Input & response for real-time speech translation.

Core requirements across all these use-cases

- The security protections of the modern web (TLS encryption, congestion control, CORS)
- Client-server architecture
- Bi-directional communication
- Send reliable and ordered data (streams) with minimal latency
- Send unreliable and unordered datagrams with minimal latency
- Continuously maintain consent to send data (back pressure)
- Identifiable using a URI

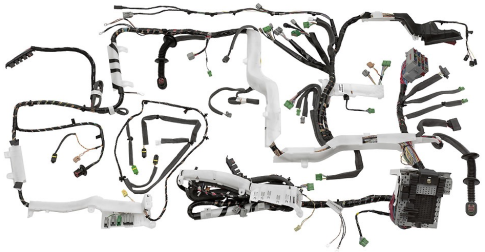
Protocol Options

REST API (HTTPS)	• Slow connection establishment • Lossless delivery requires retransmission which • High header overhead for small amount of data • No option for fast, unreliable delivery
WebSockets	• Head of line blocking - all messages must be se order even if they are independent and some of needed • No option for fast, unreliable delivery
WebRTC Data Channel	• High connection establishment overhead due to
Roll your own UDP transport	• Poor interoperability since you must support an and server
Chunked encoded segment media via H1/H2	• Slow connection establishment • Segments must be requested, RTT between eac

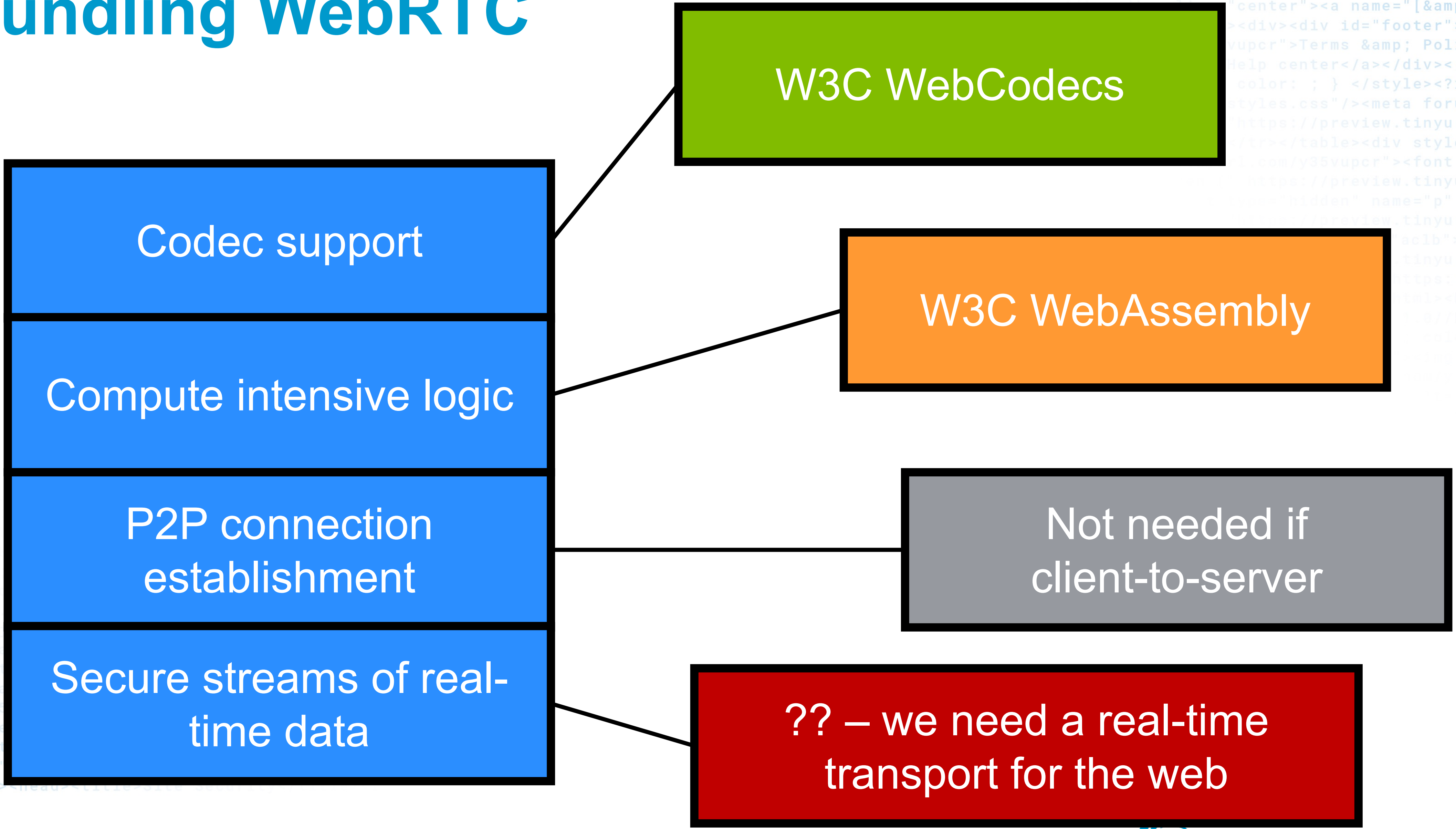


Akamai Experience the Edge

Unbundling to achieve specialization



Unbundling WebRTC



Welcome to WebTransport

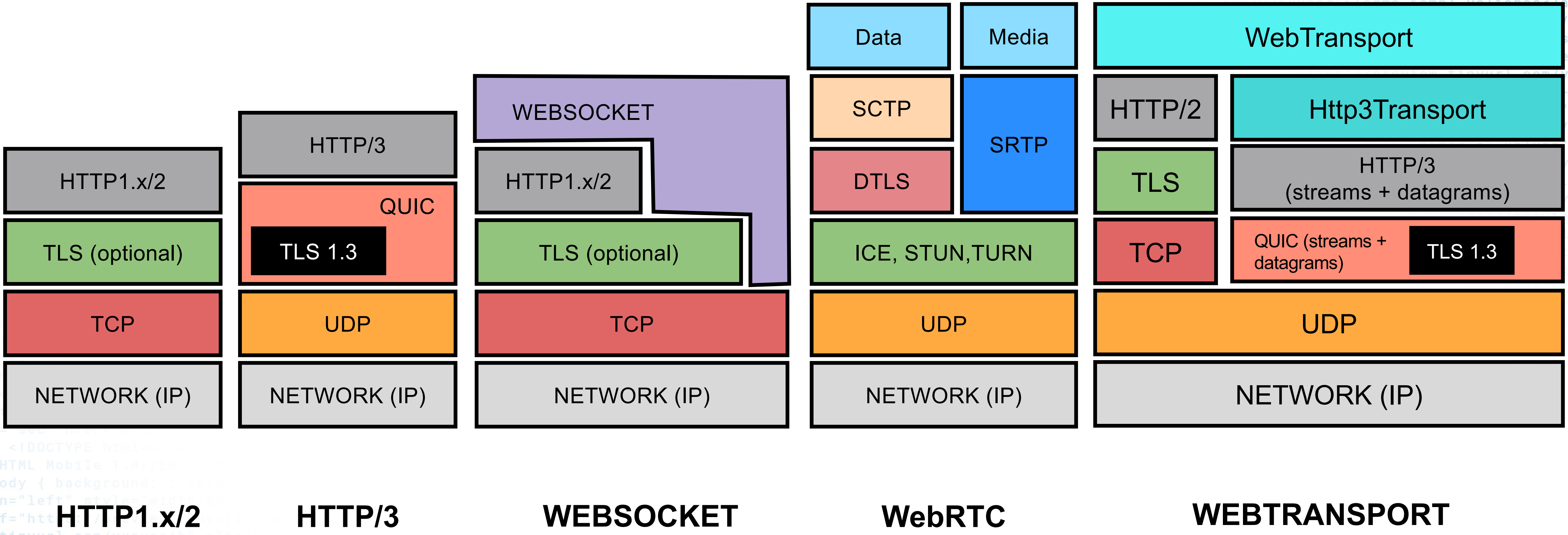
WebTransport solves the real-time data problem for the internet.

It is a **transport protocol** (specified by the IETF) and an easy-to-use **Web API** (specified by the W3C), that enables **clients** operating under the **Web security model** to communicate with a remote **server** using a **secure, multiplexed, real-time transport**.

WebTransport provides:

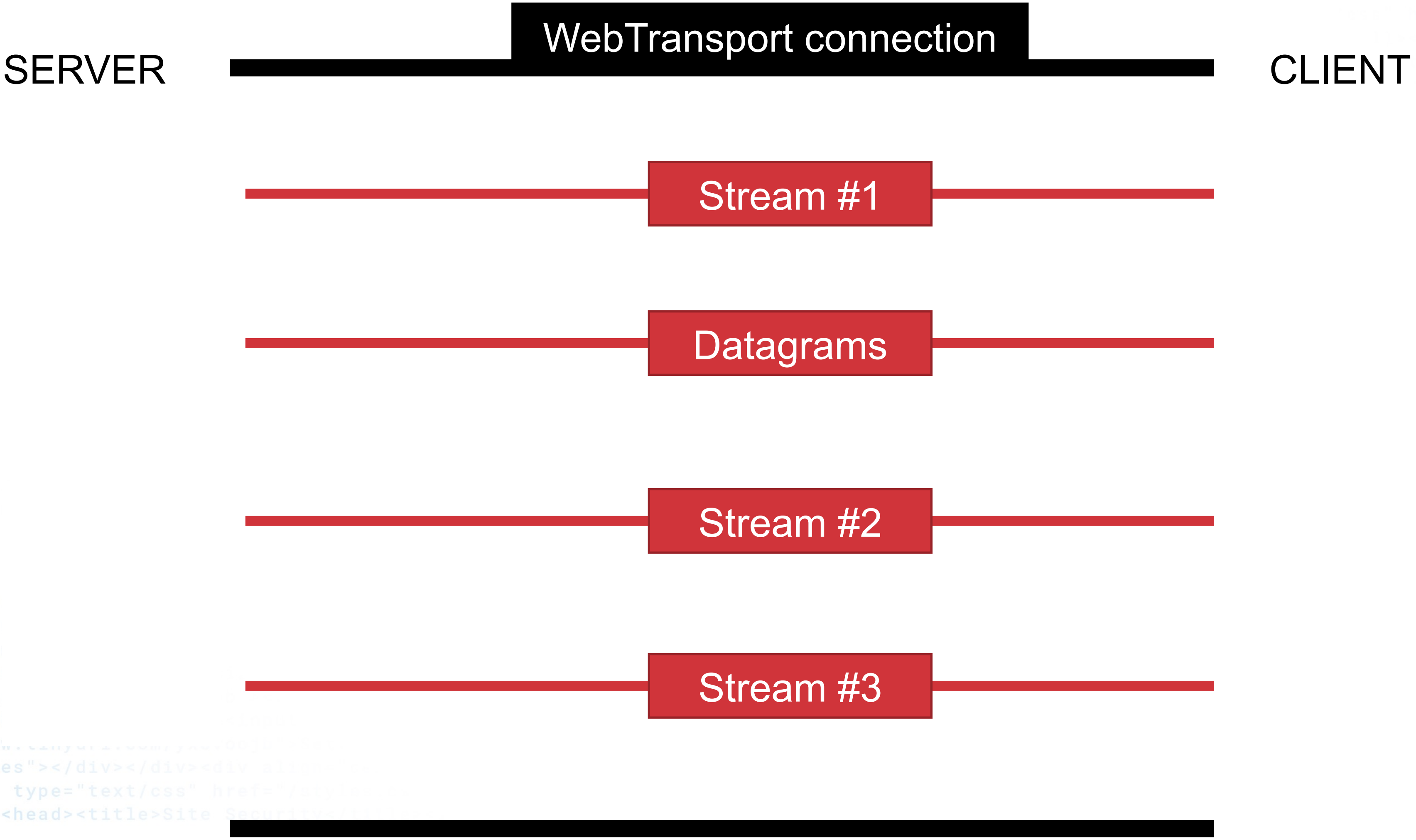
- multiple uni-directional and bi-directional streams of reliable and ordered data.
- an unreliable flow of UDP-like datagrams
- operation over HTTP/3 with fallback to HTTP/2

The stack



HTTP1.x/2
TLS (optional)
TCP
NETWORK (IP)
HTTP/3
QUIC
TLS 1.3
UDP
NETWORK (IP)
WEBSOCKET
HTTP1.x/2
TLS (optional)
TCP
NETWORK (IP)
Data
Media
SCTP
DTLS
SRTP
ICE, STUN,TURN
UDP
NETWORK (IP)
WebTransport
HTTP/2
Http3Transport
TLS
HTTP/3
(streams + datagrams)
TCP
QUIC (streams + datagrams)
TLS 1.3
UDP
NETWORK (IP)
HTTP1.x/2
HTTP/3
WEBSOCKET
WebRTC
WEBTRANSPORT

WebTransport transfer modes



Ordered and reliable

Unordered, unreliable & fast

Unordered but reliable. Use to send objects larger than one packet quickly

API Overview

- The API is in Public Working Draft status and available at https://w3c.github.io/webtransport/
- Offered under Secure Context (https) only
- A modern API that leverages web platform primitives such as Streams and Promises and works well with async and await.
- WebTransport URLs must begin with https and must specify the port.
- Can run in Web Workers

WebTransport

Editor's Draft, 18 August 2021

This version:
https://w3c.github.io/webtransport/

Latest published version:
https://www.w3.org/TR/webtransport/

Feedback:
public-webtransport@w3.org with subject line

Issue Tracking:
GitHub
Inline In Spec

Editors:
Bernard Aboba (Microsoft Corporation)
Victor Vasiliev (Google)
Yutaka Hirano (Google)

Former Editors:
Peter Thatcher (Google)
Robin Raymond (Optical Tone Ltd.)

Code example #1: Sending a buffer of datagrams

```
async function sendDatagrams(url, datagrams) {
    const wt = new WebTransport(url);
    const writer = wt.datagrams.writable.getWriter();
    for (const datagram of datagrams) {
        await writer.ready;
        writer.write(datagram).catch(() => {});
    }
}
```

Code example #2: Receiving datagrams

```
async function receiveDatagrams(url) {
    const wt = new WebTransport(url);
    for await (const datagram of wt.datagrams.readable) {
        processTheData(datagram);
    }
}
```


Code example #3: Sending data over a stream

```
async function sendData(url, data) {
  const wt = new WebTransport(url);
  const writable = await wt.createUnidirectionalStream();
  const writer = writable.getWriter();
  await writer.write(data);
  await writer.close();
}
```

Code example #4: Receiving a stream and leveraging piping

```
async function receiveText(url, createWritableStreamForTextData) {
  const wt = new WebTransport(url);
  for await (const readable of wt.incomingUnidirectionalStreams) {
    try {
    await readable
      .pipeThrough(new TextDecoderStream("utf-8"))
      .pipeTo(createWritableStreamForTextData());
   } catch (e) {
    console.error(e);
   }
}
```


Give it a try …

- Server examples
  - AIOQuic (will be used for Web Platform Tests) https://github.com/aiortc/aioquic/blob/main/examples/http3_server.py
  - Google Chrome samples https://github.com/GoogleChrome/samples/tree/gh-pages/webtransport

- Client examples
  - Chrome https://webrtc.internaut.com/wt/ (Chrome has had a WebTransport origin trial since v84+)
  - Client demo https://googlechrome.github.io/samples/webtransport/client.html

What doesn’t WebTransport provide?

WebTransport is not the answer to everything. It does not give you:

- Support of p2p connections – WebRTC is still best for that
- A built-in messaging framework, a la WebSockets.onmessage
- Native framing for audio or video payloads – no RTP or RTCP provided as part of the spec.

These omissions are by design. The creators want it to be a low-level tool. They envisage flexible libraries being used to implement application specific behaviors.

What is exciting about WebTransport?

- A chance to unify the transport and API between
  - Video conferencing & telephony applications
  - Gaming
  - Low latency & live media delivery
- It will look like Http/3 to firewalls, proxies, network switches etc. This can greatly facilitate its reach and robustness.
- Browser support gives you billions of addressable clients (in addition to native OS support).
- Datagram access in JavaScript ☺
- When combined with WebCodecs and WebAssembly, closes the gap between native and browser RTC applications.

Status as of Sept 2021

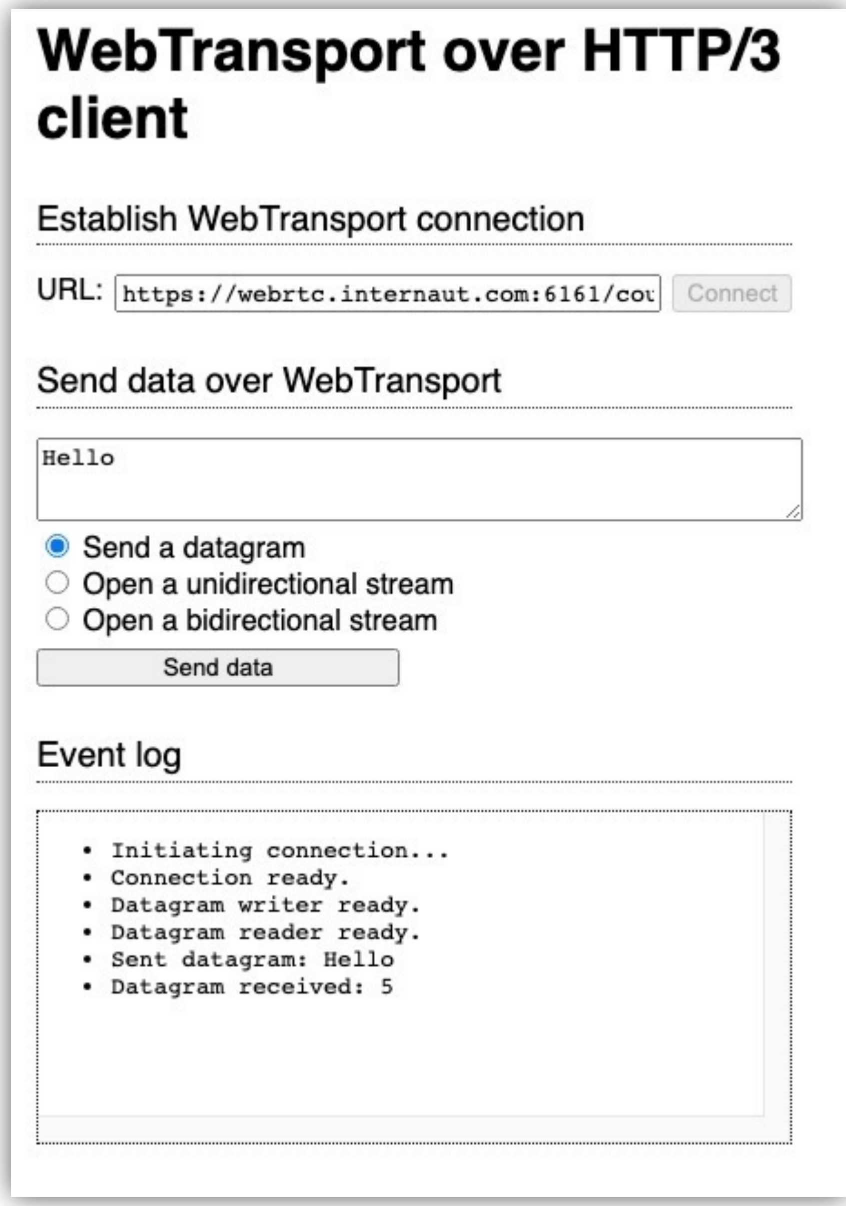
- Chrome have signaled intent to ship WebTransport around Nov 2021. Firefox are implementing but have not yet announced a release date.
- An echo server for Web Platform Tests will soon be available.
- Experimentation and feedback are welcome!
- IETF and W3C are planning the first interop event as soon as public server(s) are available. We'll be communicating more about that shortly.

Akamai Experience the Edge

WebTransport summary

- **WebTransport is solving the real-time data problem for the internet**
- WebTransport is a protocol (specified by the IETF) and a Web API (specified by the W3C), that enables clients constrained by the Web security model to communicate with a remote server using a secure, multiplexed, real-time transport.
- WebTransport provides for uni-directional and bi-directional streams of reliable ordered data between a client and server, as well as an unreliable flow of UDP-like datagrams.
- WebTransport uses modern Web Platform features such as streams, promises and Http/3 and provides more flexible solutions than those currently provided by WebSockets and WebRTC.
- The W3C WebTransport API is currently in First Public Working Draft status. An initial browser implementation is available, along with a server for Web Platform Tests. Experimentation is encouraged and feedback can be provided by filing a github issue at https://github.com/w3c/webtransport/issues

Live demo



WebTransport over HTTP/3 client
Establish WebTransport connection
URL: https://webrtc.internaut.com:6161/cou
Connect
Send data over WebTransport
Hello
Send a datagram
Open a unidirectional stream
Open a bidirectional stream
Send data
Event log
Initiating connection...
Connection ready.
Datagram writer ready.
Datagram reader ready.
Sent datagram: Hello
Datagram received: 5

Thank you for your time.

Questions?