Transport Services
API for WebRTC

Tommy Pauly
TAPS
IETF 103, November 2018, Bangkok
Background

WebRTC over QUIC proposal has inspired discussion around applying a transport-independent API

https://w3c.github.io/webRTC-quic/

Existing low-level APIs are transport-specific (RTCSctpTransport, RTCDtlsTransport, RTCIceTransport)
Raised Questions

Is Rendezvous sufficiently specified?

Do the states transitions match?

Does TAPS have any API gaps?

Are the data transfer models compatible?
Rendezvous

Resolve candidates (*RTCIceTransport*)

```java
[]Preconnection := Preconnection.Resolve()
```

Establish Connections (*RTCQuicTransport/RTCSSCTPTransport*)

```java
Preconnection.Rendezvous()
Preconnection -> RendezvousDone<Connection>
```

For established Connections on multi-streaming protocols, is the delivered Connection a specific stream?
State Transitions
Comparing TAPS to WebRTC proposal

<table>
<thead>
<tr>
<th>TAPS States (Section 9)</th>
<th>TAPS Events (Section 11)</th>
<th>QUIC WebRTC States</th>
</tr>
</thead>
<tbody>
<tr>
<td>Establishing</td>
<td>--</td>
<td>Connecting</td>
</tr>
<tr>
<td>Established</td>
<td>Ready</td>
<td>Connected</td>
</tr>
<tr>
<td>Closing</td>
<td>--</td>
<td>--</td>
</tr>
<tr>
<td>Closed</td>
<td>Closed</td>
<td>Closed</td>
</tr>
<tr>
<td>Closed</td>
<td>ConnectionError</td>
<td>Failed</td>
</tr>
</tbody>
</table>
API Gaps
Supporting Stop-Sending

WebRTC over QUIC proposes abortReading

A hard shutdown of the ReadableStream.

STOP_SENDING frame in QUIC

TAPS does not currently include this notion

When is this required as opposed to closing both with RST_STREAM+STOP_SENDING?
Data Transfer

TAPS provides Message semantics as well as Cloning connections for multiplexing

RTCQuicTransport presents a Stream abstraction

RTCSctpTransport presents a Data Channel abstraction
Data Transfer
Sending Data

**TAPS**

```java
Connection.Send(
    messageData,
    messageContext,
    endOfMessage)
```

**QUIC WebRTC**

```java
writable/writeBufferedAmount/writingAborted

write(buffer, length)
waitForWriteBufferedAmountBelow()
abortWriting()

dictionary RTCQuicStreamWriteParameters
{
    Uint8Array data;
    boolean finished = false;
}
```
Data Transfer
Receiving Data

**TAPS**

```java
Connection.Receive(
    minIncompleteLength,
    maxLength)
```

**QUIC WebRTC**

```java
readable/readableAmount/readingAborted
readInto(buffer, length)->result
waitForReadable()
abortReading()
```

```java
dictionary RTCQuicStreamReadResult
{
    unsigned long amount;
    boolean finished = false;
}
```