W3C WebRTC WG Meeting

July 18, 2023
8 AM - 10 AM

Chairs: Bernard Aboba
Harald Alvestrand
Jan-Ivar Bruaroey

W3C WG IPR Policy

- This group abides by the W3C Patent Policy https://www.w3.org/Consortium/Patent-Policy/
- Only people and companies listed at https://www.w3.org/2004/01/pp-impl/47318/status are allowed to make substantive contributions to the WebRTC specs

Welcome!

- Welcome to the July 2023 interim meeting of the W3C WebRTC WG, at which we will cover:
  - Extended Use Cases, setMetadata(), IceController, Encoded Transform, device-id in permissions
- Future meetings:
  - September 19
  - October 17
  - November 21
  - December 12

About this Virtual Meeting

- Meeting info:
  - https://www.w3.org/2011/04/webrtc/wiki/July_18_2023
- Link to latest drafts:
  - https://w3c.github.io/mediacapture-main/
  - https://w3c.github.io/mediacapture-extensions/
  - https://w3c.github.io/mediacapture-image/
  - https://w3c.github.io/mediacapture-output/
  - https://w3c.github.io/mediacapture-screen-share/
  - https://w3c.github.io/mediacapture-record/
  - https://w3c.github.io/webrtc-pc/
  - https://w3c.github.io/webrtc-extensions/
  - https://w3c.github.io/webrtc-stats/
  - https://w3c.github.io/mst-content-hint/
  - https://w3c.github.io/webrtc-priority/
  - https://w3c.github.io/webrtc-nv-use-cases/
  - https://github.com/w3c/webrtc-encoded-transform
  - https://github.com/w3c/mediacapture-transform
  - https://github.com/w3c/webrtc-svc
  - https://github.com/w3c/webrtc-ice
- Link to Slides has been published on WG wiki
- Scribe? IRC http://irc.w3.org/ Channel: #webrtc
- The meeting is (still) being recorded. The recording will be public.
- Volunteers for note taking?

W3C Code of Conduct

- This meeting operates under W3C Code of Ethics and Professional Conduct

- We're all passionate about improving WebRTC and the Web, but let's all keep the conversations cordial and professional

Virtual Interim Meeting Tips

This session is (still) being recorded

- **Click Raise hand to get into the speaker queue.**
- **Click Lower hand to get out of the speaker queue.**
- **Please wait for microphone access to be granted before speaking.**
- **If you jump the speaker queue, you will be muted.**
- **Please use headphones when speaking to avoid echo.**
- **Please state your full name before speaking.**
- **Poll mechanism may be used to gauge the “sense of the room”.**

Understanding Document Status

- Hosting within the W3C repo does ***not*** imply adoption by the WG.
  - WG adoption requires a Call for Adoption (CfA) on the mailing list.
- Editor’s drafts do ***not*** represent WG consensus.
  - WG drafts ***do*** imply consensus, once they’re confirmed by a Call for Consensus (CfC) on the mailing list.
  - Possible to merge PRs that may lack consensus, if a note is attached indicating controversy.

Issues for Discussion Today

- 08:10 - 08:45 AM WebRTC Extended Use Cases (Bernard, Sun & Tim)
- 08:45 - 09:05 AM setMetadata (Palak Agarwal)
- 09:05 - 09:20 AM Encoded Transform (Youenn)
- 09:20 - 09:35 AM Ice Controller API (Sameer Vijakar)
- 09:35 - 09:50 AM deviceId in permissions.query (Jan-Ivar)
- 09:50 - 10:00 AM Wrapup and Next Steps (Chairs)

Time control:

- A warning will be given 2 minutes before time is up.
- Once time has elapsed we will move on to the next item.

WebRTC Extended Use Cases (Bernard, Sun & Tim)

Start Time: 08:10 AM

End Time: 08:45 AM

Proposals from the May Meeting

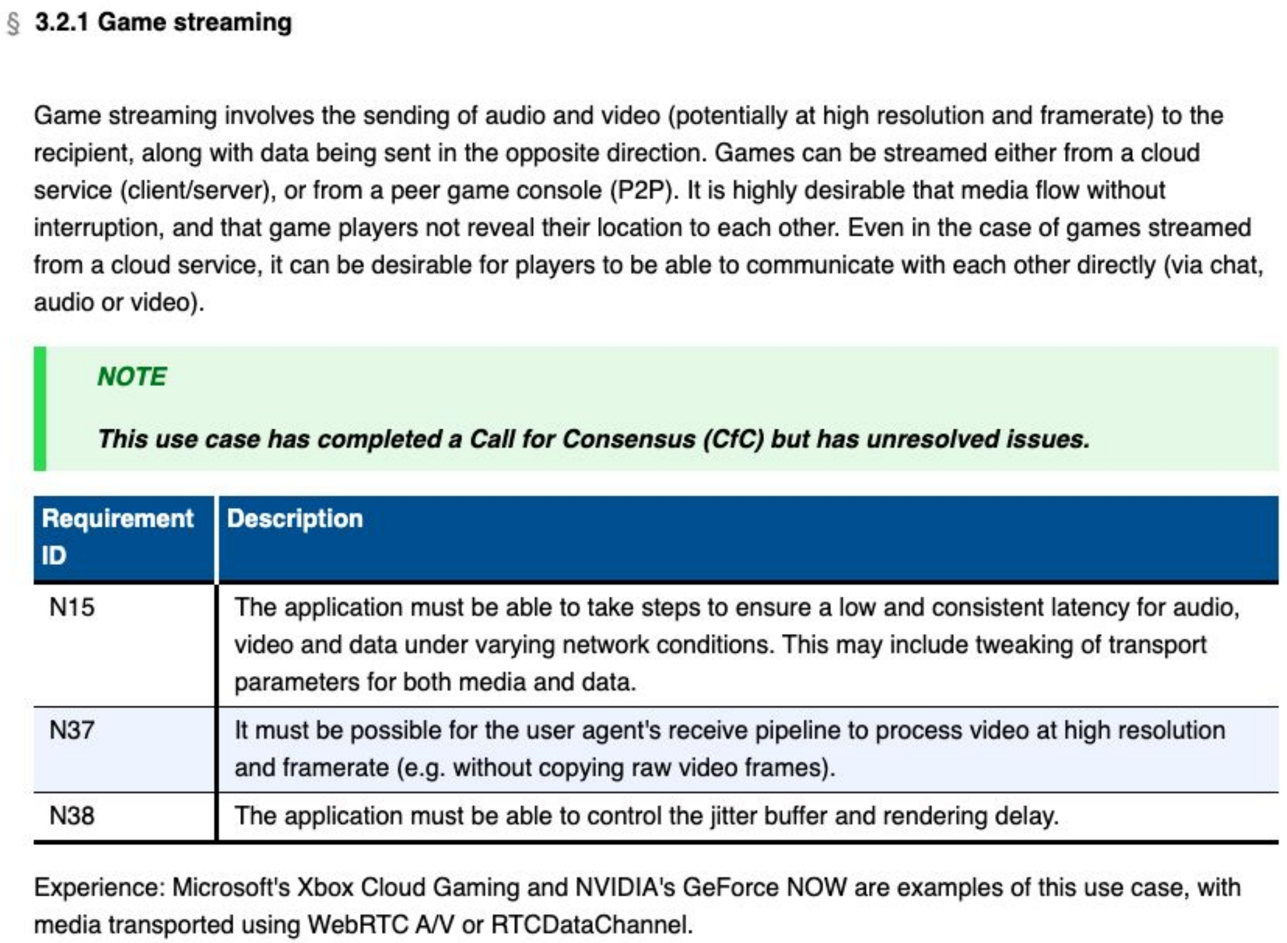
- Rename it. Proposal: “WebRTC Extended Use Cases”. Done.

- Focus on things that can only/best be done by WebRTC (p2p etc)
- Remove use cases that are now met by other standards
- Include use cases that have no requirements but extend RFC 7478
- Remove use cases that don’t get consensus within a few months
- **Remove requirements that don’t get consensus within a few months**
- Remove use cases that don’t add new requirements
- Proposed API changes should include changes to the use-case doc
- Define the relationship between this doc and explainers
- Broaden the input somehow - perhaps via webrtc.nu ?

Status of Section 3.2: Low Latency Streaming

- CFC concluded on January 16, 2023: Summary
  - 6 responses received, 5 in support, 1 no opinion
- Section 3.2: Low Latency Streaming
  - Open Issue: 103 (relates to requirements N37, N38, N39)
  - Section 3.2.1: Game Streaming
    - Open Issue: 94 (game pad input)
  - Section 3.2.2: Low Latency Broadcast with Fanout

Section 3.2.1: Game Streaming



§ **3.2.1 Game streaming**

Game streaming involves the sending of audio and video (potentially at high resolution and framerate) to the recipient, along with data being sent in the opposite direction. Games can be streamed either from a cloud service (client/server), or from a peer game console (P2P). It is highly desirable that media flow without interruption, and that game players not reveal their location to each other. Even in the case of games streamed from a cloud service, it can be desirable for players to be able to communicate with each other directly (via chat, audio or video).

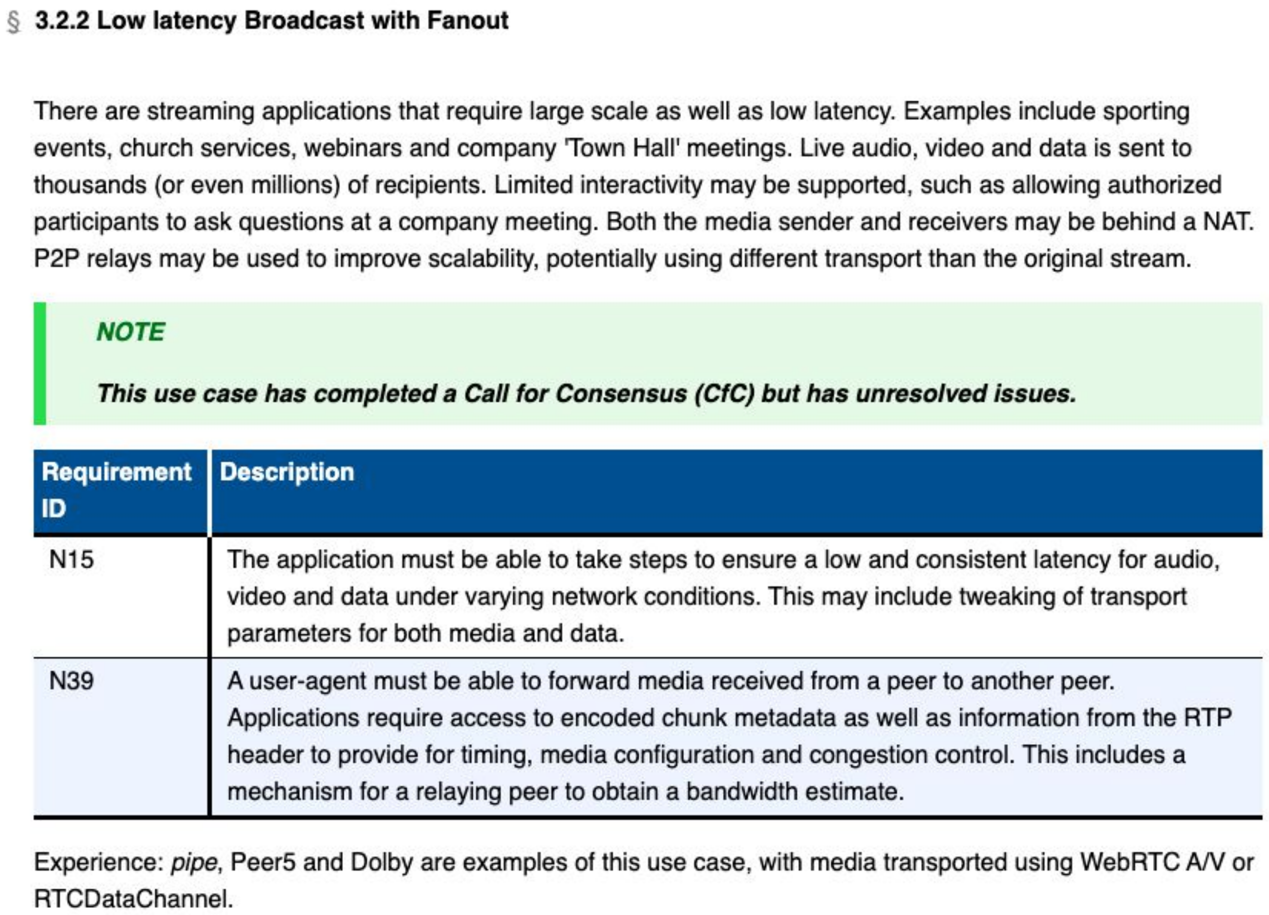
NOTE

This use case has completed a Call for Consensus (CfC) but has unresolved issues.

Requirement ID	Description
N15	The application must be able to take steps to ensure a low and consistent latency for audio, video and data under varying network conditions. This may include tweaking of transport parameters for both media and data.
N37	It must be possible for the user agent's receive pipeline to process video at high resolution and framerate (e.g. without copying raw video frames).
N38	The application must be able to control the jitter buffer and rendering delay.

Experience: Microsoft's Xbox Cloud Gaming and NVIDIA's GeForce NOW are examples of this use case, with media transported using WebRTC A/V or RTCDataChannel.

Section 3.2.2: Low latency broadcast w/fanout



§ **3.2.2 Low latency Broadcast with Fanout**

There are streaming applications that require large scale as well as low latency. Examples include sporting events, church services, webinars and company 'Town Hall' meetings. Live audio, video and data is sent to thousands (or even millions) of recipients. Limited interactivity may be supported, such as allowing authorized participants to ask questions at a company meeting. Both the media sender and receivers may be behind a NAT. P2P relays may be used to improve scalability, potentially using different transport than the original stream.

NOTE

This use case has completed a Call for Consensus (CfC) but has unresolved issues.

Requirement ID	Description
N15	The application must be able to take steps to ensure a low and consistent latency for audio, video and data under varying network conditions. This may include tweaking of transport parameters for both media and data.
N39	A user-agent must be able to forward media received from a peer to another peer. Applications require access to encoded chunk metadata as well as information from the RTP header to provide for timing, media configuration and congestion control. This includes a mechanism for a relaying peer to obtain a bandwidth estimate.

Experience: *pipe*, Peer5 and Dolby are examples of this use case, with media transported using WebRTC A/V or RTCDataChannel.

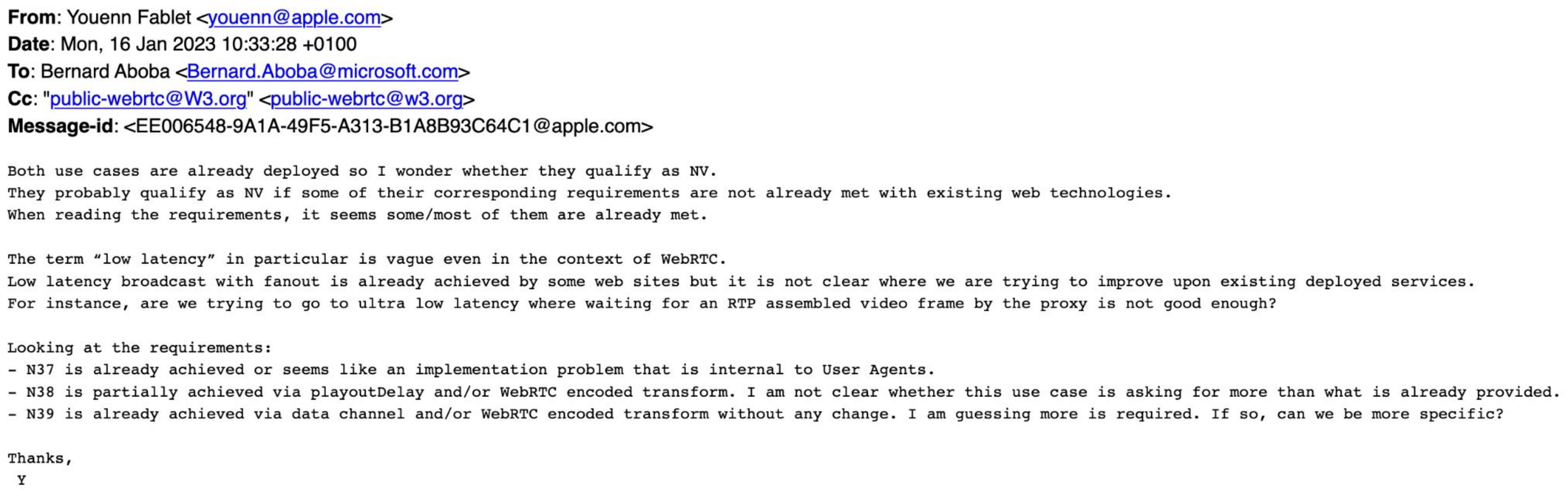
For Discussion Today

- Issues
  - Issue 94: Section 3.2.1: Improvements for Game pad input
  - Issue 103: Section 3.2: Feedback related to WebRTC-NV Low Latency Streaming Use Case
- PRs
  - PR 116: Remove specific hardware requirement (GPU) (Bernard)
  - PR 117: Section 3.2.1: Update Requirement N37 (Bernard)
  - PR 120: Section 3.2.2: Add Requirements N13 and N16 (Bernard)
  - PR 121: Section 3.2.1: Clarify meaning of “low latency” (Bernard)
  - PR 119: Section 3.2: Streaming should not require user prompts without not sending media. (Sun)
  - PR 118: Section 3.2.1: Clarify game streaming requirements (Sun & Bhavani)

Issue 94: Improvements for game pad input (Section 3.2.1)

- Existing Issues filed and PRs submitted relating to game pad input:
  - Issue 4: Should fire events instead of using passive model
  - PR 152: Add gamepad input events
- Status of incubation/development efforts:
  - Chrome status: Gamepad button and axis events
  - Documentation (03 August 2021)
  - Chromium tracking issue (last entry 20 August 2021): https://bugs.chromium.org/p/chromium/issues/detail?id=856290
- Questions
  - Are implementation efforts stalled? Any chance of revival?
  - Any relationship to work in the scope of the WEBRTC WG?
  - Is there a reason to add a requirement to Section 3.2.1?
  - Is there a useful editorial change to make (e.g. link to Issues/PRs)?
  - Or should we close the issue?

Issue #103: Feedback related to WebRTC-NV Low Latency Streaming Use Case



From: Youenn Fablet <youenn@apple.com>
Date: Mon, 16 Jan 2023 10:33:28 +0100
To: Bernard Aboba <Bernard.Aboba@microsoft.com>
Cc: "public-webrtc@W3.org" <public-webrtc@w3.org>
Message-id: <EE006548-9A1A-49F5-A313-B1A8B93C64C1@apple.com>

```
Both use cases are already deployed so I wonder whether they qualify as NV.
They probably qualify as NV if some of their corresponding requirements are not already met with existing web technologies.
When reading the requirements, it seems some/most of them are already met.

The term “low latency” in particular is vague even in the context of WebRTC.
Low latency broadcast with fanout is already achieved by some web sites but it is not clear where we are trying to improve upon existing deployed services.
For instance, are we trying to go to ultra low latency where waiting for an RTP assembled video frame by the proxy is not good enough?

Looking at the requirements:
- N37 is already achieved or seems like an implementation problem that is internal to User Agents.
- N38 is partially achieved via playoutDelay and/or WebRTC encoded transform. I am not clear whether this use case is asking for more than what is already provided.
- N39 is already achieved via data channel and/or WebRTC encoded transform without any change. I am guessing more is required. If so, can we be more specific?

Thanks,
 Y
```

Issue #103: Statement of Principles

- Requirements should be specific and actionable
  - Specific: It should be possible to determine whether the requirement is met
    - Example: “Low latency” should be defined.
  - Actionable: It should be possible to develop API proposals to address the requirements.
    - Example: Does requirement N39 require new APIs?
    - Performance requirements may not be if they can be met by software implementation improvements or better hardware, without API changes.
- It should be possible to determine whether a requirement is met
  - Required by the W3C process!
  - Is the requirement enabled by a specification or a proposed PR?
    - Example: Requirement N38 is met by jitterBufferTarget. How do we indicate that?
  - Has an Issue been filed, which if resolved, would enable the requirement to be met?
    - Example: Issue 146: Exposing decode errors / SW fallback as an event
  - Is the specification or PR testable? Is there a test?

What is the relationship of a Use Case To…

- Issues (what specific problems are being referred to)
    - Should a use case link to related Issues?
        - Example: Should Requirement N37 (Ability to support high resolution/framerate) link to related issues (e.g. hardware acceleration)?
        - Advantage: Links requirements to specific Issues whose resolution can be tracked
- API Proposals (what API proposals relate to the problems)
    - Should a use case link to API proposals?
        - Clarifies whether a use case has API proposals
        - Links use case to an API proposal whose progress can be tracked
- Explainers (How the proposals relate to the use cases)
    - Should explainers link to use cases?
        - Clarifies whether an API proposal is solving a use case

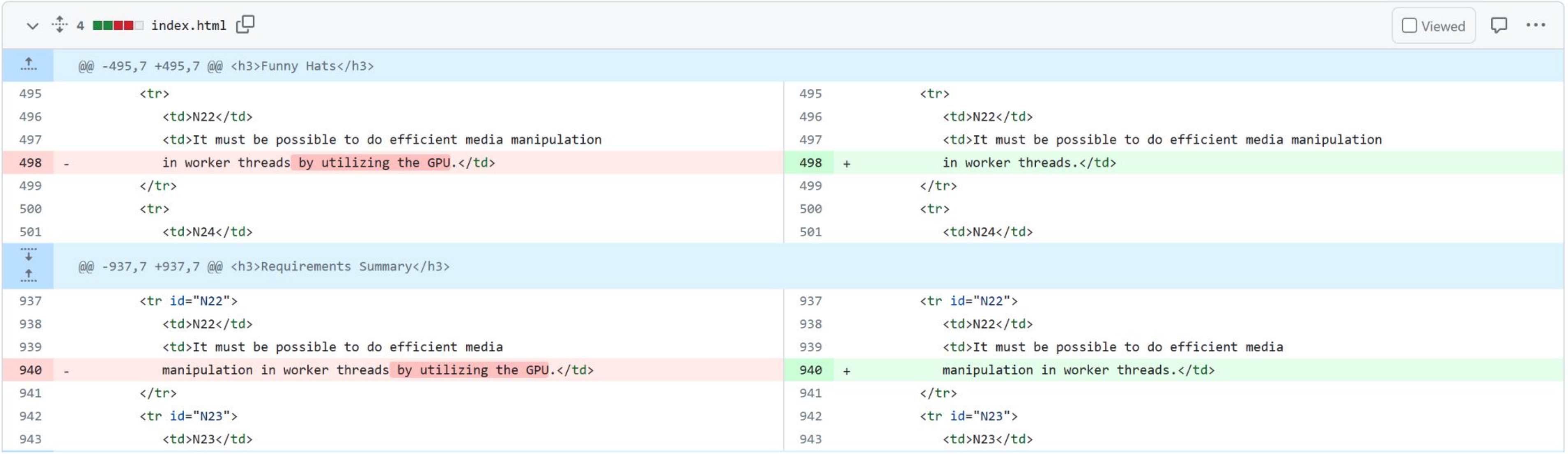
What do we do with aspirations?

- NV was a place to put hopes and dreams
- Many of them have been realized
- (although not always in the way that NV described)

Where should we put hopes and dreams ?
Are they out of scope for a standards group?

PR 116: Remove specific hardware reference (GPU)

- Rationale
  - Efficient media manipulation may involve multiple mechanisms, including hardware acceleration (GPU, NPU, etc.), WASM SIMD, conversions without copies, etc.
  - Focus on GPU is too narrow.

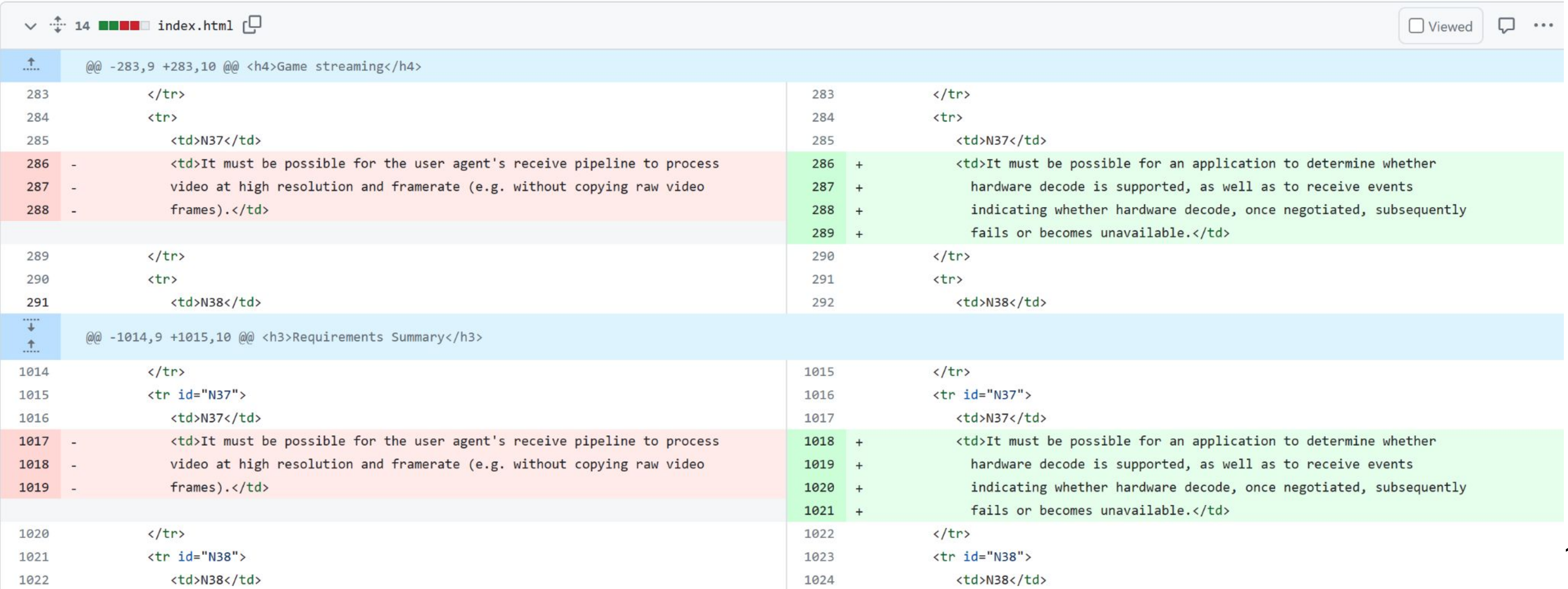


index.html

```diff
@@ -495,7 +495,7 @@ <h3>Funny Hats</h3>
         <tr>
           <td>N22</td>
           <td>It must be possible to do efficient media manipulation
-          in worker threads by utilizing the GPU.</td>
+          in worker threads.</td>
         </tr>
         <tr>
           <td>N24</td>
@@ -937,7 +937,7 @@ <h3>Requirements Summary</h3>
         <tr id="N22">
           <td>N22</td>
           <td>It must be possible to do efficient media
-          manipulation in worker threads by utilizing the GPU.</td>
+          manipulation in worker threads.</td>
         </tr>
         <tr id="N23">
           <td>N23</td>
```

PR 117: Update Requirement N37

- Rationale
  - Existing requirement N37 is not actionable.
  - Need to identify the Issue and required API changes. See:
    - Issue 146: Exposing decode errors / SW fallback as an event



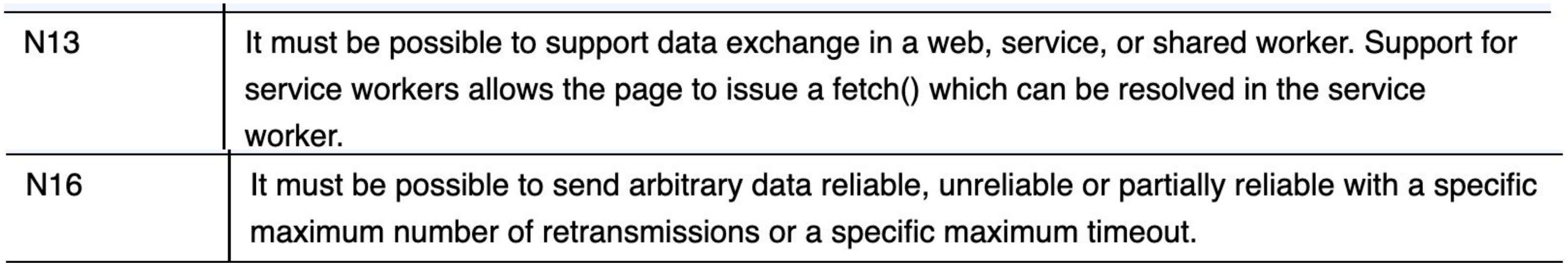
14 index.html

Viewed

```
@@ -283,9 +283,10 @@ <h4>Game streaming</h4>
         </tr>
         <tr>
            <td>N37</td>
-           <td>It must be possible for the user agent's receive pipeline to process
-           video at high resolution and framerate (e.g. without copying raw video
-           frames).</td>
+           <td>It must be possible for an application to determine whether
+             hardware decode is supported, as well as to receive events
+             indicating whether hardware decode, once negotiated, subsequently
+             fails or becomes unavailable.</td>
         </tr>
         <tr>
            <td>N38</td>
@@ -1014,9 +1015,10 @@ <h3>Requirements Summary</h3>
         </tr>
         <tr id="N37">
            <td>N37</td>
-           <td>It must be possible for the user agent's receive pipeline to process
-           video at high resolution and framerate (e.g. without copying raw video
-           frames).</td>
+           <td>It must be possible for an application to determine whether
+             hardware decode is supported, as well as to receive events
+             indicating whether hardware decode, once negotiated, subsequently
+             fails or becomes unavailable.</td>
         </tr>
         <tr id="N38">
            <td>N38</td>
```

PR 120: Section 3.2.2: Add Requirements N13 and N16 (Bernard)

- Rationale
  - Fanout implementations currently using data channel on the main thread have encountered latency issues. Solutions:
    - Support for data channel in workers
    - Support for partial reliability.

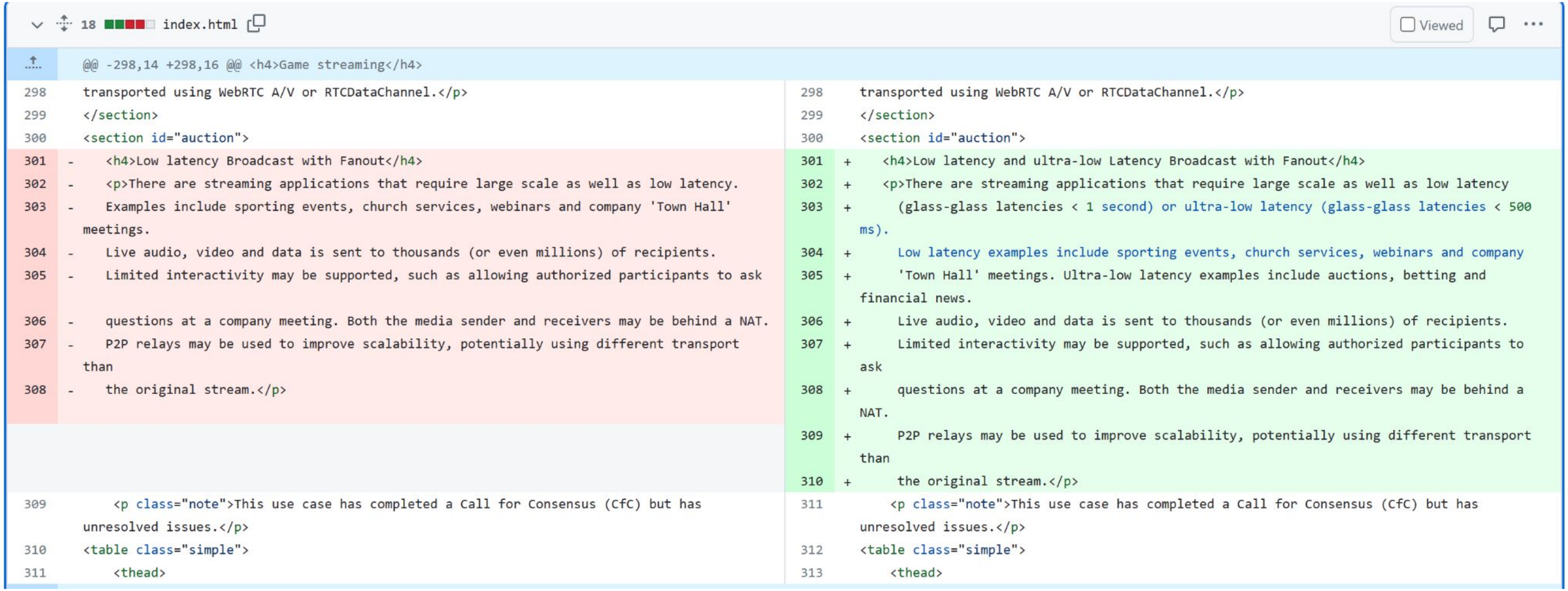


N13	It must be possible to support data exchange in a web, service, or shared worker. Support for service workers allows the page to issue a fetch() which can be resolved in the service worker.
N16	It must be possible to send arbitrary data reliable, unreliable or partially reliable with a specific maximum number of retransmissions or a specific maximum timeout.

PR 121: Section 3.2.2: Clarify meaning of “low latency” (Bernard)

- Rationale
  - There are some uses (e.g. webinars, company meetings) that require “low latency” (glass-glass latency < 1 second).
    - Existing implementations often use data channel for fanout. These can achieve acceptable latency with requirements N13 and N16.
  - There are other uses (e.g. auctions, betting) where “ultra low latency” (glass-glass latency < 500ms) is required.
    - WebRTC is popular for these “ultra low latency” uses (e.g. WHIP/WHEP)
    - For these uses, data channel fanout may add too much latency even with N13 and N16.
      - Requirement N39 covers RTP fanout (which uses low latency congestion control such as gcc, rather than the NewReno in data channel)
      - Requirement N43 proposed (see next presentation)

PR 121: Section 3.2.2: Clarify meaning of “low latency” (cont’d)



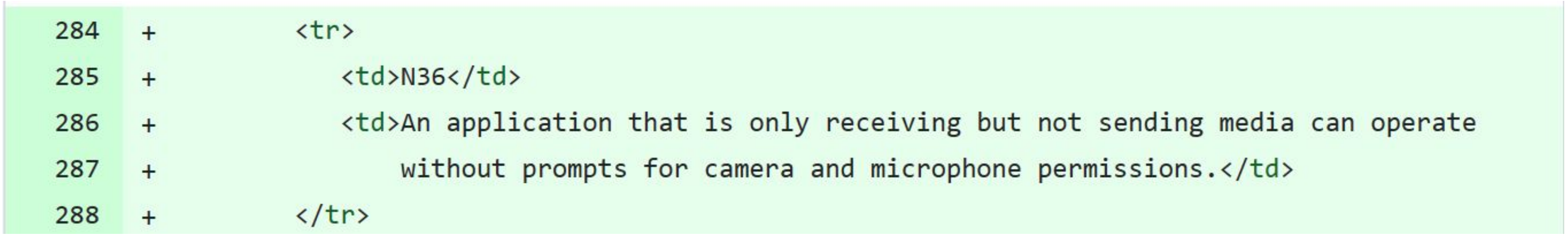
```
@@ -298,14 +298,16 @@ <h4>Game streaming</h4>
```

```
  transported using WebRTC A/V or RTCDataChannel.</p>
  </section>
  <section id="auction">
-    <h4>Low latency Broadcast with Fanout</h4>
-    <p>There are streaming applications that require large scale as well as low latency.
-    Examples include sporting events, church services, webinars and company 'Town Hall' meetings.
-    Live audio, video and data is sent to thousands (or even millions) of recipients.
-    Limited interactivity may be supported, such as allowing authorized participants to ask
-    questions at a company meeting. Both the media sender and receivers may be behind a NAT.
-    P2P relays may be used to improve scalability, potentially using different transport than
-    the original stream.</p>
      <p class="note">This use case has completed a Call for Consensus (CfC) but has unresolved issues.</p>
  <table class="simple">
      <thead>
```

```
  transported using WebRTC A/V or RTCDataChannel.</p>
  </section>
  <section id="auction">
+    <h4>Low latency and ultra-low Latency Broadcast with Fanout</h4>
+    <p>There are streaming applications that require large scale as well as low latency
+       (glass-glass latencies < 1 second) or ultra-low latency (glass-glass latencies < 500 ms).
+       Low latency examples include sporting events, church services, webinars and company
+       'Town Hall' meetings. Ultra-low latency examples include auctions, betting and financial news.
+       Live audio, video and data is sent to thousands (or even millions) of recipients.
+       Limited interactivity may be supported, such as allowing authorized participants to ask
+       questions at a company meeting. Both the media sender and receivers may be behind a NAT.
+       P2P relays may be used to improve scalability, potentially using different transport than
+       the original stream.</p>
      <p class="note">This use case has completed a Call for Consensus (CfC) but has unresolved issues.</p>
  <table class="simple">
      <thead>
```

PR 119: Section 3.2: Streaming should not require user prompts without sending media.

- Rationale
  - Section 3.2.1: Gamers will be surprised by (and suspicious of) permission prompts in games that do not use audio/video input (e.g. to support audio/video chat).
  - Section 3.2.2: Conventional streaming does not require audio/video input permissions. Why should low latency streaming be different?



```
+        <tr>
+          <td>N36</td>
+          <td>An application that is only receiving but not sending media can operate
+              without prompts for camera and microphone permissions.</td>
+        </tr>
```

PR 118: Clarify Game Streaming requirements (Section 3.2)

- Rationale: Cloud Game Characteristics
    - A highly interactive application that depends on **continuous visual feedback** to user inputs.
    - The cloud gaming latency KPI would track Click to Pixel latency - time elapsed between user input to when the game response is available at the user display (where as non-interactive applications may track G2G latency as the KPI).
    - Requires low and consistent latency. Desirable C2P latency range is typically 30 - 150ms. A latency higher than 170 ms makes high precision games unplayable.
    - Loss of video is highly undesirable. Garbled or corrupt video with fast recovery may be preferable in comparison to a video freeze.
    - Motion complexity can be high during active gameplay scenes.
    - Consistent latency is critical for player adaptability. Varying latency requires players to adapt continuously which can be frustrating and break gameplay.
    - The combination of high complexity, ultra low latency and fast recovery will require additional adaptive streaming and recovery techniques.

PR 118: Clarify Game Streaming requirements (Section 3.2)

ID	Requirement	Description	Benefits to Cloud Gaming	Is it Cloud Gaming Specific?
N48 (New)	Recovery using non-key frames	WebRTC must support a mode in which it allows video decoding to continue even after a frame loss without waiting for a key frame This enables addition of recovery methods such as using frames containing intra coded macroblocks and coding units.	Players can continue to game with partially intelligible video. Fast recovery from losses on the network	Can be used by any application where video corruption is preferred to video freezes
N49 (New)	Loss of encoder -decoder synchronicity notification	The WebRTC connection should generate signals indicating to encoder about loss of encoder-decoder synchronicity (DPB buffers) and sequence of the frame loss.(RFC 4585 section-6.3.3: Reference Picture Selection Indication)	Fast recovery from losses on network. Helps application to choose right recovery method in lossy network.	Can be used by any application where video corruption is preferred to video freezes
N50 (New)	Configurable RTCP transmission interval	Application must be able to configure RTCP feedback transmission interval (Ex: Transport-wide RTCP Feedback Message)	Gaming is sensitive to congestion and packet loss resulting in higher latency. Consistent RTCP feedback helps application to adapt video quality to varying network (BWE and packet loss).	Can be used by any application where latency buildup is not acceptable.
N51 (New)	Improve accuracy of Jitter buffer control	Extend adaptation of the jitter buffer to account for jitter in the pipeline upto the frame render stage	Increases accuracy of jitter buffer adaptation and helps maintain consistent latency	Helps all low latency applications, but is necessary for Cloud gaming

Discussion (End Time: 08:45)

-

setMetadata() (Palak Agarwal)

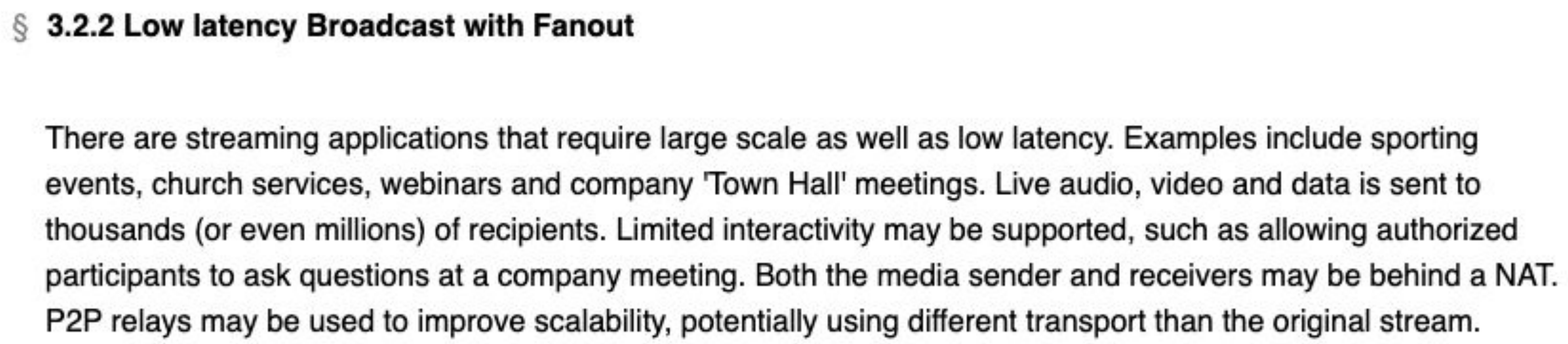
Start Time: 08:45 AM

End Time: 09:05 AM

setMetadata() for redundant relay PCs

Palak Agarwal, Google

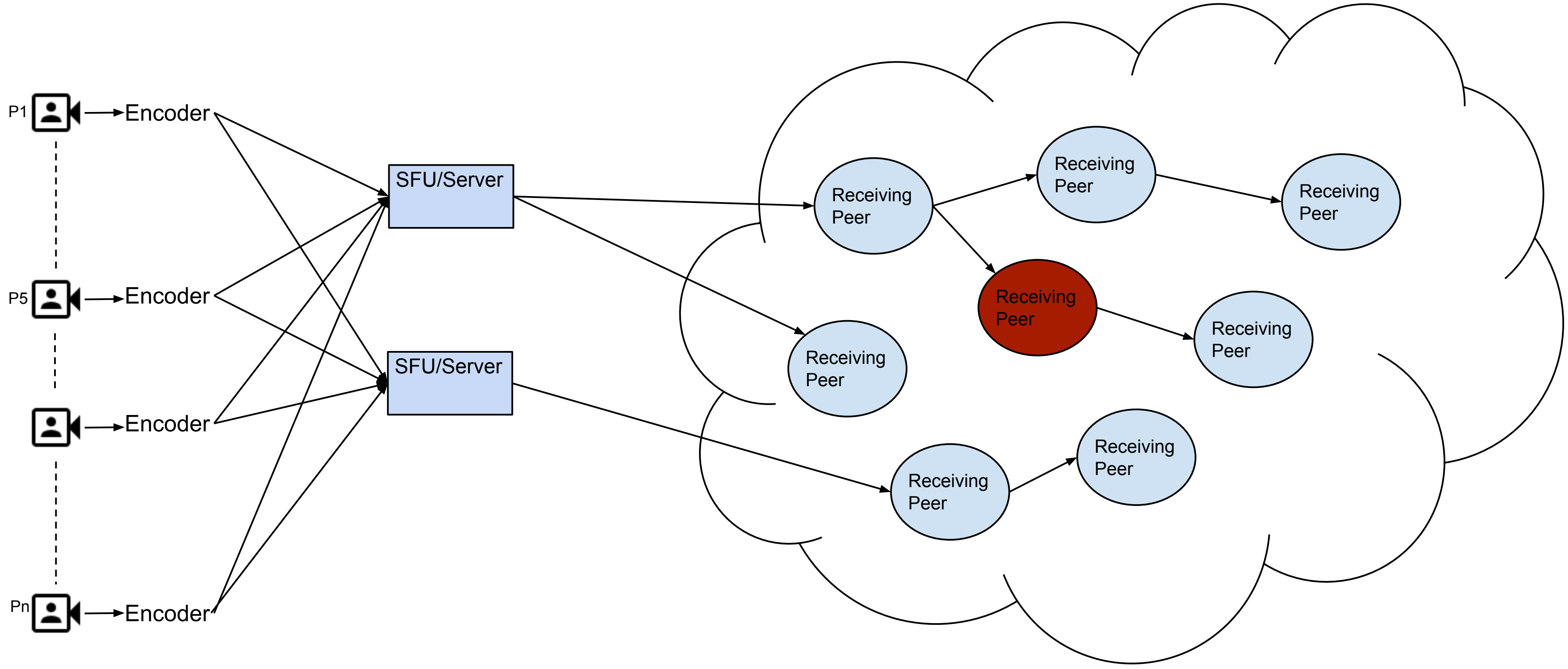
Existing use case - Section 3.2.2



§ **3.2.2 Low latency Broadcast with Fanout**

There are streaming applications that require large scale as well as low latency. Examples include sporting events, church services, webinars and company 'Town Hall' meetings. Live audio, video and data is sent to thousands (or even millions) of recipients. Limited interactivity may be supported, such as allowing authorized participants to ask questions at a company meeting. Both the media sender and receivers may be behind a NAT. P2P relays may be used to improve scalability, potentially using different transport than the original stream.

Example scenario



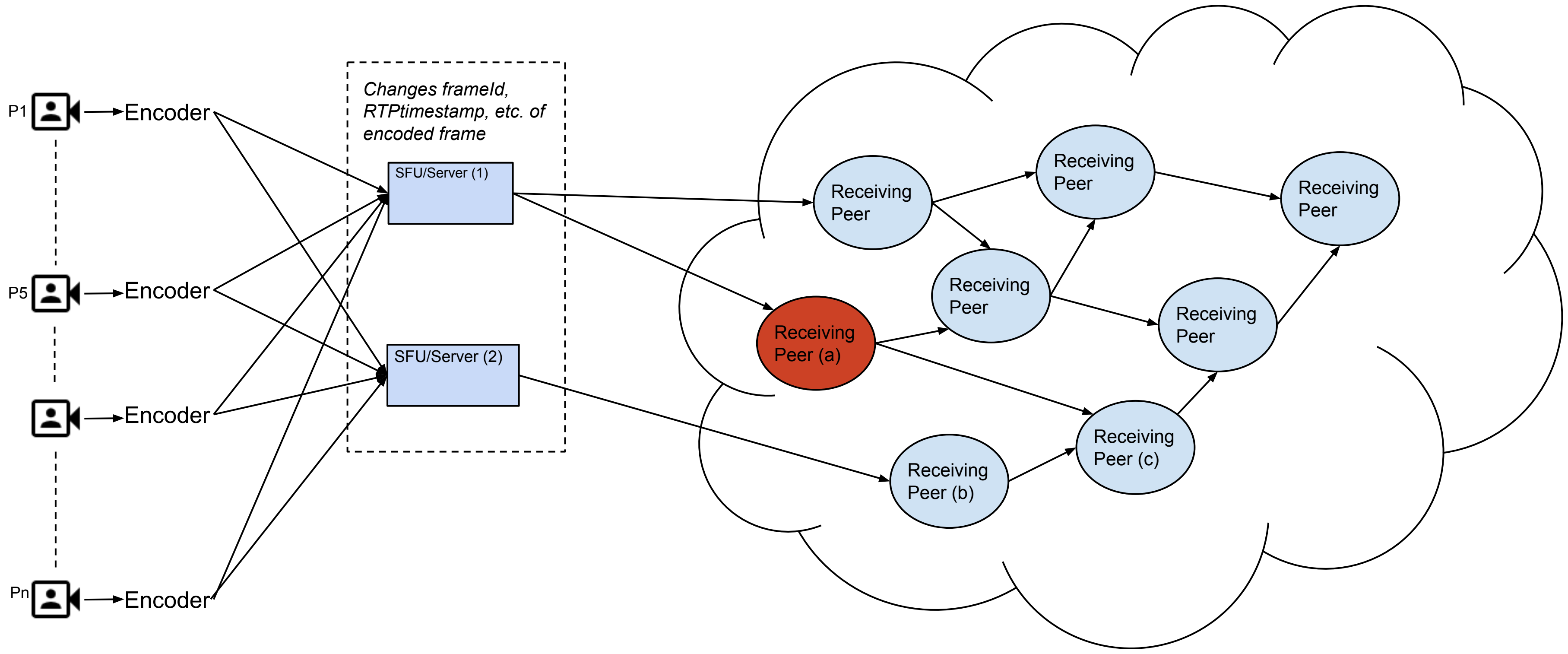
Problem: Peer fails unexpectedly

- Poses a streaming issue in Relay P2Ps. Causes disruption, non-continuous playout

- Need for handling peer failures seamlessly (cannot wait for a keyframe)

  Solution: Use redundant communication channels (i.e., multiple PCs)

Solution: Add redundant PCs

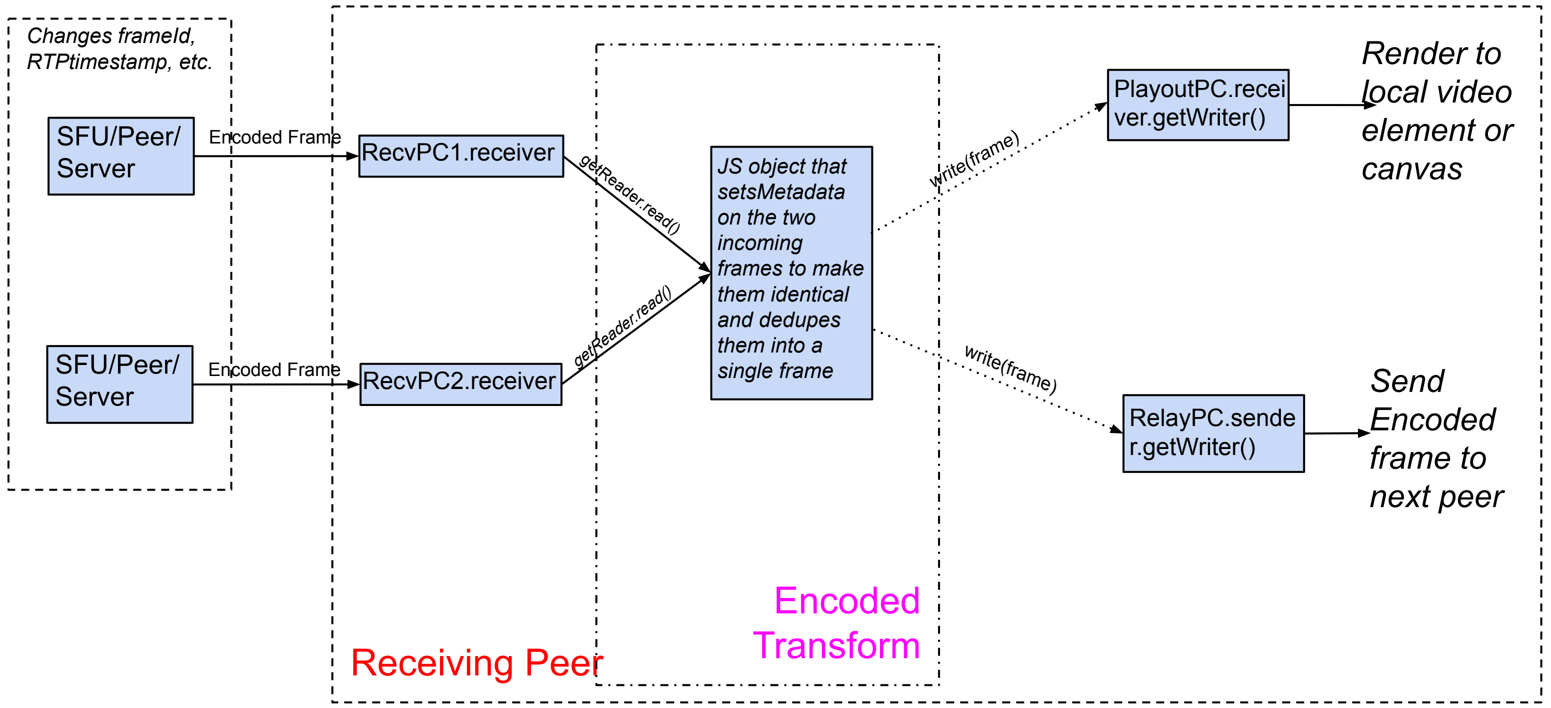


Problem: Switch between frames from PCs

- Even with redundant channels, there will be interruptions/glitches while switching to another PC as the incoming encoded frames are not identical

  Proposed solution: Update the metadata of frames from either PC such that frames with the same payload become interchangeable

Solution: setMetadata() for encoded frames



Changes frameId, RTPtimestamp, etc.
SFU/Peer/ Server
Encoded Frame
SFU/Peer/ Server
Encoded Frame
RecvPC1.receiver
RecvPC2.receiver
getReader.read()
getReader.read()
JS object that setsMetadata on the two incoming frames to make them identical and dedupes them into a single frame
write(frame)
write(frame)
PlayoutPC.receiver.getWriter()
Render to local video element or canvas
RelayPC.sender.getWriter()
Send Encoded frame to next peer
Encoded Transform
Receiving Peer

Sample code

```
// Let recvPc1, recvPc2 be the receiving PCs.
recvPc{1|2}.ontrack = evt => {
  transferFrames(evt.receiver.createEncodedStreams().readable.getReader());
};

// Let relayPc be the PC used to relay frames to the next peer.
relayPcWriter = relayPc.sender.createEncodedStreams().writable.getWriter();

async function transferFrames(reader) {
    while (true) {
        const {frame, done} = await reader.read();
        if (done) return;

        frame.timestamp = getUnifiedTimestamp(frame);
        frame.setMetadata(getUnifiedMetadata(frame));
        if(!isDuplicate(frame)) {
            relayPcWriter.write(frame);
        }
    }
}
```

Proposed requirement change

Requirement ID	Description
N15	The application must be able to take steps to ensure a low and consistent latency for audio, video and data under varying network conditions. This may include tweaking of transport parameters for both media and data.
N39	A user-agent must be able to forward media received from a peer to another peer. Applications require access to encoded chunk metadata as well as information from the RTP header to provide for timing, media configuration and congestion control. This includes a mechanism for a relaying peer to obtain a bandwidth estimate.
N43	The application can modify metadata on outgoing frames so that they fit smoothly within the expected sequence of timestamps and sequence numbers.

Proposed API change

```
interface RTCEncodedVideoFrame {
    readonly attribute RTCEncodedVideoFrameType type;
    ~~readonly~~ attribute unsigned long timestamp; // RTP timestamp
    attribute ArrayBuffer data;
    RTCEncodedVideoFrameMetadata getMetadata();
    void setMetadata(RTCEncodedVideoFrameMetadata metadata);
};
```

For this application, we need to change *frameId* and *dependencies*.

#162 outlines other possible metadata modifications.

Proposed API change

```
interface RTCEncodedAudioFrame {
    ~~readonly~~ attribute unsigned long timestamp; // RTP timestamp
    attribute ArrayBuffer data;
    RTCEncodedAudioFrameMetadata getMetadata();
};
```

Discussion (End Time: 09:05)

-

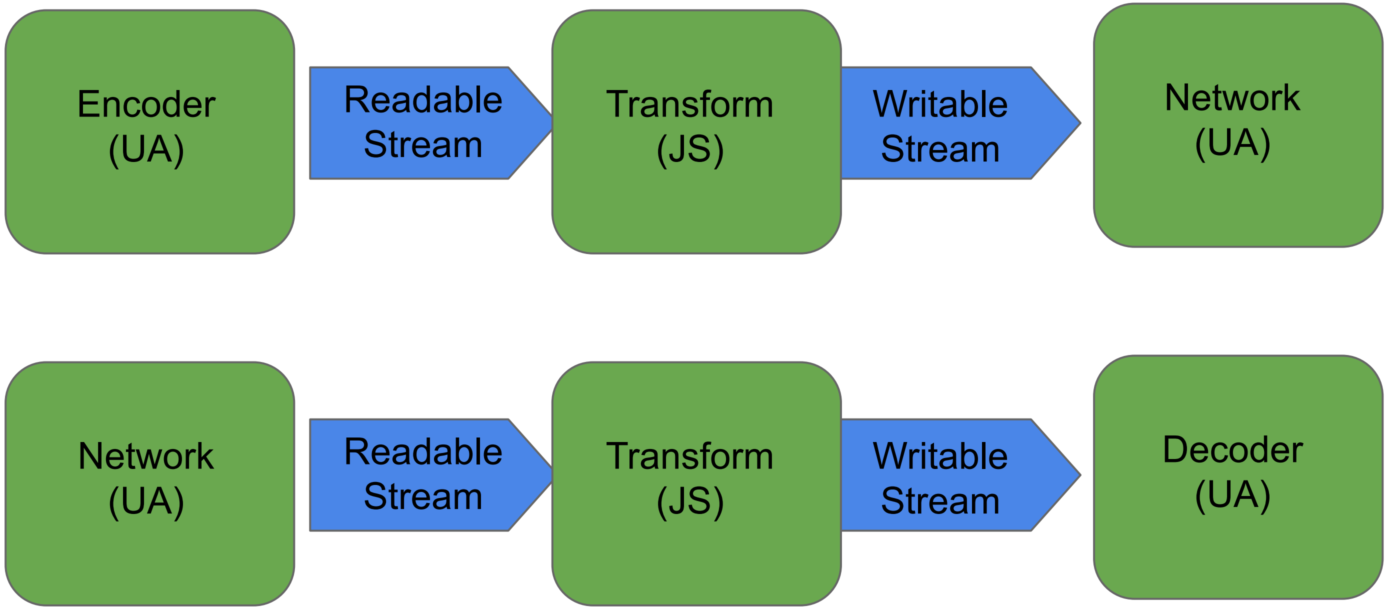
Encoded Transform (Youenn)

Start Time: 09:05 AM

End Time: 09:20 AM

Issue 188: Clarify why backpressure should be disabled

- WebRTC Encoded Transform pipeline



Issue 188: Clarify why backpressure should be disabled

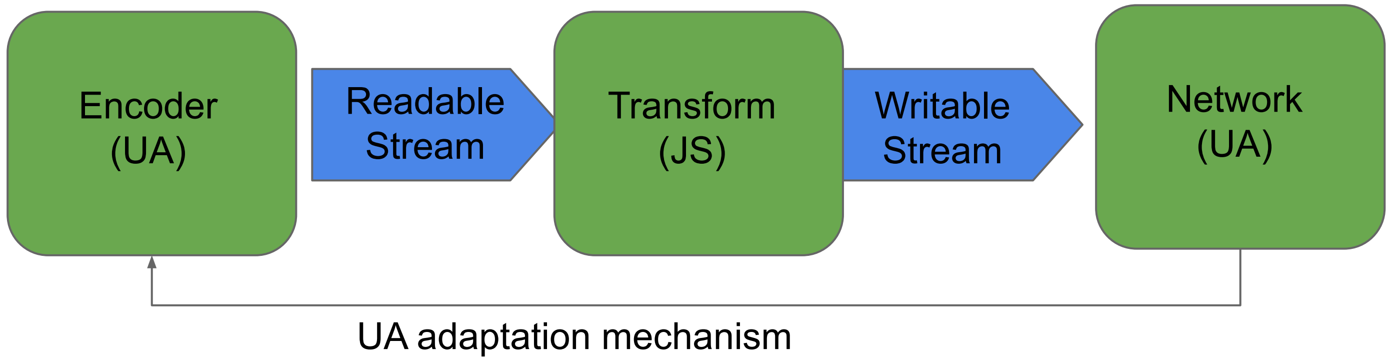
- Is backpressure observable to the JS transform?

```
onrtctransform = event => {
    process(event.transformer.readable.getReader()
          , event.transformer.writable.getWriter());
}

async function process(reader, writer)
{
    // read chunk
    const chunk = await reader.read();
    if (chunk.done)
        return;
    // write chunk
    writer.write(chunk.value);
    // wait for writer to be ready
    await writer.ready;
    // process next chunk
    process(reader, writer);
}
```

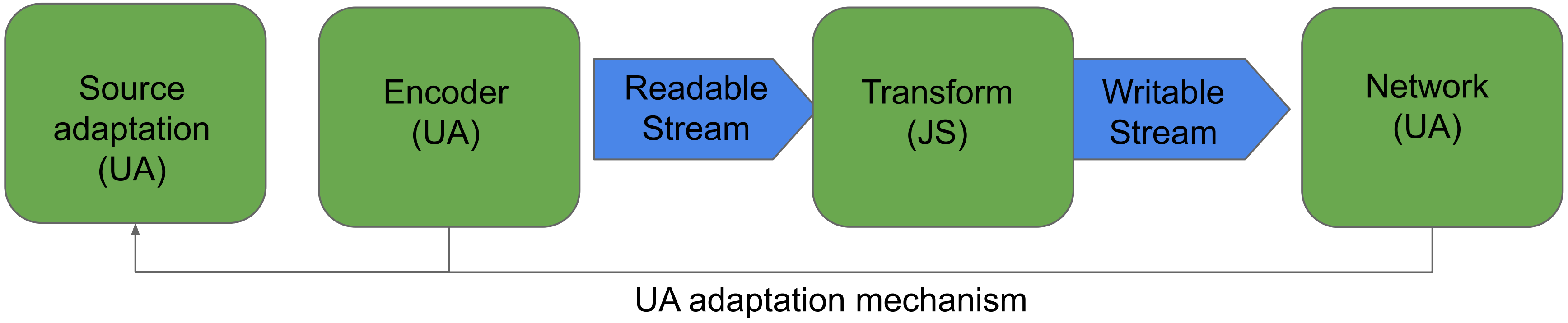
Promise resolution based on backpressure

Issue 188: Clarify why backpressure should be disabled



- What if transform is writing too much data
    - Network will not be able to sustain, packets will be dropped
    - UA will be notified of this, at some point
    - UA will instruct encoder to reduce throughput, at some point

Issue 188: Clarify why backpressure should be disabled

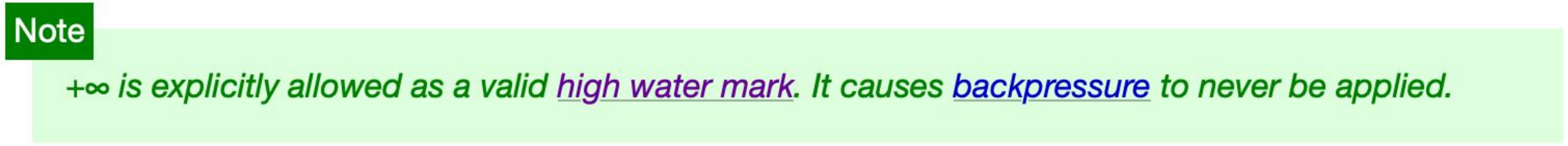


- What if transform is too slow
  - UA knows this without having to rely on streams backpressure
  - UA will reduce frame rate by dropping frames prior encoder

Issue 188: Clarify why backpressure should be disabled

- Key takeaway: backpressure is
  - Not needed here as UA knows both ends of the transform
  - A great mechanism in some contexts
    - Reliable networking like for file exchange
  - Not very useful in lossy contexts
    - Trading reliability for latency, adaptation will happen outside of backpressure

- Fortunately, WhatWG streams spec acknowledges this


Note

+∞ is explicitly allowed as a valid high water mark. It causes backpressure to never be applied.

- Proposal: update specification to use +∞ and mention rationale as a design note

Discussion (End Time: 09:20)

-

Ice Controller API (Sameer Vijaykar)

Start Time: 09:20 AM

End Time: 09:35 AM

ICE candidate pair selection and nomination - RFC 8445

- ICE RFC 8445 (and subsequent updates - RFC 8838 Trickle ICE and RFC 8863 ICE PAC) are strict wrt nomination.

- Changing the nominated candidate pair - *requires ICE restart*

  Section 8.1.1: Once the controlling agent has successfully nominated a candidate pair (Section 7.2.5.3.4), the agent MUST NOT nominate another pair for same component of the data stream within the ICE session. Doing so requires an ICE restart.

- Continue connectivity checks on non-nominated pairs - *maybe OK?*

  Section 8.3.1: Once a checklist has reached the Completed state, the agent SHOULD wait an additional three seconds, and then it can cease responding to checks or generating triggered checks on all local candidates other than the ones that became selected candidates.

- Rejecting a nomination on the controlling side - *perhaps OK*

  Section 7.3.1.5: If the controlled agent does not accept the request from the controlling agent, the controlled agent MUST reject the nomination request with an appropriate error code response.

ICE candidate pair selection and nomination - RFC 8445

- So how to change selected pair when directed by the application?
- Expensive and slow to change with an ICE restart, i.e.
  - follow same steps as an ICE restart, fire `negotiationneeded`
  - retain candidates from previous ICE session
  - nominate the application-indicated pair if checks succeed
- Alternative - selection without nomination - permitted by RFC 8445
  - Section 8.1.1: The criteria for stopping the connectivity checks and for picking a pair for nomination are outside the scope of this specification … data can always be sent on any valid pair, without nomination.
  - Controlling side simply starts sending data on a different valid pair.
  - STUN checks may be sent indefinitely for keep-alives, upper limit suggested but not mandated.
  - Prevent removal of candidates with cancelable `onicecandidatepairremove` event.
  - Prevent nomination on controlling side with cancelable `selectedcandidatepairchange` event.

Discussion (End Time: 09:35)

-

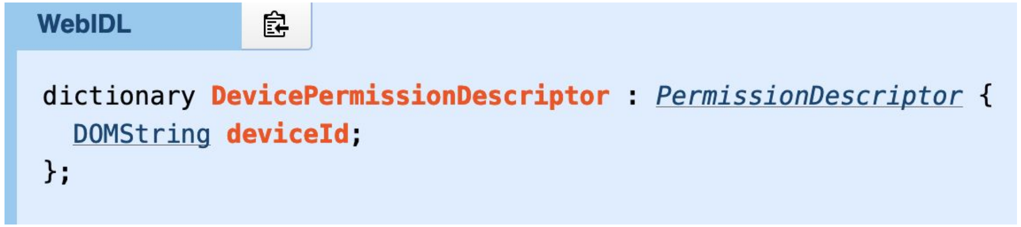
deviceId in permissions.query (Jan-Ivar)

Start Time: 09:35 AM

End Time: 09:50 AM

Issue 965: deviceId in permissions.query() is unimplemented fingerprinting surface (Jan-Ivar)

Our Permissions Integration adds `deviceId` for `"camera"`, `"microphone"` & `"speaker-selection"` permissions.


WebIDL

```
dictionary DevicePermissionDescriptor : PermissionDescriptor {
  DOMString deviceId;
};
```

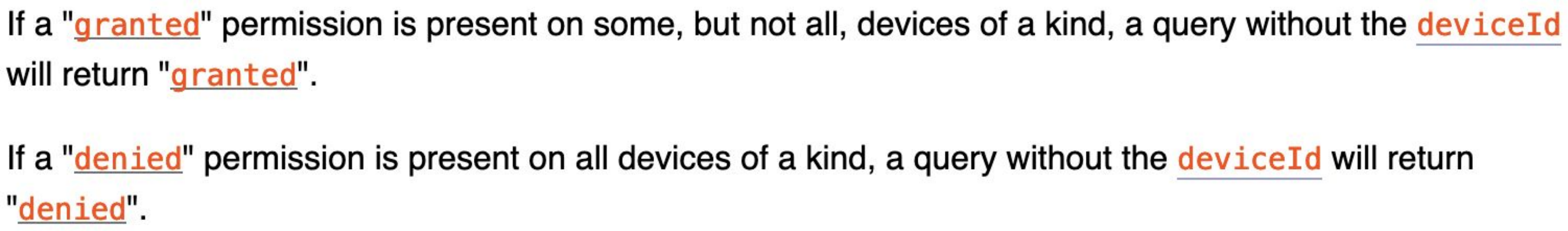
The idea was for JS to query permissions of individual media devices, which is only useful on browsers that implement per-device permissions, currently Firefox. Mozilla, however, won't be implementing the deviceId part, over fingerprinting concerns that would extend beyond those of other browsers.

There are currently zero implementations, one (manual setSinkId) WPT, and AFAIK no implementations planned.

I propose we remove this API (the `deviceId` member of `permissions.query({name, deviceId})`) from mediacapture-main and mediacapture-output.

Issue 965: deviceId in permissions.query() is unimplemented fingerprinting surface (Jan-Ivar)

1. Fingerprinting surface too much: 1 bit per device in users' systems; and defeats device exposure mitigations in enumerateDevices()
2. Existing API sufficient to negotiate consent for a camera and microphone:



If a "`granted`" permission is present on some, but not all, devices of a kind, a query without the `deviceId` will return "`granted`".

If a "`denied`" permission is present on all devices of a kind, a query without the `deviceId` will return "`denied`".

   Firefox (the only browser to maintain temporal per-device permissions) will prefer returning already granted devices instead of prompting when possible, e.g. with `getUserMedia({video: true, audio: true})`.

3. Feature at risk / No web compat

Discussion (End Time: 09:50)

-

Wrapup and Next Steps

Start Time: 09:50 AM

End Time: 10:00 AM

Thank you

Special thanks to:

WG Participants, Editors & Chairs