W3C WebRTC WG Meeting

April 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 April 2023 interim meeting of the W3C WebRTC WG, at which we will cover:
  - Grab Bag, RtpTransport, IceController, Playout-Delay
- Future meetings:
  - May 16
  - June 27
  - July 18
  - September 19
  - October 17
  - November 21
  - December 12

About this Virtual Meeting

- Meeting info:
  - https://www.w3.org/2011/04/webrtc/wiki/April_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.

To Harald, Peter and Sameer

Issues for Discussion Today

- 08:10 - 08:40 AM Grab Bag
- 08:40 - 09:10 AM RtpTransport (Peter)
- 09:10 - 09:40 AM IceController (Sameer & Peter)
- 09:40 - 09:55 AM playoutDelay (Jan-Ivar)
- 09:55 - 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.

Grab Bag

Start Time: 08:10 AM

End Time: 08:40 AM

For Discussion Today

- PR 147: Add RTCRtpEncodingParameters.codec to change the active codec (Florent)
- Stats/571: RTX and FEC stats are incomplete (Fippo)
- Issue 146: Exposing decode errors / SW fallback as an Event (Bernard)
- Issue 170: Incompatible SVC Metadata (Bernard)
- Issue 93: MediaStreamTrack audio delay/glitch capture stats (Henrik)
- PR 173: Add presentationTimeStamp to RTCEncodedVideoFrameMetaData (Tony)

PR 147: Add RTCRtpEncodingParameters.codec to change the active codec (Florent)

- As previously discussed in previous meetings PR 147 is being worked on to add support for an API to change the codec used for an encoding using `setParameters()`.

- The proposed API changes `RTCRtpEncodingParameters`, it will impact both video and audio sender

- It might not have been clear if it impacted non-simulcast usage or audio. The intent is to make it work with both. Use case: change codec without renegotiation. Any objections?

```
partial dictionary RTCRtpEncodingParameters {
    RTCRtpCodec codec;
}
```

Stats/751: RTX and FEC stats are incomplete (Fippo)

- RTX: outbound-rtp has retransmittedBytesSent
  - but no corresponding inbound-rtp value
- FEC: inbound-rtp has fecPacketsReceived
  - outbound-rtp has has corresponding counter only in provisional stats
  - no fecBytesReceived either, important e.g. for FlexFEC bitrate
- Proposal:
  - add retransmittedPacketsReceived and retransmittedBytesReceived to inbound-rtp
  - add fecPacketsSent and fecBytesSent to outbound-rtp
  - add fecBytesReceived to inbound-rtp
  - mark as "ready for PR", merge once we have at least one implementation

Issue 146: Exposing decode errors / SW fallback as an Event (Bernard)

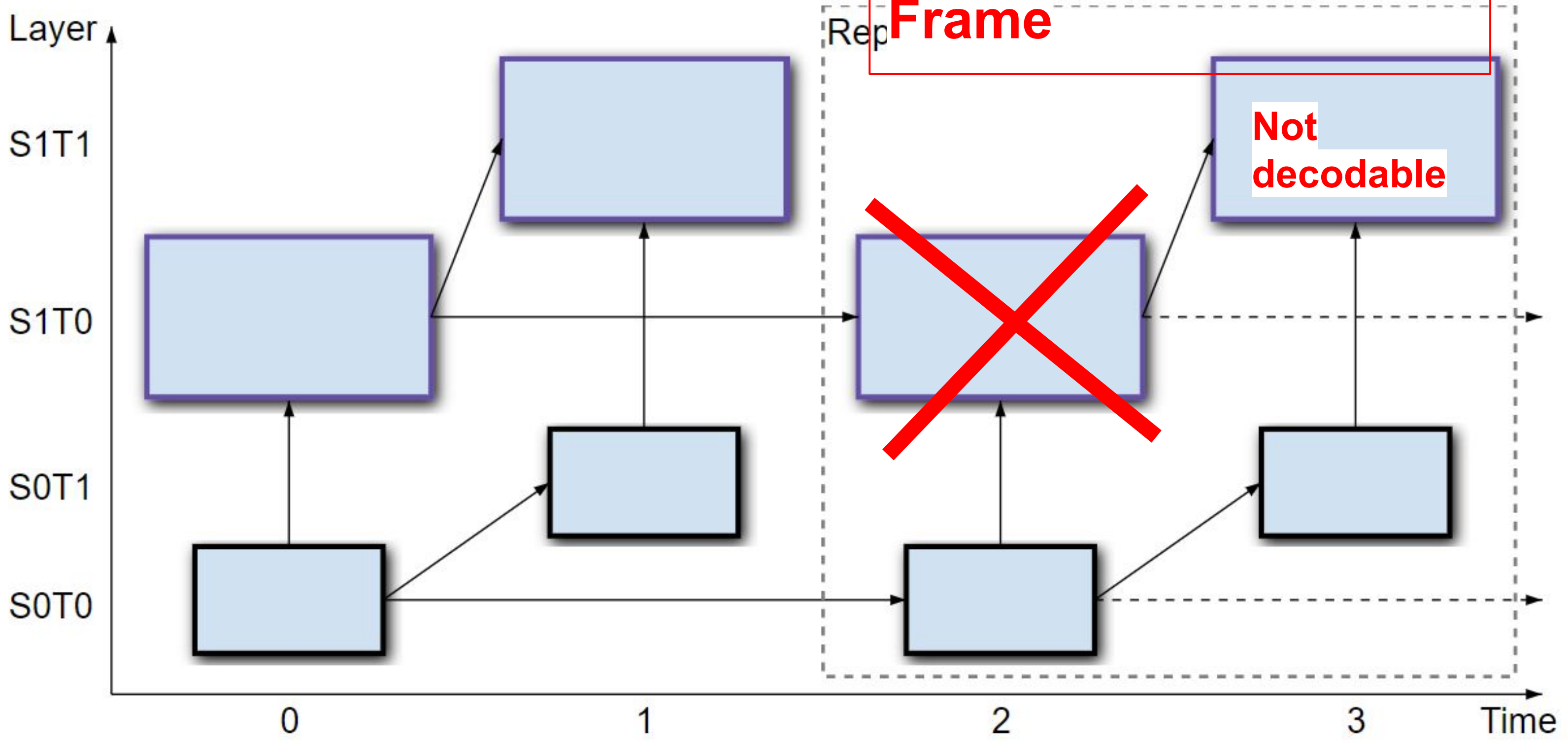
- As discussed in the MEDIA WG it would be useful to distinguish between:
  - ***Data error***: the encoded chunk fed to the decoder could not be decoded.
    - Hardware decoders are more strict than software decoders, so a software decoder might not error on the same input.
    - Application can failover to software.
      - Developer needs to capture bitstream & investigate the interop issue.
      - Nothing actionable for the user.
  - ***Resource problem***: the hardware decoder is unavailable.
    - Hardware resources might have been allocated to another application, or might be unavailable for some other reason (GPU crash).
    - Advice could be surfaced to the user (e.g. “Another application is affecting performance. Please quit other applications.”).
    - Application might be able to re-acquire resources.
- Proposal: Mark as “Ready for PR”?

Issue 170: Incompatible SVC Metadata (Bernard)

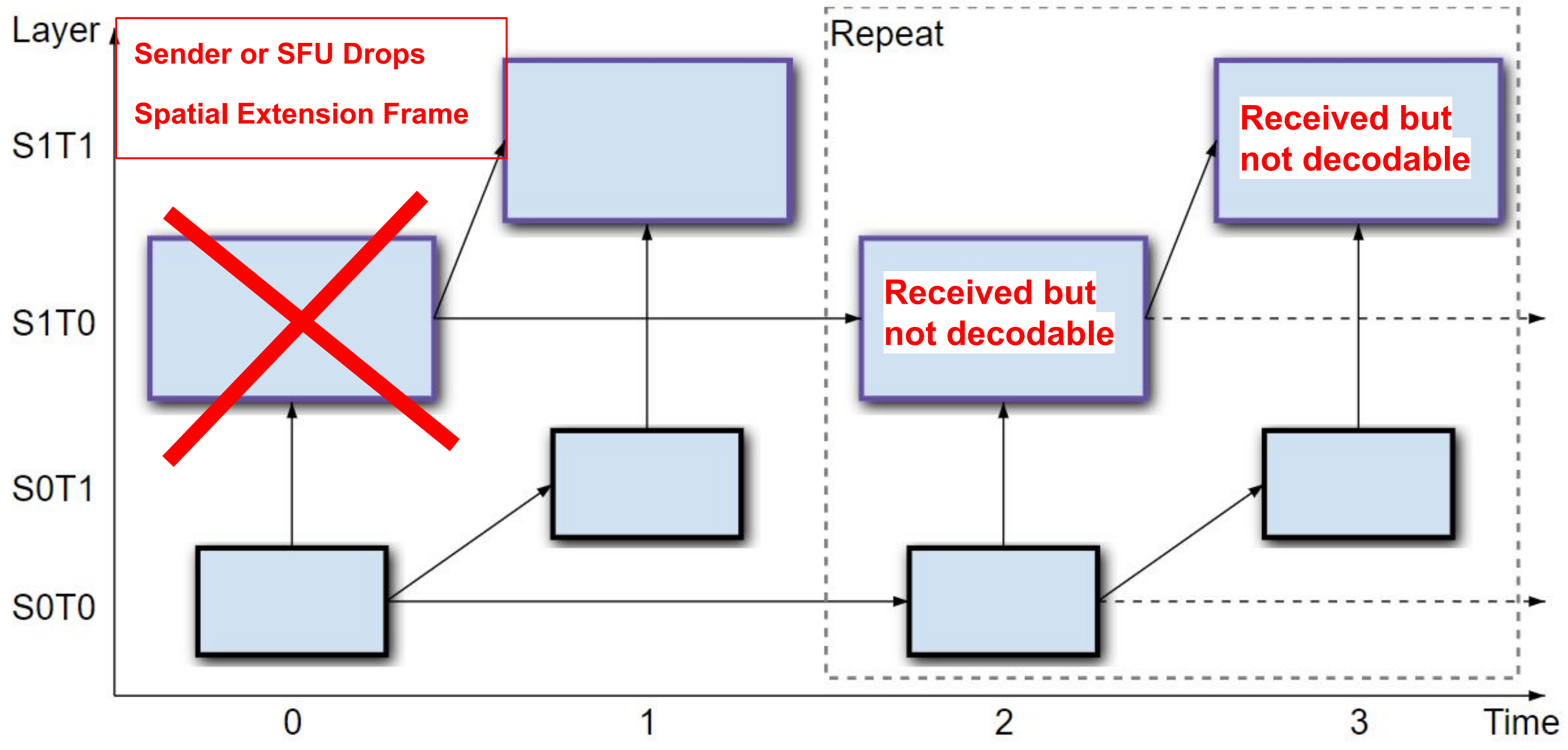
- When spatial scalability modes are selected, the sender (or SFU) can drop layers at any time.
- However, if the sender (or SFU) drops a spatial layer, how can it be added back?
  - Sender or SFU cannot just send subsequent frames at the same spatial layer.
    - Subsequent frames depend on previous frames at the same spatial layer, so they are not decodable.
    - Even a frame whose dependencies have been received may not be decodable.
      - Decodability requires an unbroken ***chain of dependencies***.
  - Receiver needs to quickly determine:
    - Is a received frame decodable?
    - Is a received frame necessary for the desired resolution/framerate (decode target)?

Issue 170: SVC Metadata (cont'd)

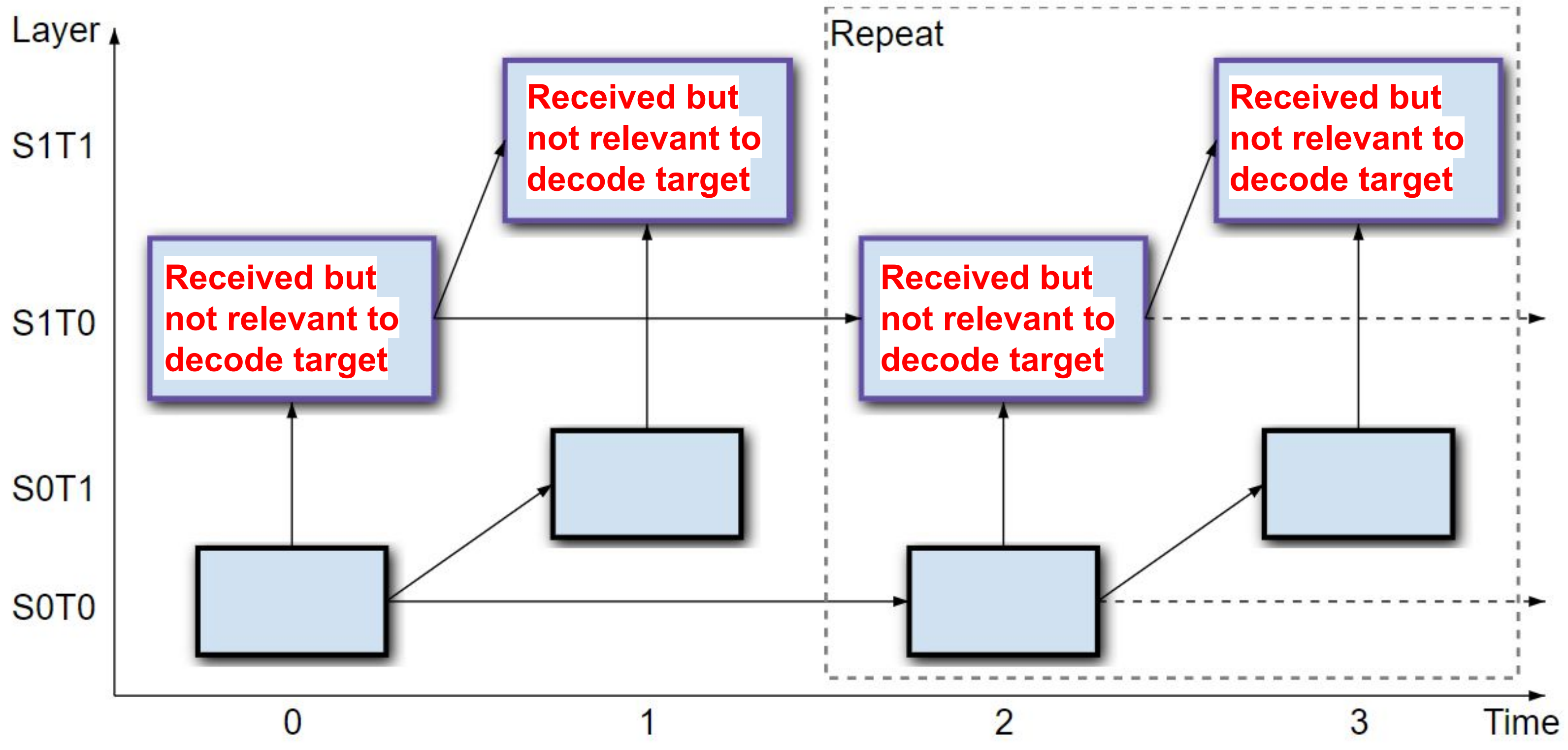
Sender or SFU Drops Spatial Extension Frame



Issue 170: SVC Metadata (cont'd)

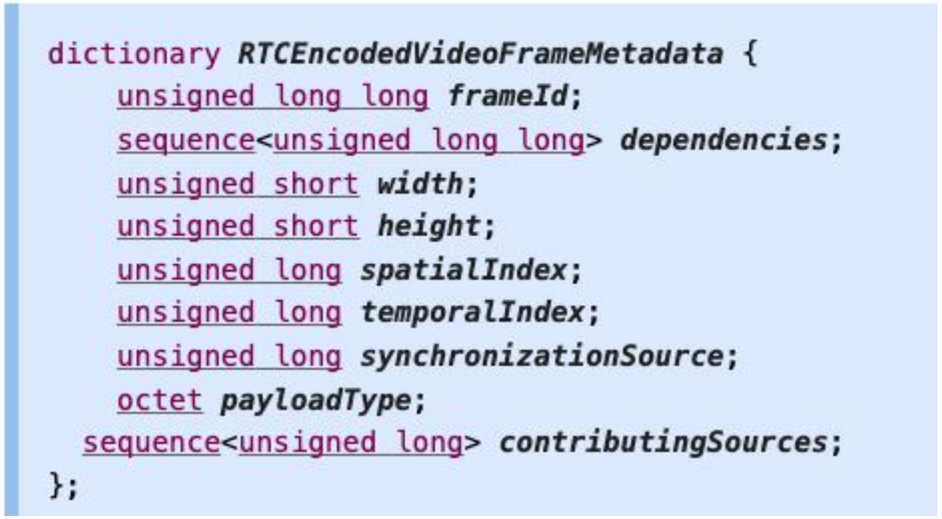


Issue 170: SVC Metadata (cont'd)

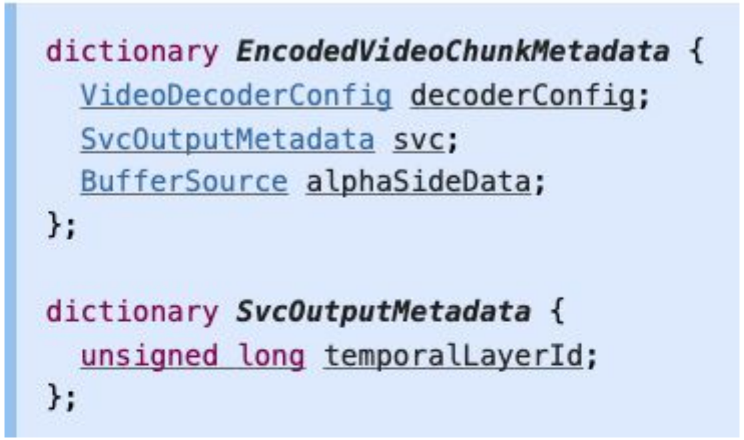

Layer
Repeat
S1T1
S1T0
S0T1
S0T0
Received but not relevant to decode target
Received but not relevant to decode target
Received but not relevant to decode target
Received but not relevant to decode target
0
1
2
3
Time

Issue 170: SVC Metadata (cont'd)

- Problem: **RTCEncodedVideoFrameMetadata** is:
  - Insufficient to support spatial scalability
    - Does not include chain info or decode targets
  - Incompatible with WebCodecs



```
dictionary RTCEncodedVideoFrameMetadata {
    unsigned long long frameId;
    sequence<unsigned long long> dependencies;
    unsigned short width;
    unsigned short height;
    unsigned long spatialIndex;
    unsigned long temporalIndex;
    unsigned long synchronizationSource;
    octet payloadType;
  sequence<unsigned long> contributingSources;
};
```

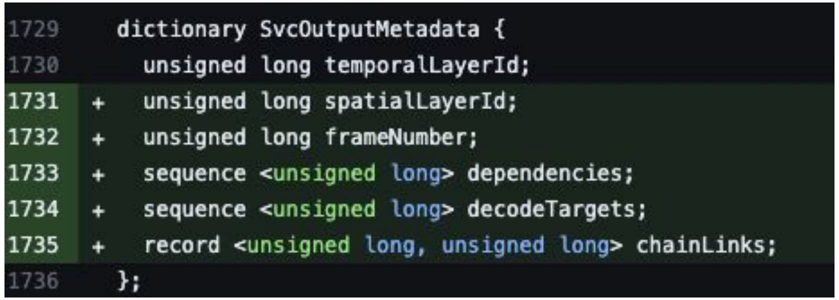


```
dictionary EncodedVideoChunkMetadata {
  VideoDecoderConfig decoderConfig;
  SvcOutputMetadata svc;
  BufferSource alphaSideData;
};

dictionary SvcOutputMetadata {
  unsigned long temporalLayerId;
};
```

Issue 170: SVC Metadata (cont'd)

- Previous discussion in WEBRTC WG
  - Submit a WebCodecs PR, bring solution back to WEBRTC WG.
- Discussion in MEDIA WG
  - PR 654: Extend EncodedVideoFrameMetadata for SVC
  - Detailed proposal



```
  dictionary SvcOutputMetadata {
    unsigned long temporalLayerId;
+   unsigned long spatialLayerId;
+   unsigned long frameNumber;
+   sequence <unsigned long> dependencies;
+   sequence <unsigned long> decodeTargets;
+   record <unsigned long, unsigned long> chainLinks;
  };
```

  - frameNumber, dependencies should be unsigned long
  - WebCodecs: what is the state of implementation in Chromium?
    - Peter investigated.
  - Encoded Transform: Have dependencies, spatialIndex and temporalIndex been implemented? If not, should we remove them?

Issue 93: MediaStreamTrack audio delay/glitch capture stats (Henrik)

Audio capture stats were recently added in getStats()'s RTCAudioSourceStats:

```
double                  droppedSamplesDuration;
unsigned long           droppedSamplesEvents;
double                  totalCaptureDelay;
unsigned long long      totalSamplesCaptured;
```

RTCAudioSourceStats are only present when a MediaStreamTrack is attached to an RTCRtpSender. These audio capture metrics are "*Only applicable if this media source is backed by an audio capture device.*"

Important capture quality metrics not available elsewhere:

- Allows calculating glitches in captured audio:
  `delta droppedSamplesDuration / delta totalSamplesDuration`
- Allows calculating average capture delay:
  `delta totalCaptureDelay / delta totalSamplesCaptured`

Issue 93: MediaStreamTrack audio delay/glitch capture stats (Henrik)

Issue webrtc-stats#741 was filed, making the metrics *Feature at Risk* because:

1. getUserMedia() is frequently used outside of WebRTC, so WebRTC getStats() is a bad location to put these metrics.
2. We should talk about audio frames, not audio samples (e.g. multi-channel).
   - As per terminology section, the stats spec actually means “audio frames” when it says “audio samples”. We should say what we mean here: audio frames.
3. It was not obvious why audio may be dropped, so we should clarify that this happens when audio is not processed in a timely manner.

MediaStreamTrack.getFrameStats() was recently added for video capture metrics. There is a need to have track audio capture metrics as well. **Proposal:**

1. Mark “ready for PR” to move capture metrics to MediaStreamTrack and add clarifications.
2. track.getFrameStats(), track.getAudioStats(), track.getStats()?

PR 173 adding presentationTime to RTCEncodedVideoFrameMetadata (Tony)

- Scenario: Web App doing both Raw transforms and Encoded transforms on a local video track sent over a PC
  - For a given encoded frame, which (transformed) raw frame does it correspond to?
- Provided in WebCodecs via EncodedVideoChunk.timestamp and VideoFrame.timestamp - matching presentation timestamps
- RTCEncodedFrame has a `timestamp`, alas the RTP timestamp
- => Can we add the presentation timestamp to RTCEVFMetadata, to match VideoFrame.timestamp?

Discussion (End Time: 08:40)

-

RtpTransport (Peter Thatcher)
Start Time: 08:40 AM
End Time: 09:10 AM

WebRTC Combines Media and Transport

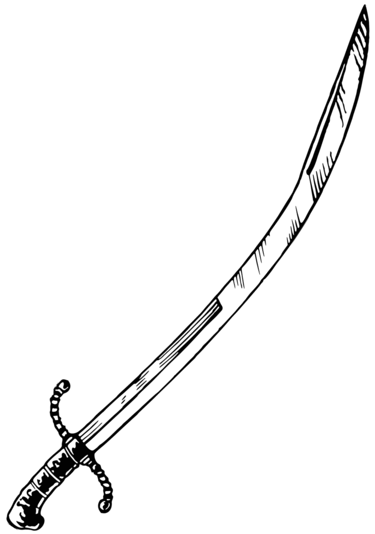
(https://commons.wikimedia.org/wiki/File:Antoon_Claeissens_-_The_Judgement_of_Solomon_-_WGA04956.jpg)

The Combination has Pros and Cons…

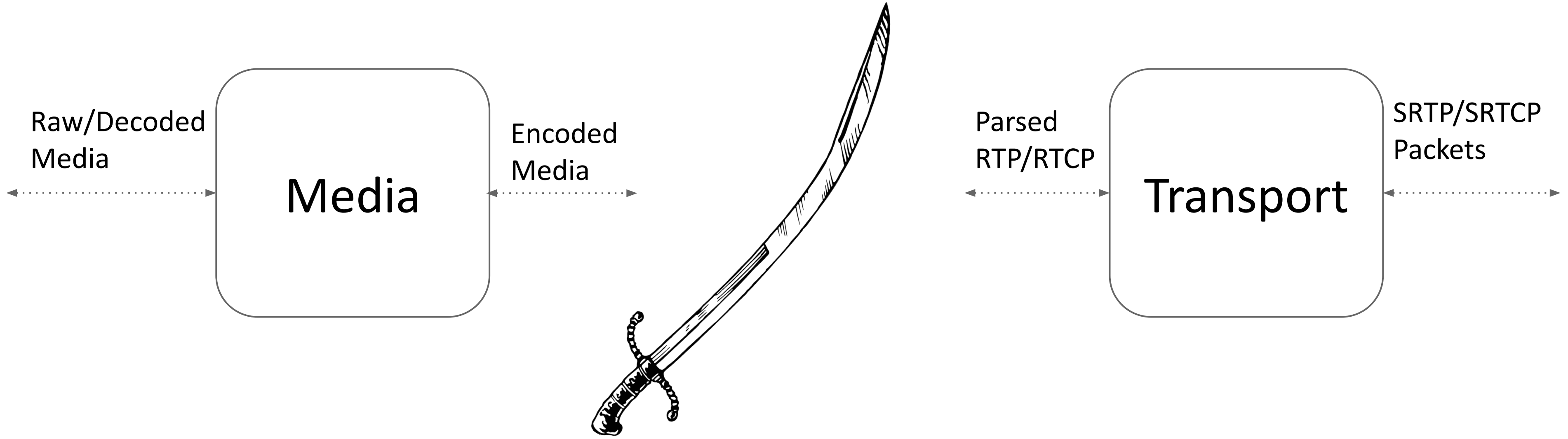
- Pros
  - Much of the time, “it just works”.
- Cons:
  - Difficult to do things that can’t be expressed in SDP
    - Control of encoding parameters (e.g. screen content coding)
    - Differential resilience (e.g. RTX/FEC/RED only for base layer)
    - Custom resilience (e.g. support for custom FEC)
    - Controlling and monitoring hardware acceleration (encode/decode)
  - Difficult to support bleeding edge codecs
    - Bring your own audio codec (BYOC)
    - Video codecs already supported in low-level APIs (e.g. receiving HEVC)
    - Codecs that require new RTCP messages (e.g. LRR and VP9/AV1)

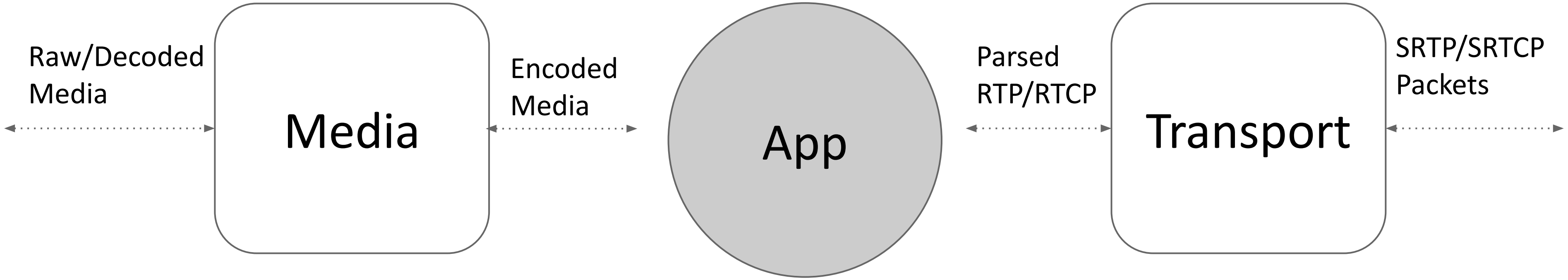
What if we separated the two halves?

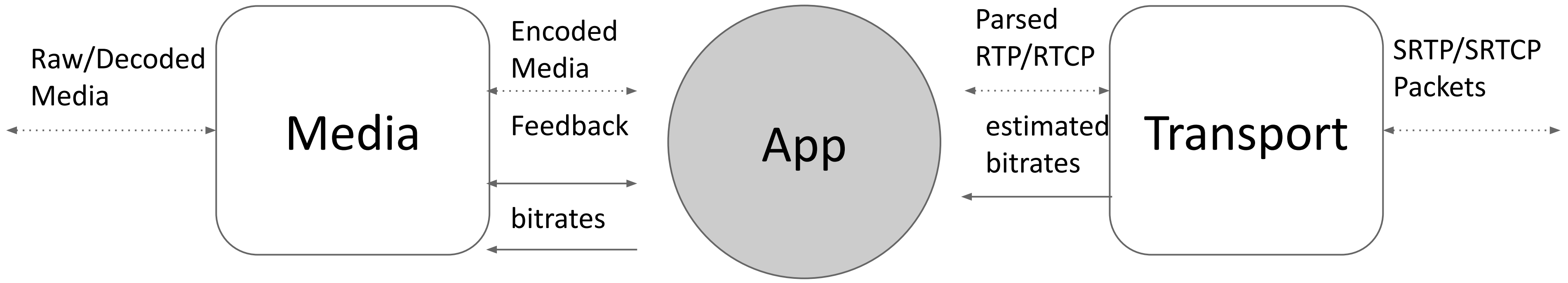
Media



Transport


Raw/Decoded Media
Media
Encoded Media
Parsed RTP/RTCP
Transport
SRTP/SRTCP Packets


Raw/Decoded Media
Media
Encoded Media
App
Parsed RTP/RTCP
Transport
SRTP/SRTCP Packets



In this presentation we will focus on the RTPTransport.

RtpTransport

- Sends and receives RTP/RTCP
- Encrypted
- Congestion controlled
- RTP sender API is "stream of packets in; stream of packets out on the wire"
- RTP receiver API is “stream of packets received on the wire, stream of packets out”
- Supports workers
- Can be combined with WebCodecs
- Can be constructed from a DtlsTransport
  - DtlsTransport can come from an RTCPeerConnection negotiating datachannel-only
  - Also possible to construct a stand-alone DtlsTransport from an IceTransport.

Relationship to RTCPeerConnection

An RtpTransport is constructed with a DtlsTransport, which currently requires a PeerConnection with SCTP data channels, which requires some SDP (although data-channel-only SDP is far more simple than RTP SDP). Here is how we could fix that (when combined with the new IceTransport APIs):

```
[Constructor(RTCIceTransport transport, sequence<RTCCertificate> certificates), Exposed=Window]
partial interface RTCDtlsTransport {
  undefined start(RTCDtlsParameters remoteParameters);
  undefined stop();
};
dictionary RTCDtlsParameters {
  RTCDtlsRole role = "auto";
  required sequence<RTCDtlsFingerprint> fingerprints;
};
```

Relationship To WebCodecs

WebCodecs can be used with RtpTransport.

You do your own packetization, jitter buffer, and rate adaptation

Compared to WebTransport

If you squint, RtpTransport looks a lot like WebTransport with datagrams.

But it can also be p2p, which WebTransport cannot (yet?).

And it has good latency-sensitive congestion control which WebTransport does not (yet?)

NV07

“A user agent can receive audio/video without requiring construction of a corresponding sender object”

- App can create RtpTransport without an RTCPeerConnection.

NV08

“It is possible to select the sending and/or receiving codec as well as rtcp parameters and header extensions without negotiation.”

- App can control RTP header extensions sent/received by RtpTransport without an RTCPeerConnection.
- App can directly control RTCP control messages sent/received by RtpTransport without an RTCPeerConnection

NV09

“The user agent must be able to control robustness (RTX, RED, FEC) applied to individual simulcast and SVC layers.”

- App can directly control RTX, RED, and FEC using the RtpTransport for all RTP packets, including those for different layers.
- App can also do custom forms of RTX, RED, and FEC.

NV15

“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.”

- App can know estimated bitrate of RtpTransport
- App can send and receive data with RtpTransport

NV38 (Non-consensus)

"RtpAudioSender and RtpVideoReceiver have a control for the minimum and maximum jitter buffer delay. If the web app wishes, it could set the maximum to 0, which disables the jitter buffer, and then run its own jitter buffer before providing inputs. "

- App can directly set min and max jitter buffer delay.
  - RTPTransport receiver pipeThrough Jitter Buffer TransformStream which can be configured for min/max delay.

NV39 (Non-consensus)

“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.”

- App can directly access encoded data and metadata from RtpTransport
- App can know estimated bitrate of RtpTransport
- App can modify and send RTP packets from one RtpTransport to others

NV40 (Non-consensus)

“An application can create an outgoing WebRTC connection without activating an encoder.”

- App can create an RtpTransport without creating an AudioSender/VideoSender

NV43 (Non-consensus)

“The application can modify metadata on outgoing frames so that they fit smoothly within the expected sequence of timestamps and sequence numbers.”

- App can directly control metadata of sent using RTP transport.

WebIDL for RtpTransport

```
interface RtpTransport {
  constructor(DtlsTransport dtlsTransport, RtpTransportParameters parameters = {});

  Promise<RtpSentPacket> sendRtpPacket(RtpPacketInit rtpPacket);
  attribute eventhandler onrtppacketreceived;  // of RtpReceivedPacket

  Promise<void> sendRtcpPacket(RtcpPacketInit rtcpPacket);
  attribute eventhandler onrtcppacketreceived;  // or ReceivedRtcpPacket

  readonly attribute unsigned long targetSendRate;  // bps
  attribute eventhandler ontargetsendratechanged;
}
```

RtpTransport example (send)

```
let peerConnection = …;
let dtlsTransport = peerConnection.getTransceivers()[0].dtlsTransport;
let rtpTransport = new RtpTransport(dtlsTransport);

rtpTransport.sendRtpPacket({
  payloadType: 99,
  ssrc: 999,
  timestamp: …,
  payload: …,
  …
});

rtpTransport.sendRtcpPacket({
  …
});
```

RtpTransport example (rate control)

```
rtpTransport.ontargetsendratechanged = () => {
  // Allocate bitrate using rtpTransport.targetSendRate
};
```

TODO

- Discuss/define the Media side

Questions

Is this a good direction to go?

Discussion (End Time: 09:10)

-

IceController (Sameer & Peter)

Start Time: 09:10 AM

End Time: 09:40 AM

WebRTC ICE improvements

Incremental this time

Since Last Time

- 1 proposal with lots of common ground
- Split into many increments
- Meets all the NV requirements

Order of incremental improvements

- Prevent removal of candidate pairs
- Remove candidate pairs
- Control selection of candidate pair
- (?) Observe candidate pair states
- Observe result/RTT of outgoing checks
- Control frequency of outgoing checks of particular candidate pairs
- Prevent outgoing checks of particular candidate pairs
- Control order and timing of outgoing checks
- Observe presence of not of incoming checks or media for particular candidate pairs
- Gather local candidates for new network interfaces
- Re-gather local candidates of previously failed network interfaces
- Prevent removal of local candidates
- Remove local candidates
- Construct IceTransport without PeerConnection
- Support forking

N04 - maintain multiple candidate pairs

- Prevent removal of candidate pairs
- Remove candidate pairs
- Control selection of candidate pair
- (?) Observe candidate pair states
- Observe result/RTT of outgoing checks
- Control frequency of outgoing checks of particular candidate pairs
- Prevent outgoing checks of particular candidate pairs
- Control order and timing of outgoing checks
- Observe presence of not of incoming checks or media for particular candidate pairs
- Gather local candidates for new network interfaces
- Re-gather local candidates of previously failed network interfaces
- Prevent removal of local candidates
- Remove local candidates
- Construct IceTransport without PeerConnection
- Support forking

N15 - ensure a low and consistent latency

- Prevent removal of candidate pairs
- Remove candidate pairs
- Control selection of candidate pair
- (?) Observe candidate pair states
- Observe result/RTT of outgoing checks
- Control frequency of outgoing checks of particular candidate pairs
- Prevent outgoing checks of particular candidate pairs
- Control order and timing of outgoing checks
- Observe presence of not of incoming checks or media for particular candidate pairs
- Gather local candidates for new network interfaces
- Re-gather local candidates of previously failed network interfaces
- Prevent removal of local candidates
- Remove local candidates
- Construct IceTransport without PeerConnection
- Support forking

N14 - minimize ICE connectivity checks

- Prevent removal of candidate pairs
- Remove candidate pairs
- Control selection of candidate pair
- (?) Observe candidate pair states
- Observe result/RTT of outgoing checks
- Control frequency of outgoing checks of particular candidate pairs
- Prevent outgoing checks of particular candidate pairs
- Control order and timing of outgoing checks
- Observe presence of not of incoming checks or media for particular candidate pairs
- Gather local candidates for new network interfaces
- Re-gather local candidates of previously failed network interfaces
- Prevent removal of local candidates
- Remove local candidates
- Construct IceTransport without PeerConnection
- Support forking

N01 - control candidate gathering

- Prevent removal of candidate pairs
- Remove candidate pairs
- Control selection of candidate pair
- (?) Observe candidate pair states
- Observe result/RTT of outgoing checks
- Control frequency of outgoing checks of particular candidate pairs
- Prevent outgoing checks of particular candidate pairs
- Control order and timing of outgoing checks
- Observe presence of not of incoming checks or media for particular candidate pairs
- Gather local candidates for new network interfaces
- Re-gather local candidates of previously failed network interfaces
- Prevent removal of local candidates
- Remove local candidates
- Construct IceTransport without PeerConnection
- Support forking

N01 - control candidate pruning

- Prevent removal of candidate pairs
- Remove candidate pairs
- Control selection of candidate pair
- (?) Observe candidate pair states
- Observe result/RTT of outgoing checks
- Control frequency of outgoing checks of particular candidate pairs
- Prevent outgoing checks of particular candidate pairs
- Control order and timing of outgoing checks
- Observe presence of not of incoming checks or media for particular candidate pairs
- Gather local candidates for new network interfaces
- Re-gather local candidates of previously failed network interfaces
- Prevent removal of local candidates
- Remove local candidates
- Construct IceTransport without PeerConnection
- Support forking

N02 - establish multiple connections

- Prevent removal of candidate pairs
- Remove candidate pairs
- Control selection of candidate pair
- (?) Observe candidate pair states
- Observe result/RTT of outgoing checks
- Control frequency of outgoing checks of particular candidate pairs
- Prevent outgoing checks of particular candidate pairs
- Control order and timing of outgoing checks
- Observe presence of not of incoming checks or media for particular candidate pairs
- Gather local candidates for new network interfaces
- Re-gather local candidates of previously failed network interfaces
- Prevent removal of local candidates
- Remove local candidates
- Construct IceTransport without PeerConnection
- Support forking

Prevent Candidate Pair Removal

Option A (cancellable event)

```
partial interface RTCIceTransport  {
  attribute EventHandler oncandidatepairremoved;
};
interface IceCandidatePairRemovedEvent : Event {
  readonly attribute RTCIceCandidatePair pair;
  readonly attribute bool cancellable;
};
```

Option B (turn on/off automatic behavior)

```
partial interface RTCIceTransport  {
  attribute EventHandler oncandidatepairadded;
};
partial interface RTCIceCandidatePair  {
  attribute bool removable;  // default is true
}
```

Remove Candidate Pairs

```
partial interface RTCIceTransport  {
  undefined removeCandidatePairs(sequence<RTCIceCandidatePair> pairs);
};
```

Control Selection of Candidate Pair

```
partial interface RTCIceTransport  {
  Promise<undefined> setSelectedCandidatePair(RTCIceCandidatePair pair);
};
```

(?) Observe Candidate Pair State Changes

```
partial interface RTCIceCandidatePair {
  readonly attribute RTCIceCandidatePairState state;
  attribute EventHandler onstatechange;
};

// Almost the same as RTCIceConnectionState
enum RTCIceCandidatePairState {
  "new", "checking", "connected", "disconnected", "failed", "closed",
  "paused",
}
```

Observe result/RTT of outgoing checks

```
partial interface RTCIceTransport {
  attribute EventHandler onicechecksentandresolved; // TODO: better name
};
// Better name
interface RTCIceCheckSentAndResolvedEvent : Event {
  readonly attribute RTCIceCandidatePair candidatePair;
  readonly attribute DOMHighResTimeStamp checkSent;
  readonly attribute DOMHighResTimeStamp? responseReceived;
  readonly attribute IceCheckErrorType? error;
};
```

Control frequency of outgoing checks of candidate pairs

```
partial interface RTCIceCandidatePair {
   attribute unsigned long checkInterval;  // milliseconds
};
```

Prevent outgoing checks of particular candidate pairs

Option A (cancellable event)

```
partial interface RTCIceTransport {
  attribute EventHandler onicecheckproposed;
};
// Cancellable
interface RTCIceCheckProposedEvent : Event {
  readonly attribute RTCIceCandidatePair pair;
};
```

Option B (turn on/off automatic behavior)

```
partial interface RTCIceCandidatePair  {
  attribute bool checkable;  // default true
}
```

Control order and timing of outgoing checks

```
partial interface RTCIceCandidatePair {
  undefined sendCheck(optional long timeout);
};
```

Observe presence of incoming checks or media

```
partial interface RTCIceCandidatePair {
  readonly attribute DOMHighResTimeStamp? lastCheckReceived;
  readonly attribute DOMHighResTimeStamp? lastPacketReceived;
};
```

Gather local candidates for new network interfaces

Option A (incompatible with ICE forking)

```
partial interface RTCIceTransport {
  attribute bool gatherContinuallyCandidates;
};
```

Option B (compatible with ICE forking)

```
partial interface RTCIceTransport {
  readonly attribute RTCIceGatherer gatherer;
}

interface RTCIceGatherer  {
  attribute bool gatherContinually;
};
```

Re-gather candidates of previously failed network interfaces

Option A (incompatible with ICE forking)

```
partial interface RTCIceTransport {
  Promise<undefined> gather(IceGatherParameters);
};

dictionary IceGatherParameters {
  bool regatherNetworkInterfacesInUse = false;
};
```

Option B (compatiable with ICE forking)

```
partial interface RTCIceTransport {
  readonly attribute RTCIceGatherer gatherer;
}

interface RTCIceGatherer  {
  Promise<undefined> gather(IceGatherParameters);
};

dictionary IceGatherParameters {
  bool regatherNetworkInterfacesInUse = false;
};
```

Prevent removal of local candidates

Option A (cancellable event; no forking)

```
partial interface RTCIceTransport {
  attribute EventHandler onlocalcandidateremoved;
};
// Cancellable
interface RTCIceCandidateRemovedEvent : Event {
  readonly attribute RTCIceCandidate candidate;
};
```

Option B (turn on/off automatic behavior; forking)

```
partial interface RTCIceGatherer {
  attribute EventHandler oncandidateadded;
  attribute EventHandler onlocalcandidateremoved;
  sequence<RTCLocalIceCandidate> getCandidates();
}
interface RTCLocalIceCandidate : RTCIceCandidate {
  attribute bool removeable;  // default true
}
```

Remove local candidates

Option A (incompatible with ICE forking)

```
partial interface RTCIceTransport {
  attribute EventHandler onlocalcandidateadded;
  undefined removeLocalCandidates(
     sequence<RTCIceCandidate>);
}
```

Option B (compatible with ICE forking)

```
partial interface RTCIceGatherer {
  undefined removeCandidates(
     sequence<RTCLocalIceCandidate>);
}
```

Construct and use without PeerConnection

```
partial interface IceTransport {
  constructor();
  undefined start(RTCIceTransportParameters parameters,
                  RTCIceParameters remoteParameters);
  undefined addRemoteCandidate(RTCIceRemoteCandidateInit);
}
dictionary RTCIceTransportParameters {
  RTCIceGatherer gatherer;
  required RTCIceRole role;
}
dictionary RTCIceRemoteCandidateInit {
  … same fields as RTCIceCandidate;
}
```

```
partial interface IceGatherer {
  constructor();
  RTCIceParameters getLocalParameters();
  undefined gather(IceGatherParameters parameters);
}
partial dictionary IceGatherParameters {
  sequence<RTCIceServer> iceServers = [];
  RTCIceTransportPolicy iceTransportPolicy = "all";
}
```

Ice Forking

```
// For use with PeerConnection only
// Without PeerConnection, nothing more is needed
partial dictionary RTCConfiguration {
  IceGatherer iceGatherer;
}
```

Summary

- Many small, incremental improvements are each valuable, and make for a great whole.
- Nothing very difficult to implement or use.
- Does almost everything anyone has ever asked for.

Questions

- Does the Working Group want to pursue this direction?
- Use cancellable events or no?
- Leave the door open to ICE forking (IceGather) or no?
- Can we change an IceCandidatePair from a dictionary to an interface?

All combined (with PeerConnection)

```
partial interface RTCIceTransport  {
  readonly attribute RTCIceGatherer gatherer;
  undefined removeCandidatePairs(sequence<IceCandidatePair>);
  Promise<undefined> setSelectedCandidatePair(IceCandidatePair);
  attribute EventHandler oncandidatepairadded;
  attribute EventHandler onicechecksentandresolved;
}
```

```
partial interface RTCIceCandidatePair  {
  attribute bool removable;
  attribute unsigned long checkInterval;
  readonly attribute RTCIceCandidatePairState state;
  attribute EventHandler onstatechange;
  readonly attribute DOMHighResTimeStamp? lastCheckReceived;
  readonly attribute DOMHighResTimeStamp? lastPacketReceived;
}
```

```
dictionary RTCIceRemoteCandidateInit {
  … same fields as RTCIceCandidate …
}
```

```
enum RTCIceCandidatePairState {
  "new", "checking", "connected", "disconnected", "failed",
  "closed", "paused"
}
```

```
interface RTCIceGatherer  {
  attribute bool gatherContinually;
  sequence<RTCLocalIceCandidate> getCandidates()
  undefined removeCandidates(sequence<RTCLocalIceCandidate>);
  attribute EventHandler oncandidateadded;
  attribute EventHandler oncandidateremoved;
}
```

```
interface RTCLocalIceCandidate : RTCIceCandidate {
  attribute bool removeable;
}
```

```
interface RTCIceCheckSentAndResolvedEvent : Event {
  readonly attribute RTCIceCandidatePair candidatePair;
  readonly attribute DOMHighResTimeStamp checkSent;
  readonly attribute DOMHighResTimeStamp? responseReceived;
  readonly attribute IceCheckErrorType? error;
}
```

All combined (without PeerConnection)

```
partial interface RTCIceTransport  {
  constructor();
  undefined start(RTCIceTransportParameters parameters,
                  RTCIceParameters remoteParameters);
  undefined removeCandidatePairs(sequence<IceCandidatePair>);
  Promise<undefined> setSelectedCandidatePair(IceCandidatePair);
  attribute EventHandler oncandidatepairadded;
  attribute EventHandler onicechecksentandresolved;
}

dictionary RTCIceTransportParameters {
  RTCIceGatherer gatherer;
  required RTCIceRole role;
}

partial interface RTCIceCandidatePair  {
  attribute bool removable;
  attribute unsigned long checkInterval;
  readonly attribute RTCIceCandidatePairState state;
  attribute EventHandler onstatechange;
  readonly attribute DOMHighResTimeStamp? lastCheckReceived;
  readonly attribute DOMHighResTimeStamp? lastPacketReceived;
}

dictionary RTCIceRemoteCandidateInit {
  … same fields as RTCIceCandidate …
}
```

```
interface RTCIceGatherer  {
  constructor();
  Promise<undefined> gather(IceGatherParameters);
  RTCIceParameters getLocalParameters();
  sequence<RTCLocalIceCandidate> getCandidates()
  undefined removeCandidates(sequence<RTCLocalIceCandidate>);
  attribute EventHandler oncandidateadded;
  attribute EventHandler oncandidateremoved;
}

dictionary IceGatherParameters: RTCIceParameters {
  sequence<RTCIceServer> iceServers = [];
  RTCIceTransportPolicy iceTransportPolicy = "all";
  bool regatherNetworkInterfacesInUse = false;
  bool gatherContinually = true;
}

interface RTCLocalIceCandidate : RTCIceCandidate {
  attribute bool removeable;
}

interface RTCIceCheckSentAndResolvedEvent : Event {
  readonly attribute RTCIceCandidatePair candidatePair;
  readonly attribute DOMHighResTimeStamp checkSent;
  readonly attribute DOMHighResTimeStamp? responseReceived;
  readonly attribute IceCheckErrorType? error;
}

enum RTCIceCandidatePairState {
  "new", "checking", "connected", "disconnected", "failed",
  "closed", "paused"
}
```

Discussion (End Time: 09:40)

-

playoutDelay (Jan-Ivar)

Start Time: 09:40 AM

End Time: 09:55 AM

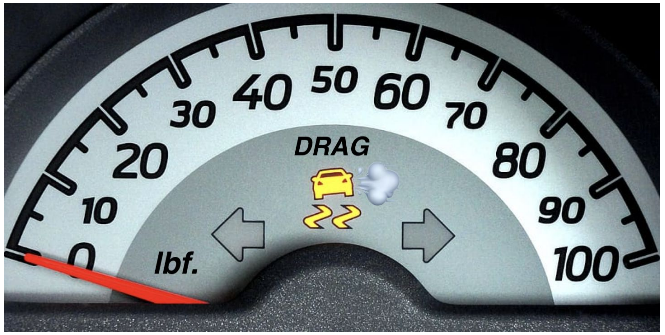
Issue 156 et al.: playoutDelay (1/2) (Jan-Ivar)

Some open issues around playoutDelay (a.k.a. playoutDelayHint in Chrome):

- #157 Does playoutDelay of video affect jitter buffer of synced audio and vice versa?
- #156 Should we clarify playoutDelay value is jitter buffer depth?
- #155 playoutDelay should use milliseconds.
- #123 RTCRtpReceiver.playoutDelay...Hint or no Hint?
- #46 How does a developer decide on a value for `playoutDelay` ?
- #12 Testability of playoutDelayHint ←--- See here for most recent discussion.

Summary of concerns / insights from #12:

1. “Delay” is a *measurement* of a **negative side-effect**
2. Vague input (jitter buf + AV sync/playout path delay?)
3. Hard to test + confusing to implement (webrtc 15085 induced video delay outside jitter buffer wasting time)
4. Instead, the desirable positive goal is **jitter-free media**



Drag’o’meter

Issue 156 et al.: playoutDelay (2/2) (Jan-Ivar)

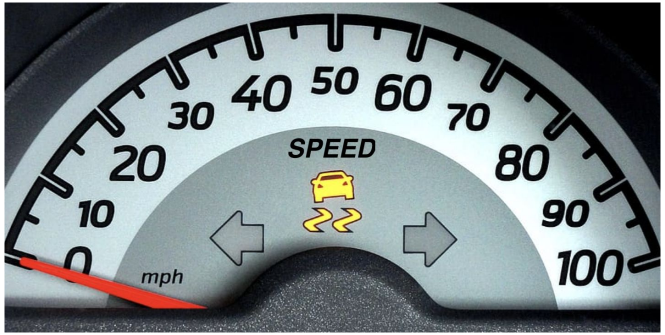
Instead, make the input value the target jitter buffer *depth*, and nothing more:

```
receiver.jitterBufferDepth = 150; // in milliseconds* (double)
```

And compare to the gradually matching delay measurement from stats (fiddle):

```
let lastInbound;
while (true) {
  const stats = receiver.getStats();
  const inbound = [...stats.values()].find(({type}) => type == "inbound-rtp");
  console.log(`Jitter buffer delay = ${getJitterBufferDelayMs(inbound, lastInbound)} ms.`);
  lastInbound = inbound;
  await wait(1000);
}

function getJitterBufferDelayMs(inbound, oldInbound) {
  if (!oldInbound) return 0;
  return ((inbound.jitterBufferDelay - oldInbound.jitterBufferDelay) /
   (inbound.jitterBufferEmittedCount - oldInbound.jitterBufferEmittedCount)
   * 1000).toFixed(2);
}
```



Speedometer

* W3C Design Principle § 8.3. Use milliseconds for time measurement

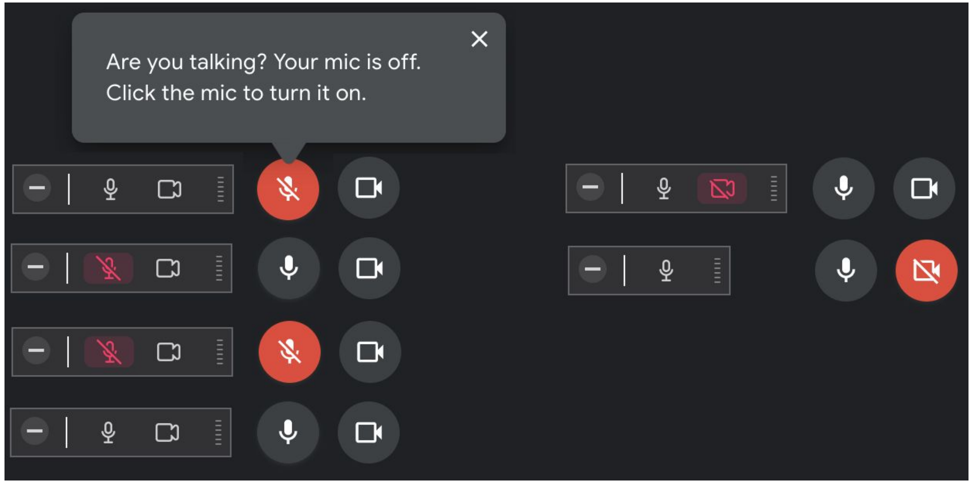
Discussion (End Time: 09:55)

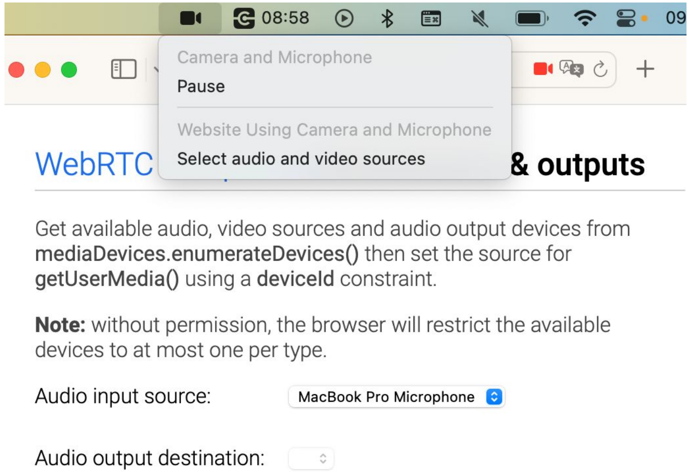
-

Extra issues - time permitting

- Issue 39: Solve user agent camera/microphone double-mute
- Issue 263: CaptureStartFocusBehavior.no-focus-change is misleading

Issue 39: Solve user agent camera/microphone double-mute


Are you talking? Your mic is off.
Click the mic to turn it on.


08:58
09
Camera and Microphone
Pause
Website Using Camera and Microphone
Select audio and video sources
WebRTC
& outputs

Get available audio, video sources and audio output devices from **mediaDevices.enumerateDevices()** then set the source for **getUserMedia()** using a **deviceId** constraint.

Note: without permission, the browser will restrict the available devices to at most one per type.

Audio input source: MacBook Pro Microphone

Audio output destination:

Issue 39: Solve user agent camera/microphone double-mute

- **OS level camera indicator**
  - Widely deployed: the green pill

    → Web sites tend to stop camera when muting

- **OS level microphone indicator**
  - Not as widely deployed (but still deployed in some OSes)

    → Web sites DO NOT tend to stop microphone when muting
  - Ability to detect whether user is speaking
    - kAUVoiceIOProperty_MutedSpeechActivityEventListener

- **Proposal: Allow application to request mute/unmute of camera/microphone capture**

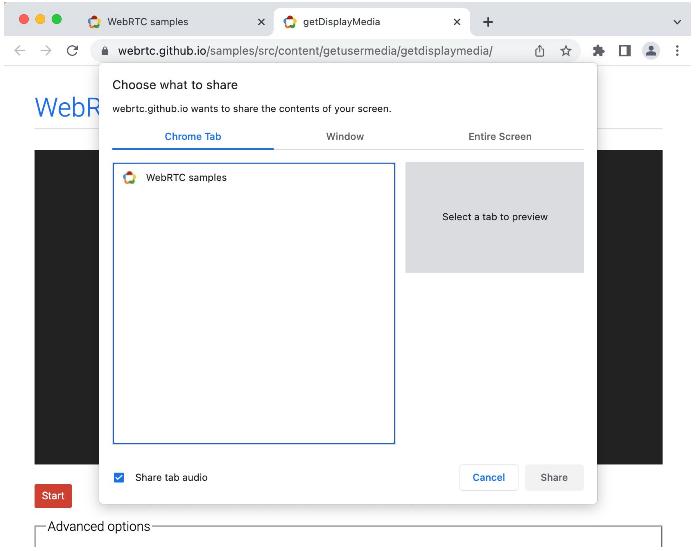
Issue 39: Solve user agent camera/microphone double-mute

- MediaStreamTrack API
  - `Promise requestMute/requestUnmute()`
    - Mute the track's source
- InputDeviceInfo API
  - `Promise requestMute/requestUnmute()`
    - Mute the device, similar to track's source
- Navigator API
  - `Promise requestCaptureMute/requestUnmute(kind)`
    - Mute all capture devices of a web page

- `requestMute` MAY mute more than just the track's source (up to UA)
- `requestUnmute` MAY prompt (up to UA)

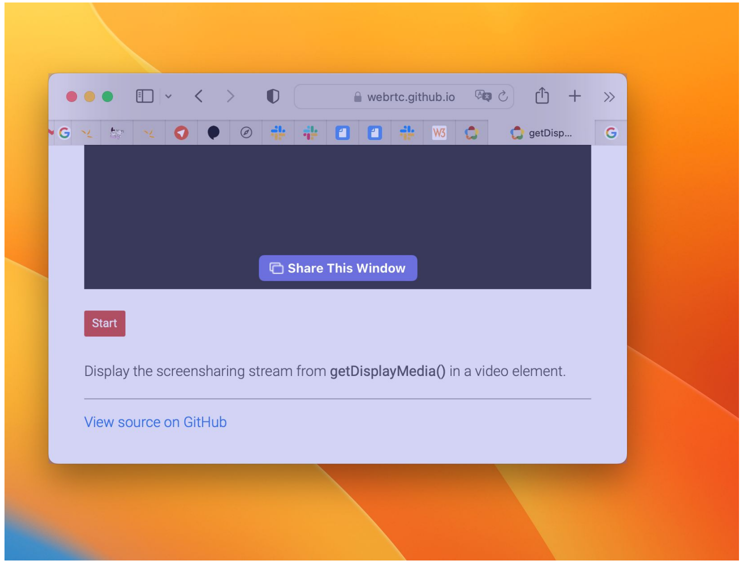
Issue 263: CaptureStartFocusBehavior.no-focus-change is misleading

- **CaptureStartFocusBehavior getDisplayMedia option**
  - `"focus-captured-surface"`: focus the user selected surface
  - `"no-focus-change"`: focus the capturing web application
- `"no-focus-change"` **definition is based on a UA picker model, not on OS level picker**


WebRTC samples
getDisplayMedia
webrtc.github.io/samples/src/content/getusermedia/getdisplaymedia/
WebR
Start
Advanced options

Choose what to share

webrtc.github.io wants to share the contents of your screen.



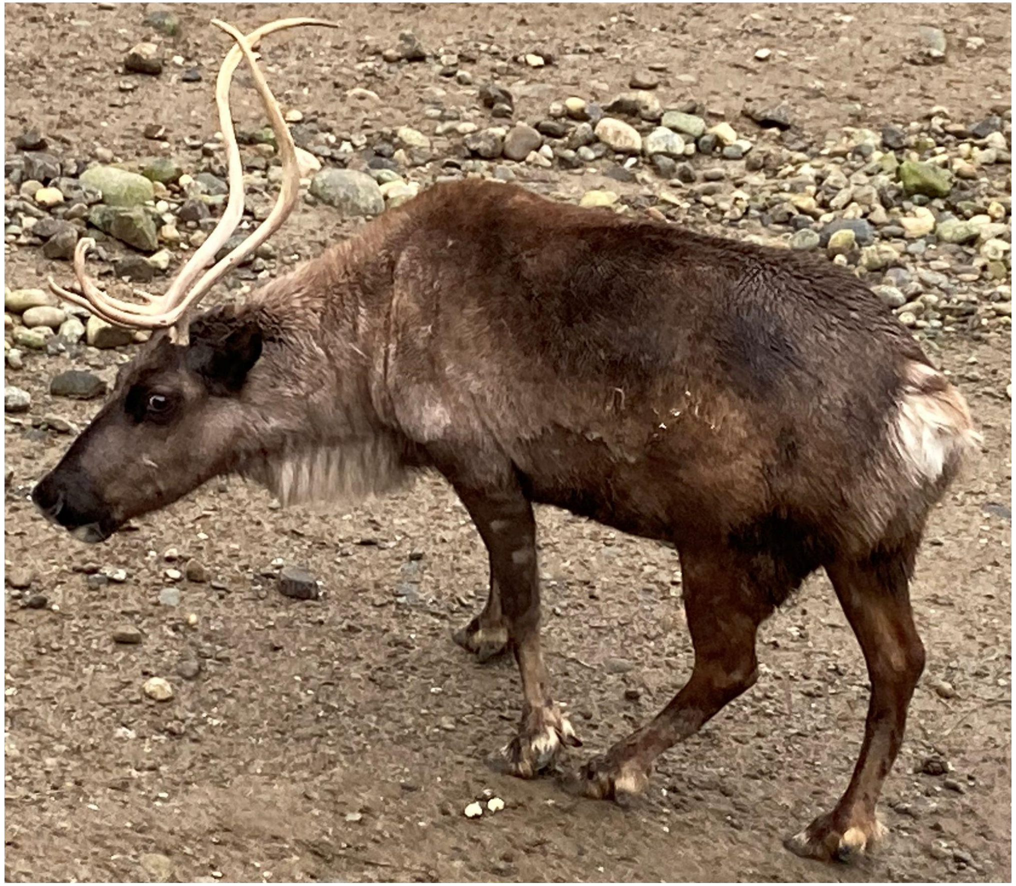
Display the screensharing stream from getDisplayMedia() in a video element.

Issue 263: CaptureStartFocusBehavior.no-focus-change is misleading

- **Main proposal: introduce a new focus behavior value**
  - `“focus-capturing-application”`
    - Equivalent to current `“no-focus-change”`

- **What about `“no-focus-change”`?**
  - Remove `“no-focus-change”`
  - Redefine `“no-focus-change”` as use the default focus behavior

Name that Mammal



Thank you

Special thanks to:

WG Participants, Editors, Chairs and The Mammal of the Month^{TM}