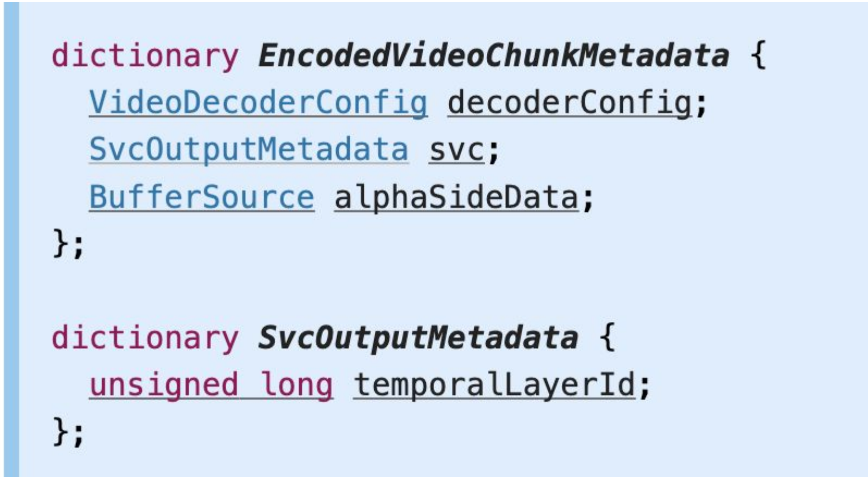
For Discussion Today

- Issue 170/Issue 619: Consistent SVC metadata
- Media Pipeline Architecture

Issue 170/Issue 619: Consistent SVC metadata

- WebCodecs defines EncodedChunkMetadata as follows:



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

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

- Dictionary has structure to allow for future expansion of SvcOutputMetadata dictionary.

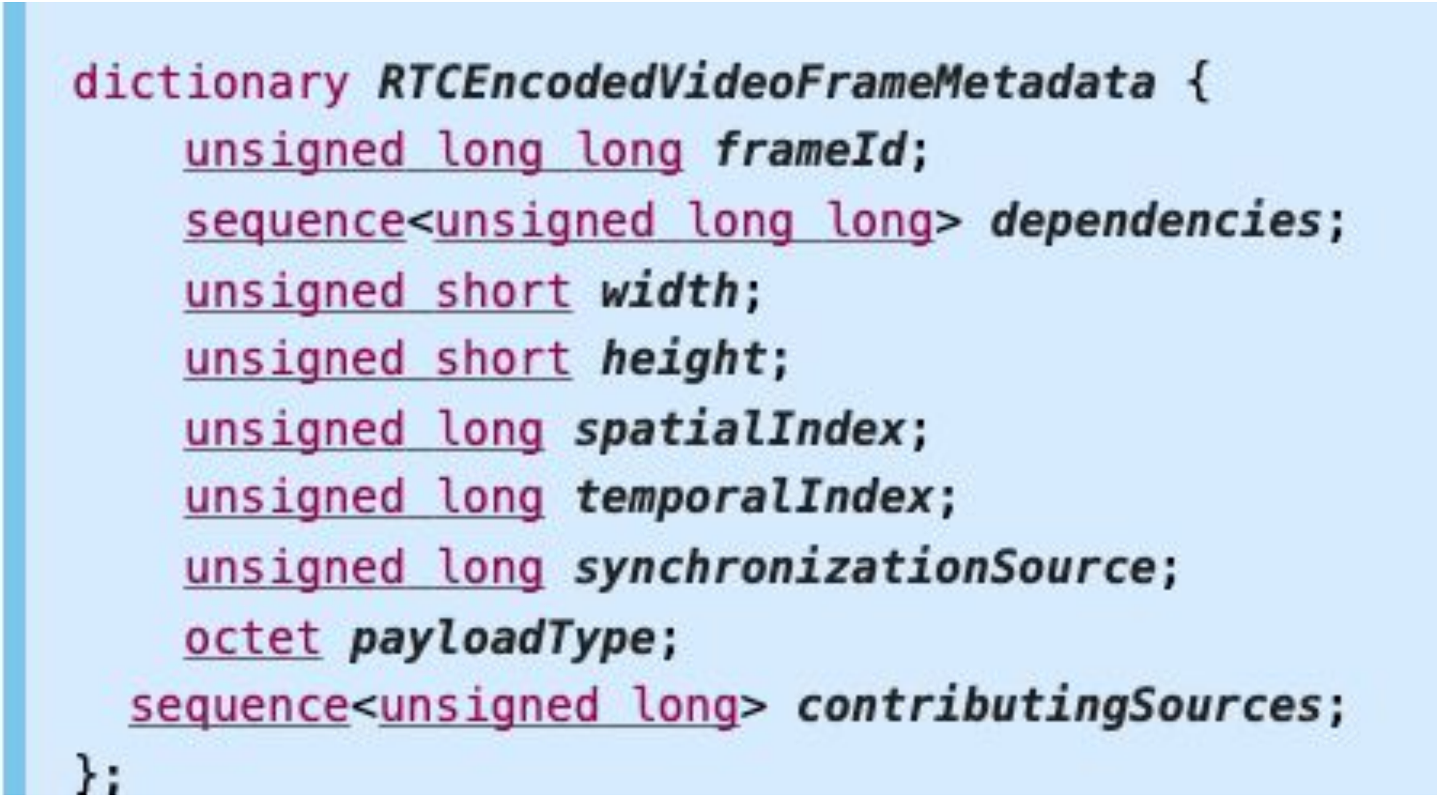
Issue 170/Issue 619: Consistent SVC metadata (cont’d)

- Complete WebCodecs SVC metadata proposal is based on the information included within the Dependency Descriptor RTP header extension:

```
dictionary EncodedVideoChunkMetadata {
// Number for identifying this frame in |dependsOnIds| and |chainLinks| (for other chunks).
unsigned short frameNumber;

// List of frameNumbers that this chunk depends on. Used to detect/handle network loss. Decoding out of order is an error.
list<unsigned long> dependsOnIds;

// IDs of the spatial layer and temporal layer this chunk belongs to.
unsigned long spatialLayerId;
unsigned long temporalLayerId;

// List of decoder targets this frame participates in. Used to know whether this frame should be sent (forwarded) to a given
receiver  depending on what decode targets the receiver is expecting. Decode target is a numerical index determined by the
encoder. No commitment that a particular number implies a given layer.
list<unsigned long> decodeTargets;

// Mapping of decode target -> the last important frame to decode prior to "this"  frame for the given decode target.
// Used to ensure we preserve decode order for the desired decode target. It is insufficient to simply satisfy the
dependencies for the current frame. See example.
map<unsigned long, unsigned long> chainLinks;
};
```

Issue 170/Issue 619: Consistent SVC metadata (cont’d)

- Comparison with RTCEncodedVideoFrameMetadata:



```
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;
};
```

Issue 170/Issue 619: Consistent SVC metadata (cont'd)

- Issues:
  - Name differences
    - `temporalLayerId` vs. `temporalIndex`
    - `spatialLayerId` vs. `spatialIndex`
  - Type mismatches:
    - `unsigned short frameNumber` vs. `unsigned long long frameId`
    - `sequence <unsigned long> dependsOnIds` vs. `sequence <unsigned long long> dependencies`
  - Missing information
    - `sequence <unsigned long> decodeTargets`
      - List of decode targets this frame participates in. Used to determine whether this frame should be forwarded to a receiver based on what decode targets the receiver is expecting.
    - `Map <unsigned long, unsigned long> chainLinks`
      - Used to ensure we preserve decode order for the desired decode target. It is insufficient to satisfy the dependencies for the current frame.
  - Proposal: submit PR to harmonize SVC metadata between Encoded Transform and WebCodecs

Media Pipeline Architecture Repo

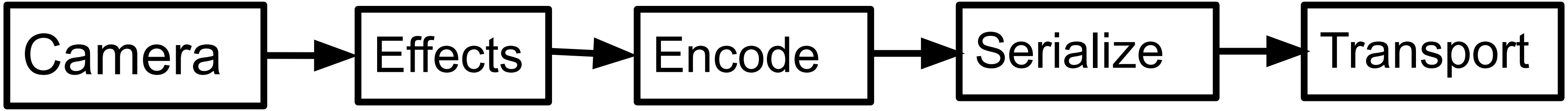
- Established based on conversation at TPAC joint meeting.
- A repository of issues and pointers to sample code covering integration of "Next Generation Web Media APIs"
- Goals:
  - To understand what "seams" and inconsistencies exist between the APIs
  - To provide some insight for new media transport designs
    - RTP over QUIC/WebTransport
  - To understand how well the APIs perform.
    - If there are issues, is it a problem in the spec, the implementation, or the sample code?
- Non-goals:
  - Finding issues in individual specs (file those in the appropriate repos)
  - Finding or mixing implementation bugs (file a browser bug)

Next generation Web media APIs

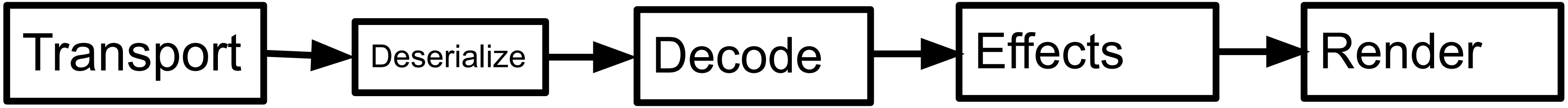
- Capture
  - Media Capture and Streams Extensions
  - Mediacapture-transform
- Encode/decode
  - WebCodecs
  - MSEv2
- Transport
  - WebTransport (HTTP/3 over QUIC)
  - WebRTC data channel in Workers (SCTP/DTLS/UDP)
- Framework
  - WHATWG Streams
  - Web Assembly

The “Pipeline” Model (WHATWG Streams)

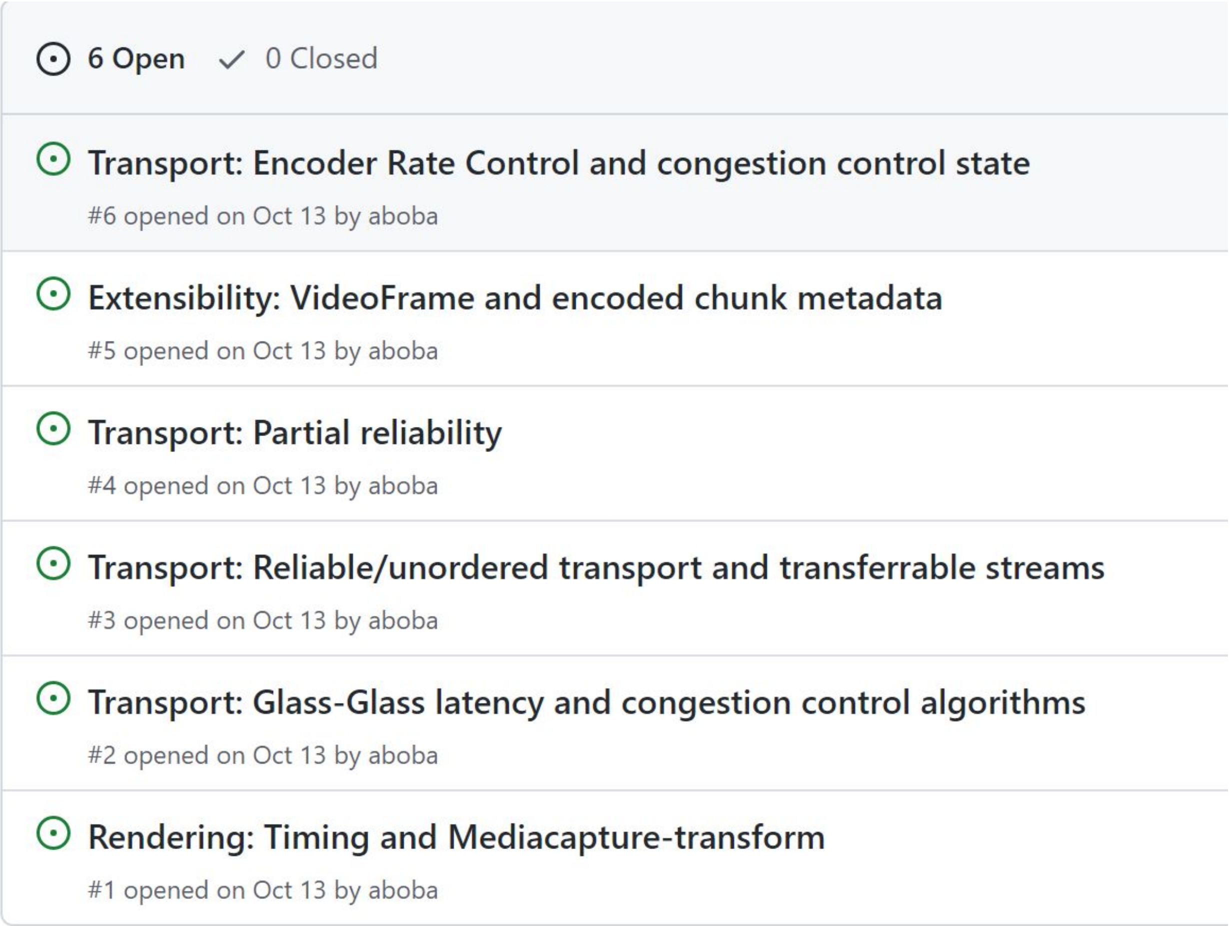
- Send



- Receive



Media Pipeline Architecture Issues



⊙ 6 Open ✓ 0 Closed

⊙ Transport: Encoder Rate Control and congestion control state
#6 opened on Oct 13 by aboba

⊙ Extensibility: VideoFrame and encoded chunk metadata
#5 opened on Oct 13 by aboba

⊙ Transport: Partial reliability
#4 opened on Oct 13 by aboba

⊙ Transport: Reliable/unordered transport and transferrable streams
#3 opened on Oct 13 by aboba

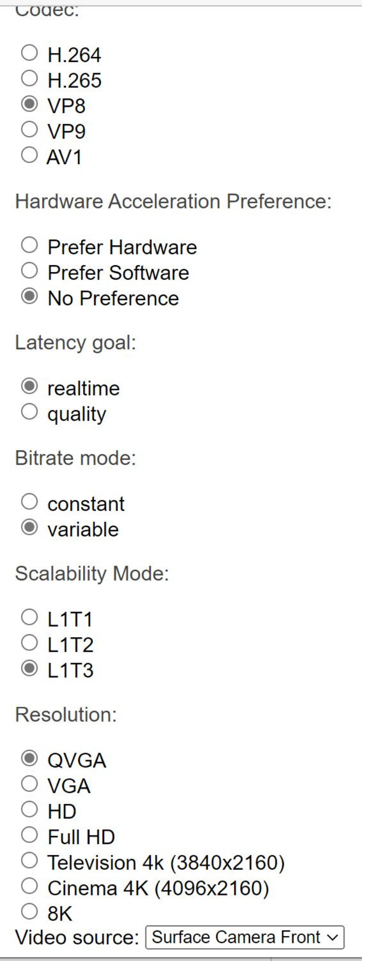
⊙ Transport: Glass-Glass latency and congestion control algorithms
#2 opened on Oct 13 by aboba

⊙ Rendering: Timing and Mediacapture-transform
#1 opened on Oct 13 by aboba

Media Pipeline Architecture Samples

- WHATWG Streams Samples:
  - PR 583: WebCodecs Encode/Decode in worker
    - Supports WebCodecs codecs and configuration knobs
    - Live site
  - PR 430: WebCodecs-WebTransport Echo in worker
    - Live site
    - Adds Serialization/Deserialization and WebTransport send/receive to PR 583.
    - Uses frame/stream transport
    - “RTP-ish” frame format
    - Supports SVC, partial reliability
    - Implements a re-ordering buffer but not a full jitter buffer (yet)

Parameters to Select



Codec:

○ H.264
○ H.265
● VP8
○ VP9
○ AV1

Hardware Acceleration Preference:

○ Prefer Hardware
○ Prefer Software
● No Preference

Latency goal:

● realtime
○ quality

Bitrate mode:

○ constant
● variable

Scalability Mode:

○ L1T1
○ L1T2
● L1T3

Resolution:

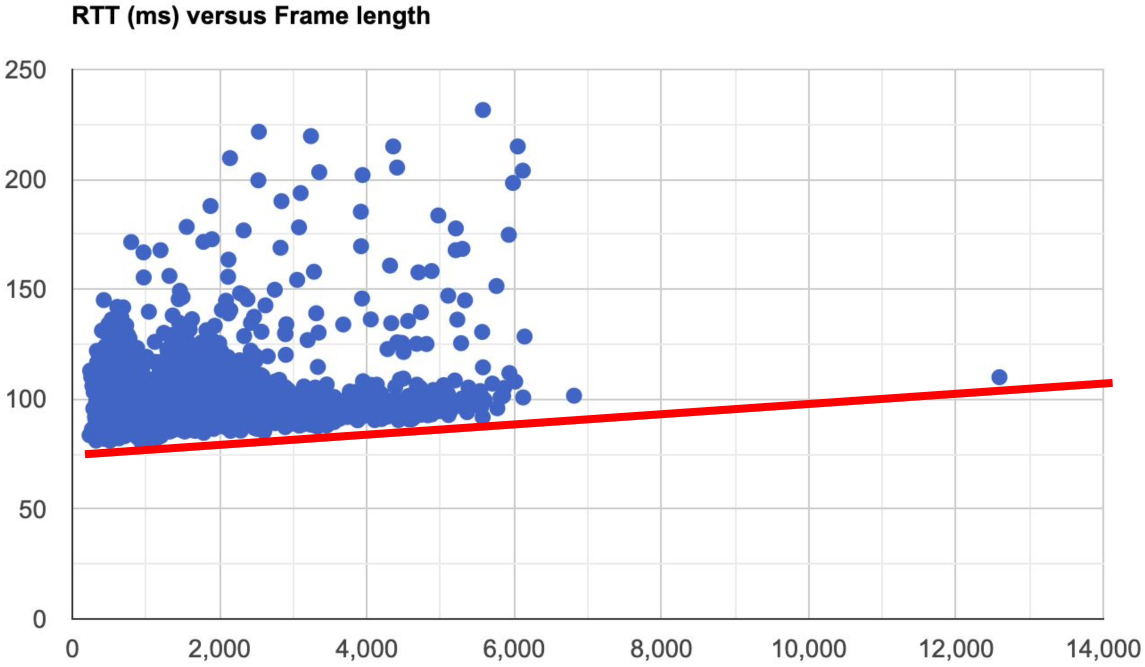
● QVGA
○ VGA
○ HD
○ Full HD
○ Television 4k (3840x2160)
○ Cinema 4K (4096x2160)
○ 8K

Video source: Surface Camera Front

- Bitrate: “Average Target Bitrate” target provided to the encoder.
- Keyframe interval: number of frames between each keyframe.
- Codec: VP8, VP9, H.264 or AV1
  - Some oddities noted with VP9 (large frame size with “realtime”)
  - AV1 most solid on MacOS
  - H.265 not supported currently.
- Hardware Acceleration Preference: hw accelerated versus software encode/decode. Hw acceleration often not available.
- Latency goal: “quality” produces smaller frame sizes, but takes (marginally) longer than “realtime”.
- Bitrate mode: constant or variable bitrate
- Scalability mode: how many temporal layers to use. Enables differential protection for the base layer.
- Resolution: reflected in getUserMedia constraints. If your camera doesn’t support the requested resolution, window will be blacked out.

Frame RTT Graph

- AV1 @ full-Hd with 418 Kbps average bitrate and 30 fps, GoP = 3000, L1T3 scalability mode
- Largest (I-)frame = 12590 octets, median (P-)frame size = 1523 octets
- I-frame is close to the transmission line, indicating that cwind > 12590.



BWE report:

{"count":2283,"loss":0,"reorder":6,"bwe":0,"bwu":417956.483387237,"seqmin":0,"seqmax":2282,"lenmin":234,"lenfquart":727,"lenmedian":1523,"lentquart":2161,"lenmax":12590,"recvsum":3980116}

RTT report:

{"count":2283,"min":80.299,"fquart":92.9,"avg":101.6191081909768,"median":98.1,"tquart":105.399,"max":231.6,"stdev":15.421836998138598,"srtt":115.60558227812218,"rttvar":7.499192348045117,"rto":145.60235167030265}