Recording a Jitsi Meet conference

Master 2
Réseaux Informatiques et Systèmes Embarqués

Boris Grozev

Supervisor:

Dr. Emil Ivov

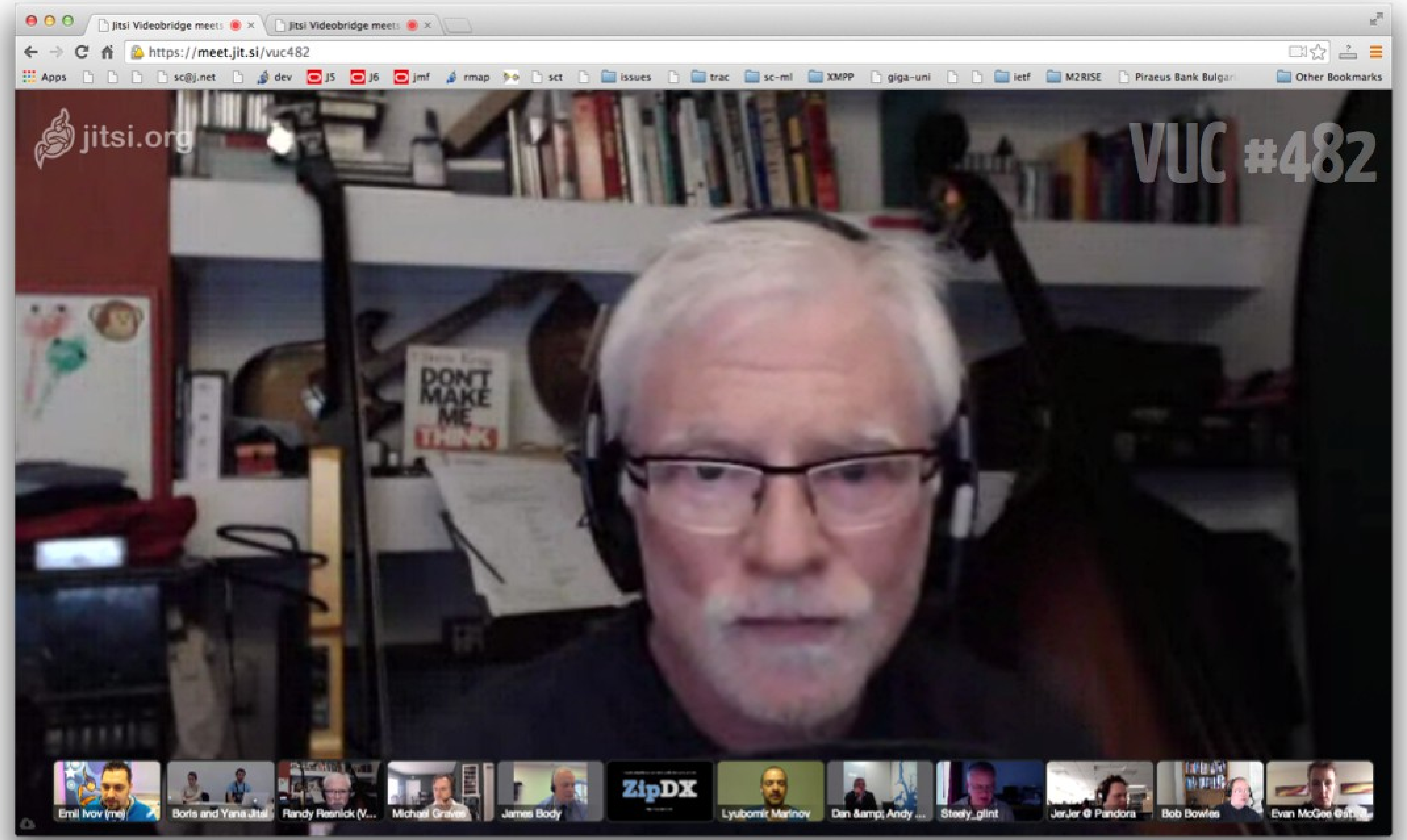
UNIVERSITÉ DE STRASBOURG

June 24, 2014

blue jimp®

- Development services
  - Based around Jitsi
    - Opensource
    - VoIP
- My role:
  - Software developer
  - Recording a video conference

1.2 Jitsi Meet



Images: Emil Ivov

2 Recording (1/3)

We have :

A set of streams

We need :

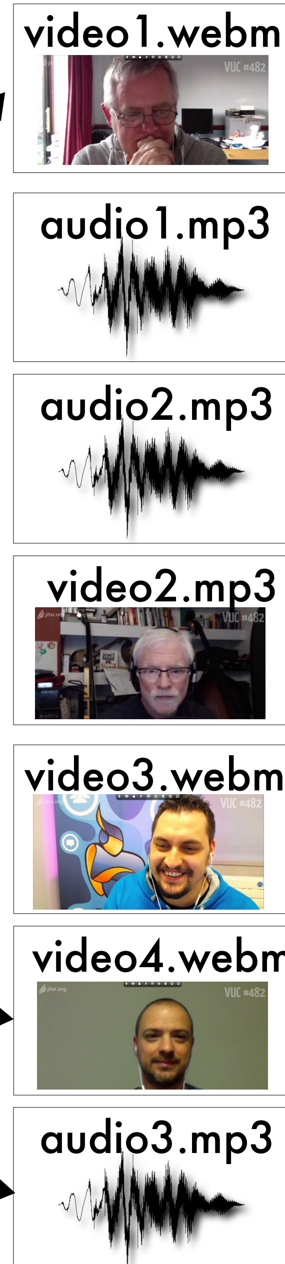
A single file

Images: Emil Ivov

2 Recording (2/3)

Stage I:

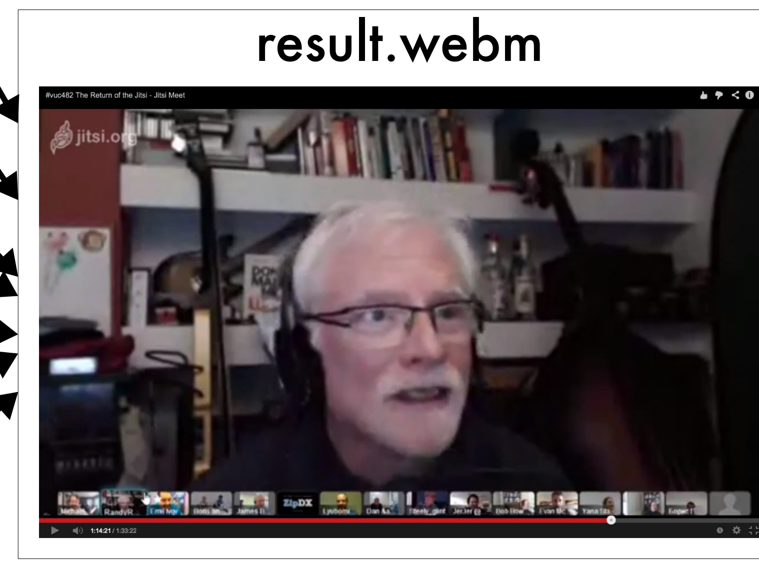
Demultiplex streams,
Depacketize,
Put in container,
Save to disk



Stage II:

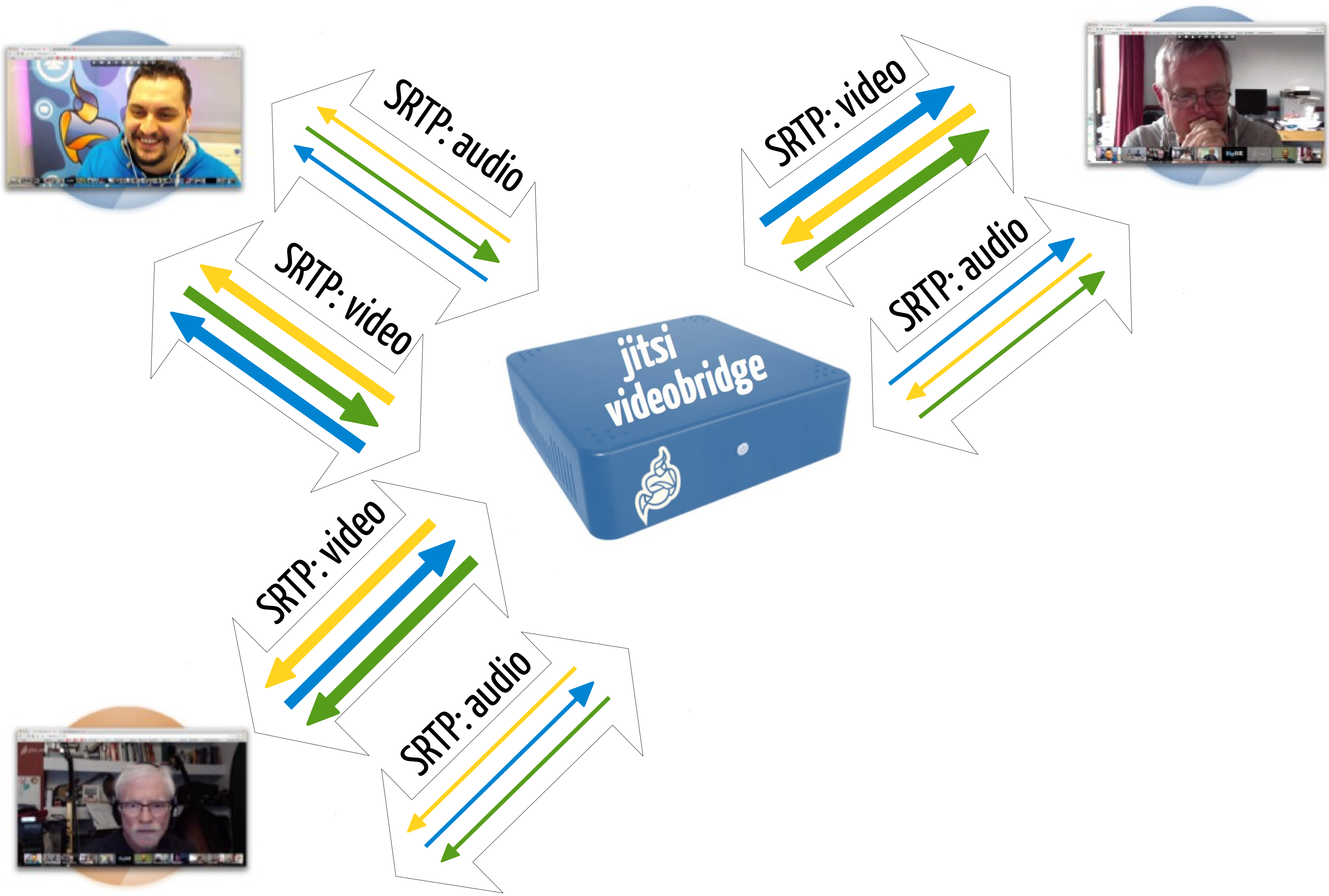
(Post-processing)

Mix audio,
Decode/Overlay/Encode video,
Mux audio and video



Images: Emil Ivov

2 Recording (3/3)



Images: Emil Ivov

The plan:

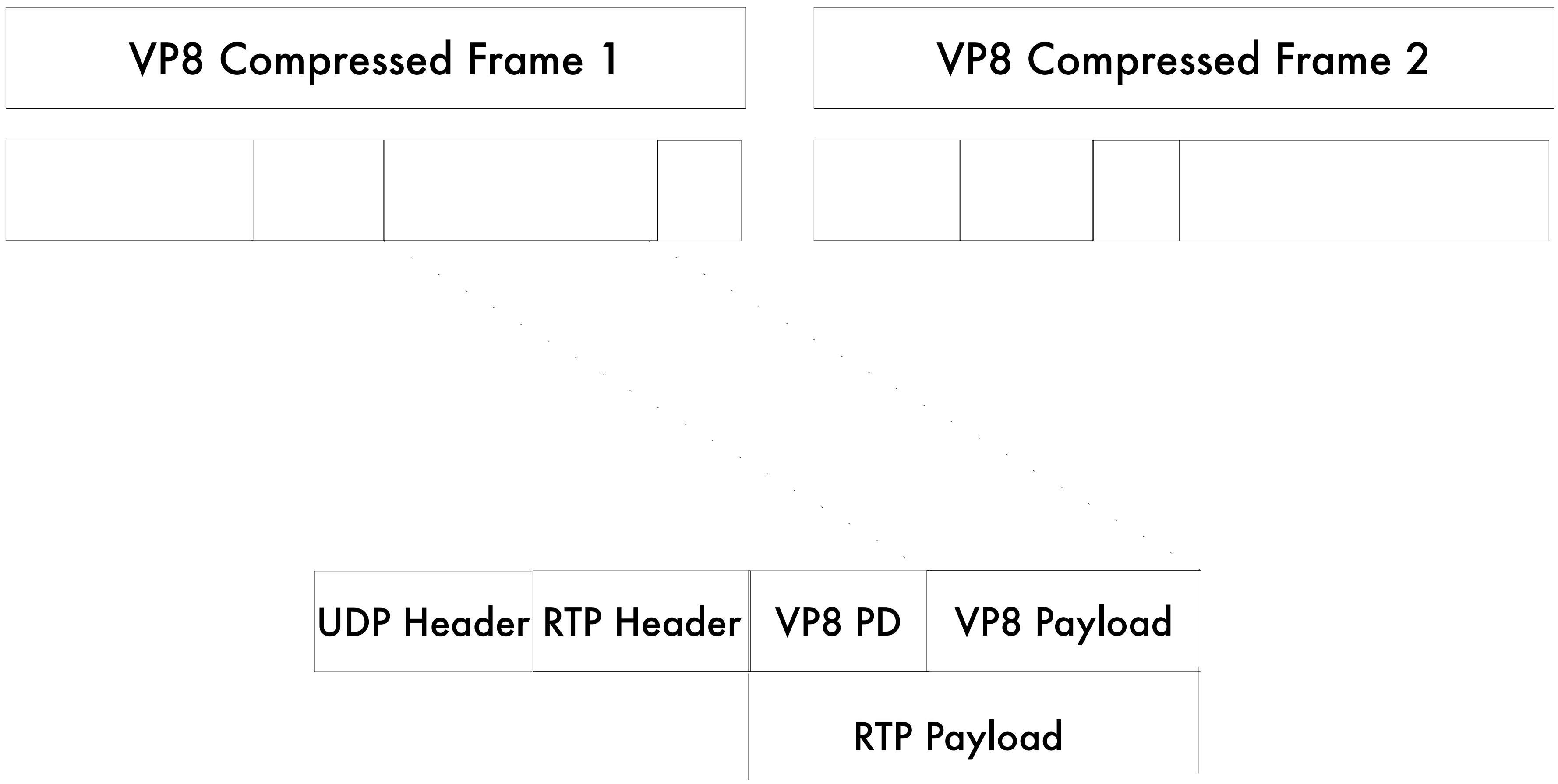
- Video
- Audio
- Metadata
- Synchronization

Images: Emil Ivov

3.1 Video: VP8

- RFC6386
- Libvpx – opensource implementation
- Wide support
  - Most browsers
  - Many players (ffmpeg, gstreamer)

3.2 Video: packetization



Packetization:

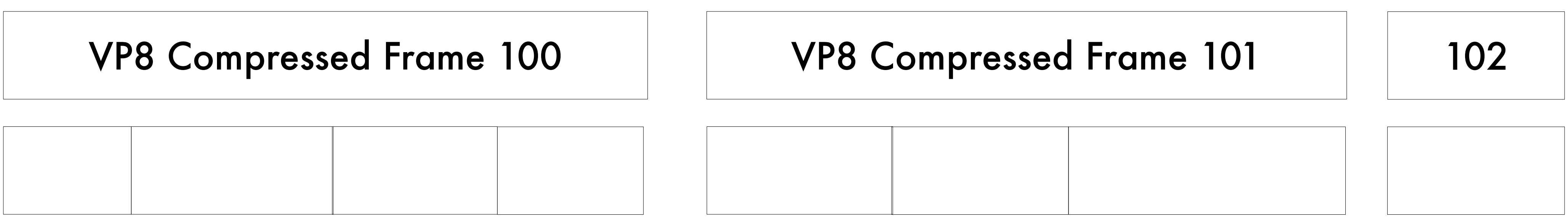
1. Split
2. Add headers: RTP + VP8 Payload Descriptor (PD)

RTP Header

```
 0                   1                   2                   3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|V=2|P|X|  CC   |M|     PT      |       sequence number         |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                           timestamp                           |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|           synchronization source (SSRC) identifier            |
+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+
|            contributing source (CSRC) identifiers             |
|                             ....                              |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

VP8 PD

```
       0 1 2 3 4 5 6 7
      +-+-+-+-+-+-+-+-+
      |X|R|N|S|R| PID |
      +-+-+-+-+-+-+-+-+
X:    |I|L|T|K| RSV   |
      +-+-+-+-+-+-+-+-+
I:    |M| PictureID   |
      +-+-+-+-+-+-+-+-+
L:    |   TL0PICIDX   |
      +-+-+-+-+-+-+-+-+
T/K:  |TID|Y| KEYIDX  |
      +-+-+-+-+-+-+-+-+
```



3.4 Video: webm

webm

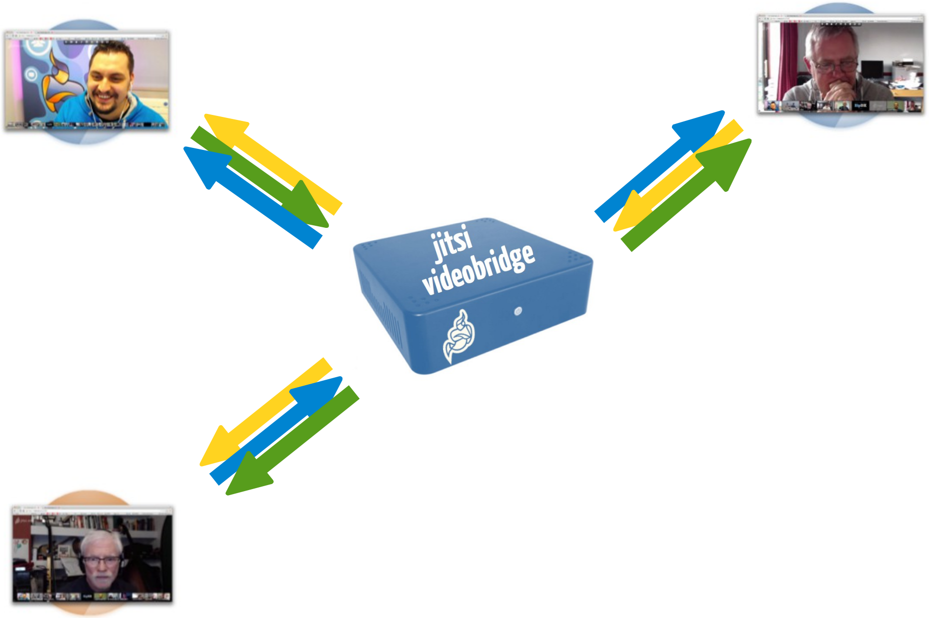
- Container format for audio and video
- Open (based on matroska)
- Designed for the web (for web-browsers)
- Our use:
  - Video only (VP8)
  - Single track: a sequence of frames

3.5 Video: RTX and FEC (1/2)

Handling packet loss:
Retransmissions (RTX) & Forward Error Correction (FEC)

RTX:

- Triggered by the clients
  - RTCP NACK
- Require at least a RTT
- How do we handle it?
  - Just keep a big buffer

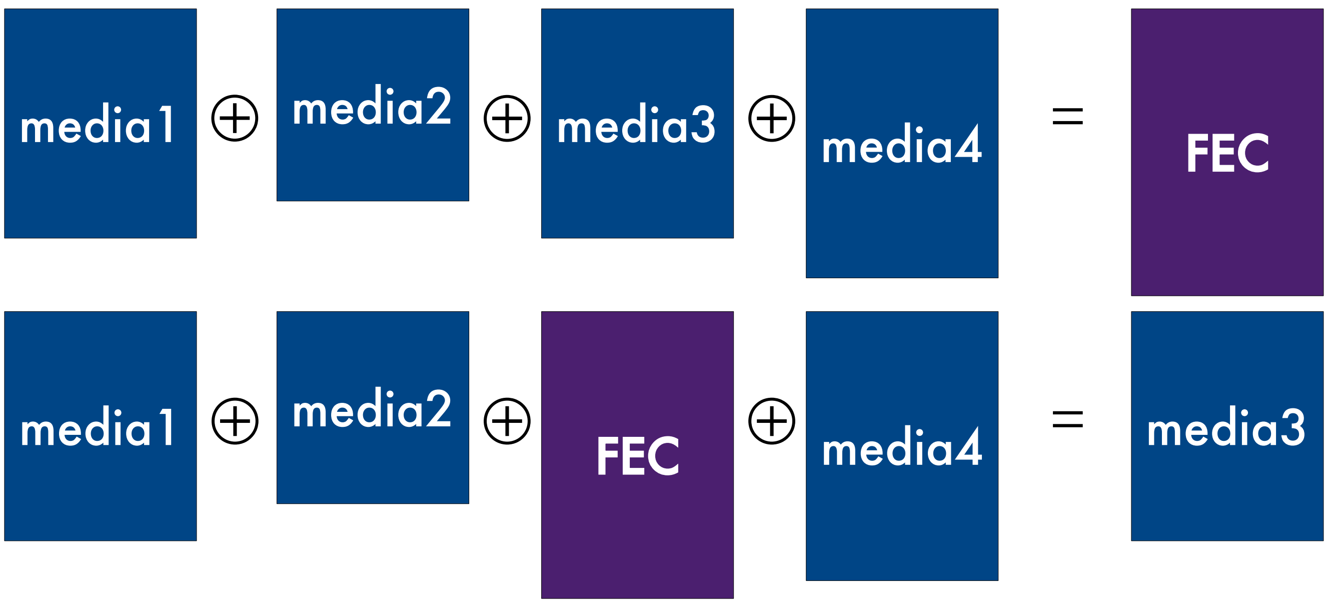


Images: Emil Ivov

3.6 Video: RTX and FEC (2/2)

Forward Error Correction (FEC, RFC5109):

- Add redundancy data
- Based on parity (similar to RAID5)



- Not triggered by the receiver
- No RTT delay
- Requires special handling

3.1 Audio: to mix or not to mix

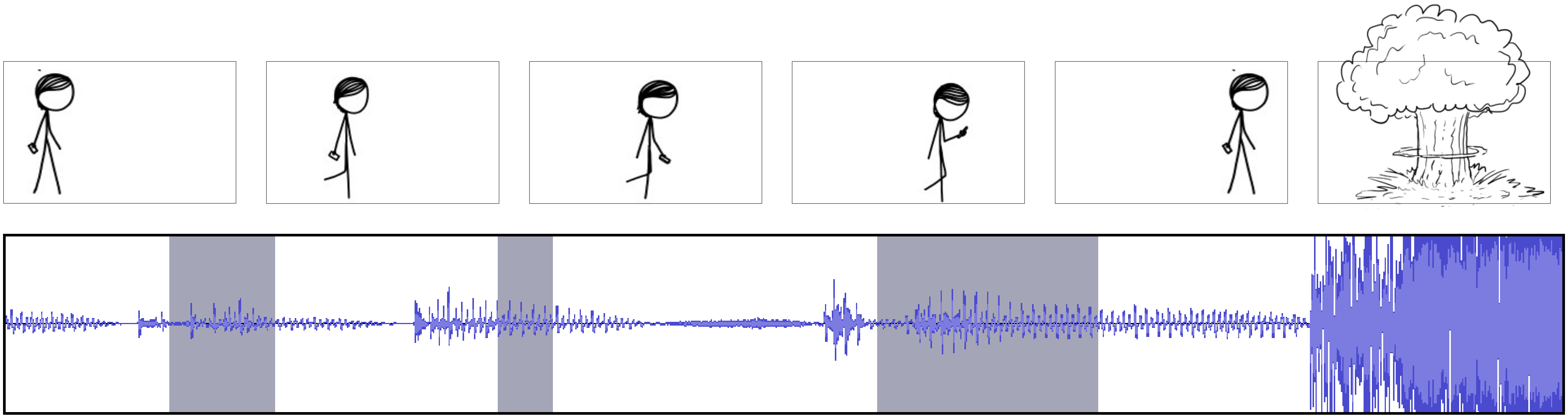
Record a mix?

\+ Already available in libjitsi

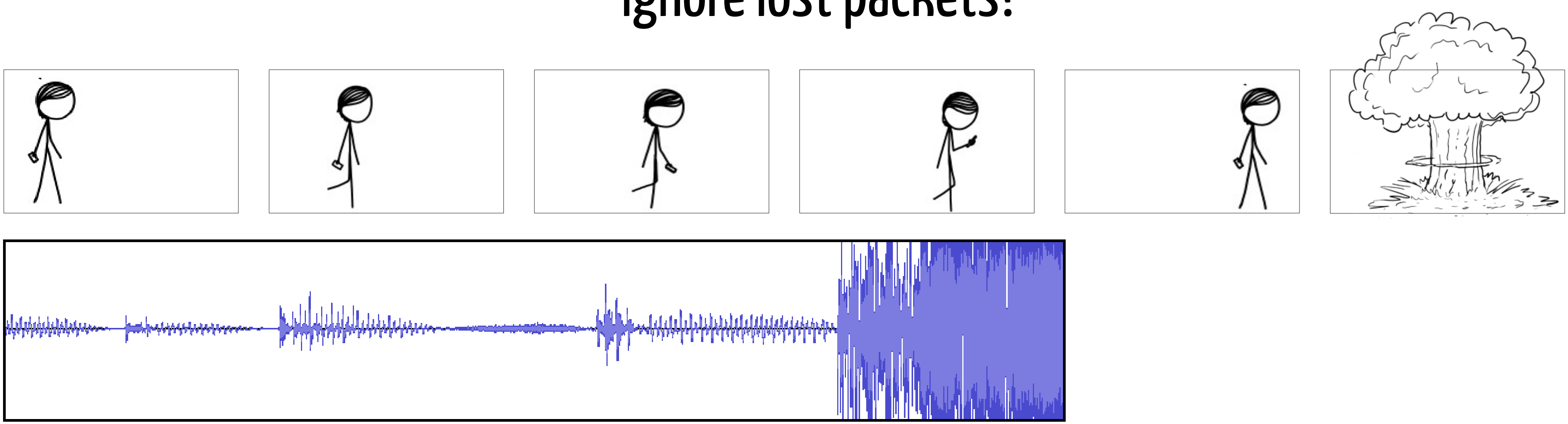
\- More expensive (CPU-wise)

\- No fine control over the streams

3.2 Audio: gaps

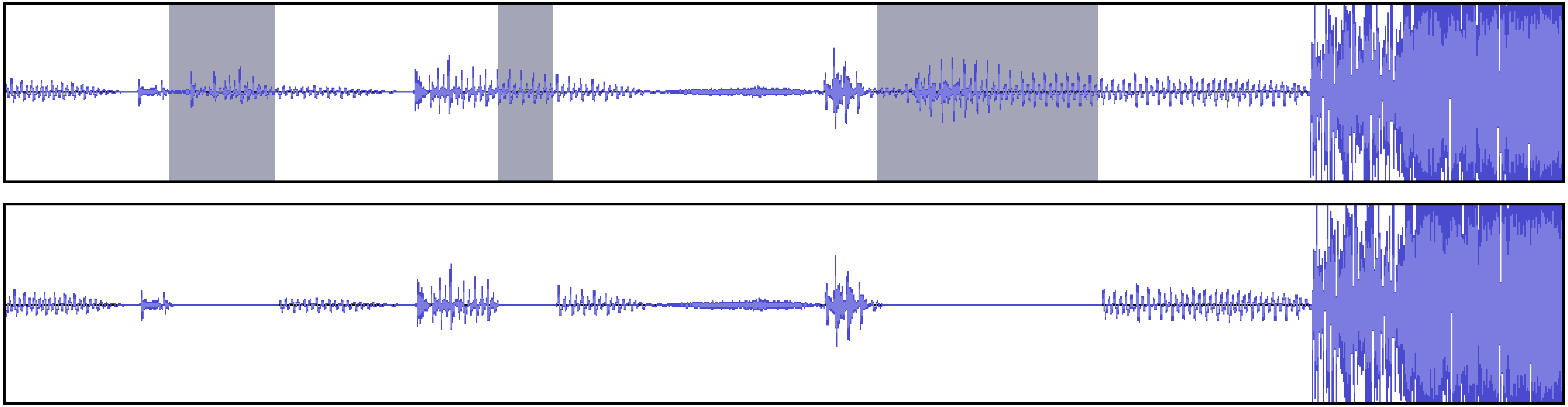


Ignore lost packets?

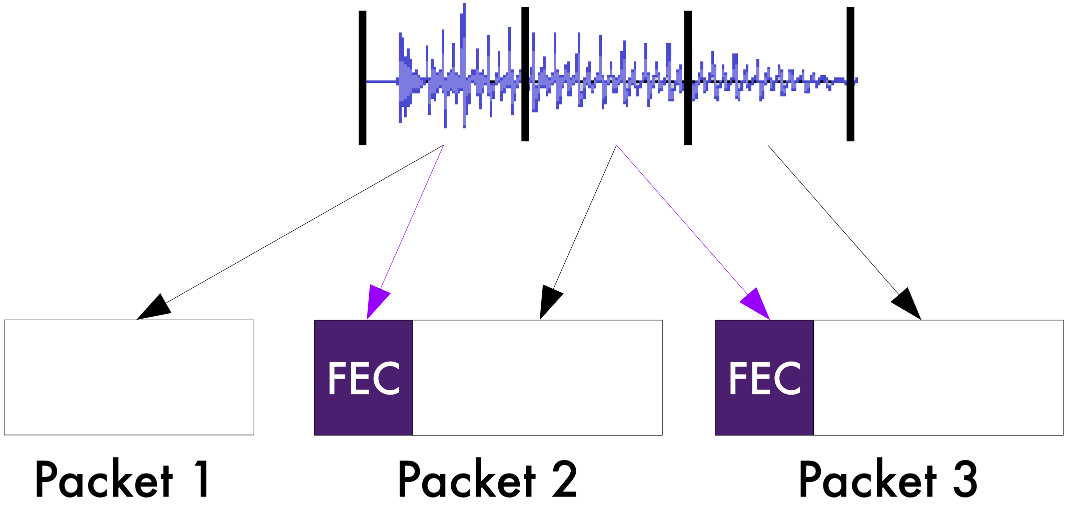


Images: xkcd

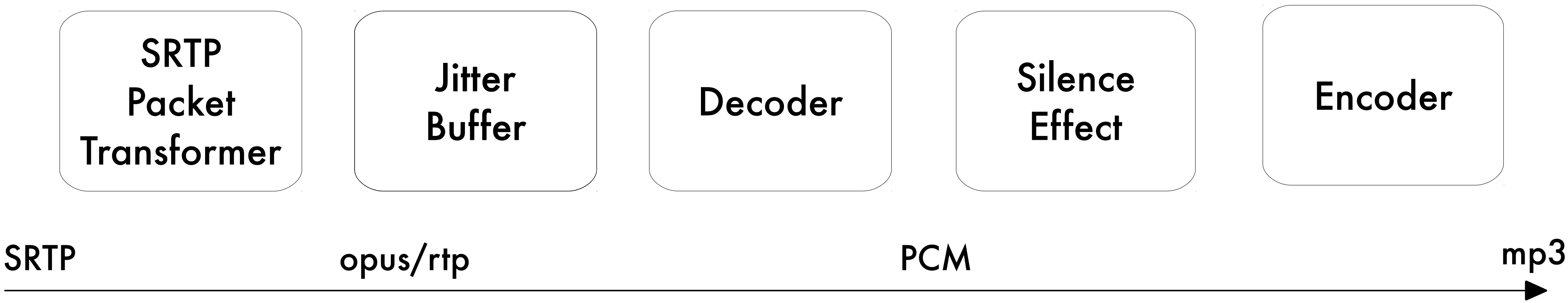
3.3 Audio: repairing gaps



- Opus' FEC – single packet (20ms), based on actual sound
- Opus' PLC – attempts to conceal packet lost (no crackling)
- Actual silence – zeros



3.4 Audio: the current scheme



Decrypt

FEC

Fill gaps (<3sec)
(zeros)

PLC
(up to 3 packets)

Reset if gap >3sec

4. Metadata

- Used in post-processing
- Indicate **WHICH** files **WHEN**
- Indicate speaker changes
- Additional information
  - Participant names
- JSON format
- Sequence of events


```
{
"instant" : 0,
"type" : "RECORDING_STARTED",
"filename" : "video1.webm",
"ssrc" : 3141592654,
"mediaType" : "video",
"participantName" : "Jane Doe",
"participantDescription" : "Chief
Example Officer"
}
{
"instant" : 1500,
"type" : "RECORDING_STARTED",
"filename" : "audio1.mp3",
"ssrc" : 271828182,
"mediaType" : "audio",
}
{
"instant" : 6500,
"type" : "SPEAKER_CHANGED",
"ssrc" : 3141592654,
"audioSsrc" : 271828182
}
```

5. Synchronization (1/2)

1. Purpose:
   - All streams mixed with the correct timing
   - Generate the correct event "instants" in metadata
2. Synchronization between participants:
   - We cannot assure it
   - Based on time of arrival of packets
3. Between the streams of a single participant (audio/video sync)
   - Use timestamps coming from the media source
     - Removes the effects of jitter

5. Synchronization (2/2)

For streams from the same source:

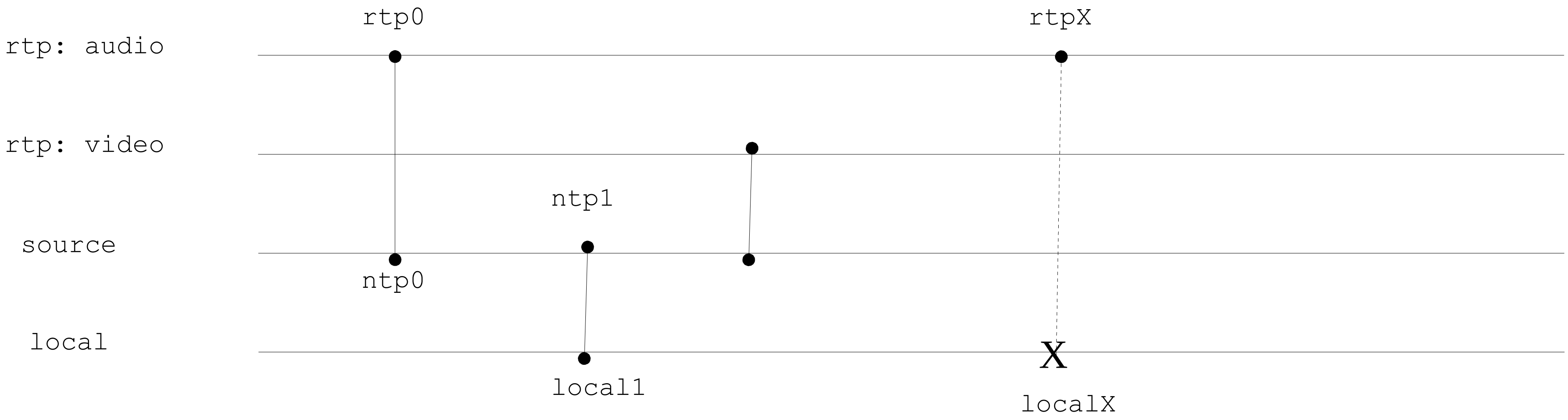
Clocks:

- RTP clock for each SSRC (rtp0)
- Wallclock for each source (ntp0)
- Local system clock (local1)

Mappings:

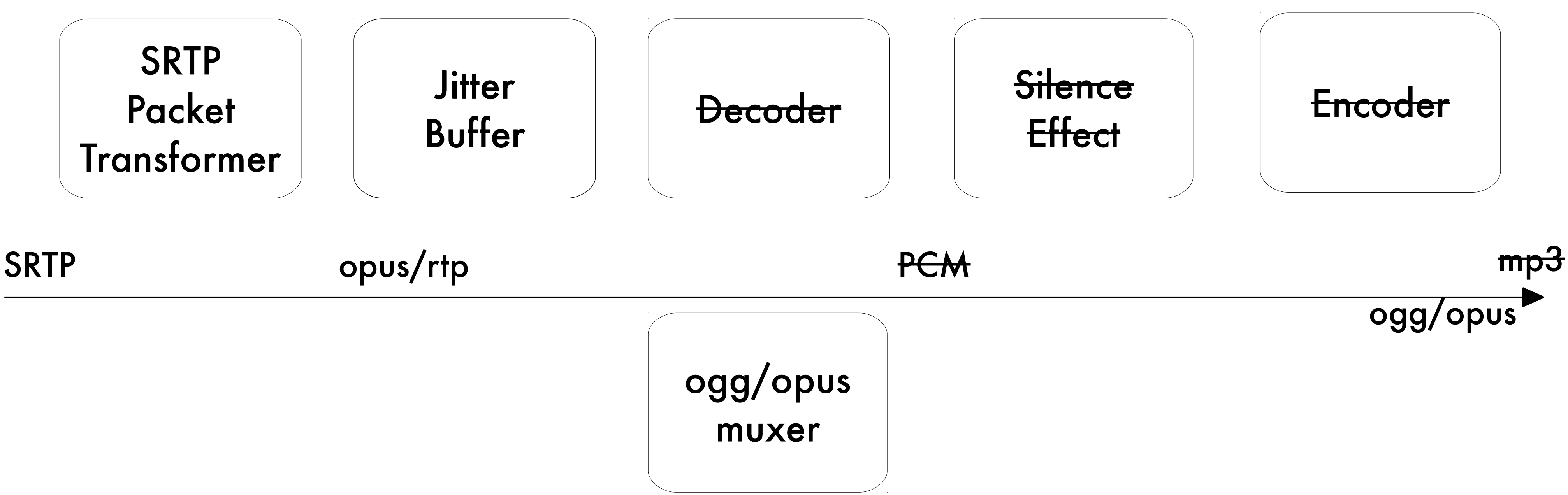
- RTP-to-Source for each SSRC (rtp0 – ntp0)
- Source-to-local for each source (ntp1 – local1)

```
localX = local1 + (ntp0 - ntp1) + (rtpX - rtp0)
```



6. Future work

- Improving RTX support
  - Requesting via RTCP NACK
- Recording in ogg/opus format
  - Avoids re-encoding
  - More efficient
  - Preserve sound quality
  - Problem: FEC



7. Demo

```
if (time_left > 0)
   demo();
else if (time_left)
   oops();
```

Thank you!