

# *Can SFUs and MCUs be friends?*

Lorenzo Miniero

 [@elminiero](#)

IIT Real-Time Communication 2020 – WebRTC Track  
October 14<sup>th</sup> 2020, Chicago, IL, USA



## A few words about me



### Lorenzo Miniero

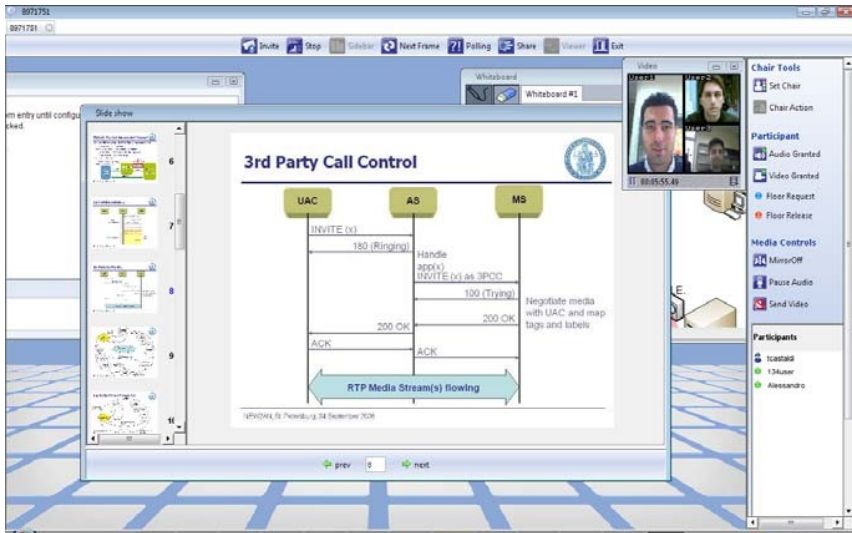
- Ph.D @ UniNA
- Chairman @ Meetecho
- Main author of Janus<sup>®</sup>

### Contacts and info

- [lorenzo@meetecho.com](mailto:lorenzo@meetecho.com)
- <https://twitter.com/elminiero>
- <https://www.slideshare.net/LorenzoMiniero>
- <https://soundcloud.com/lminiero>

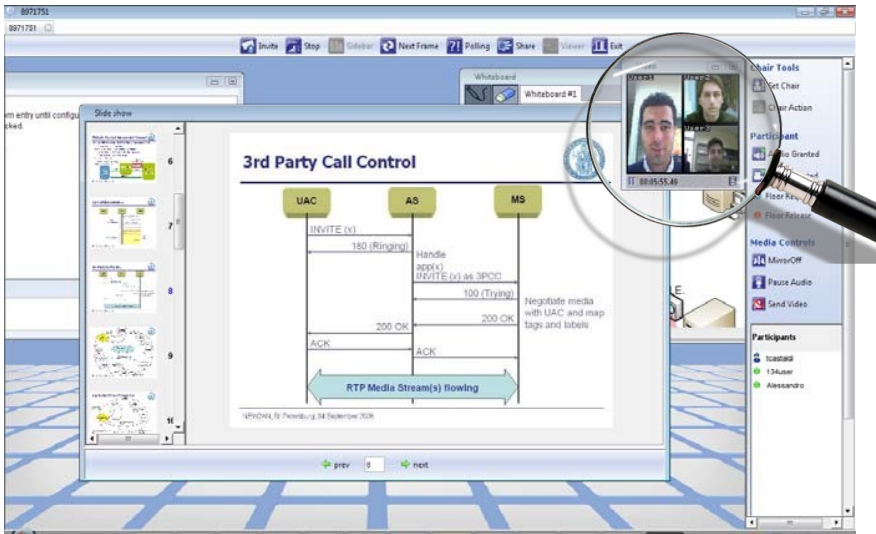


Fun fact: the first RTC server I ever wrote was an MCU 😊



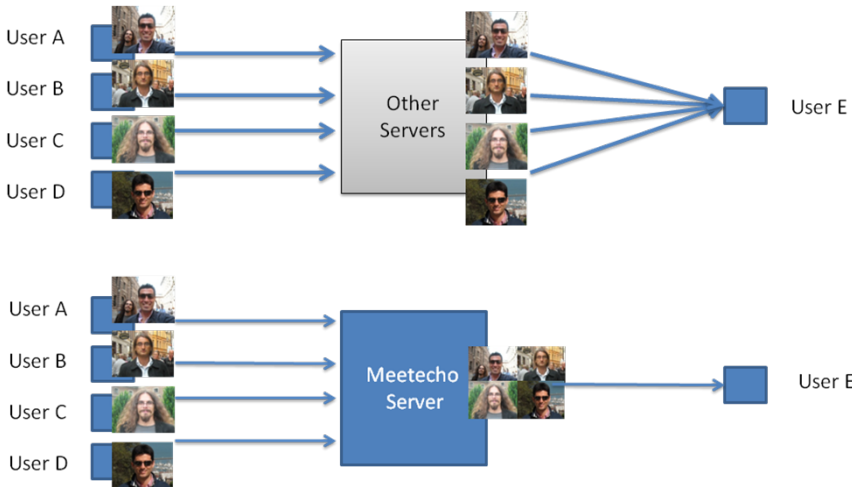


Fun fact: the first RTC server I ever wrote was an MCU 😊



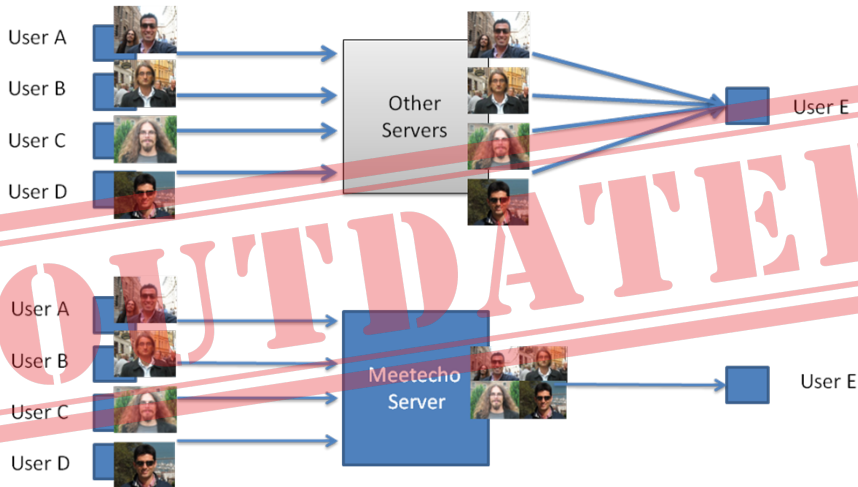


Fun fact: the first RTC server I ever wrote was an MCU 😊



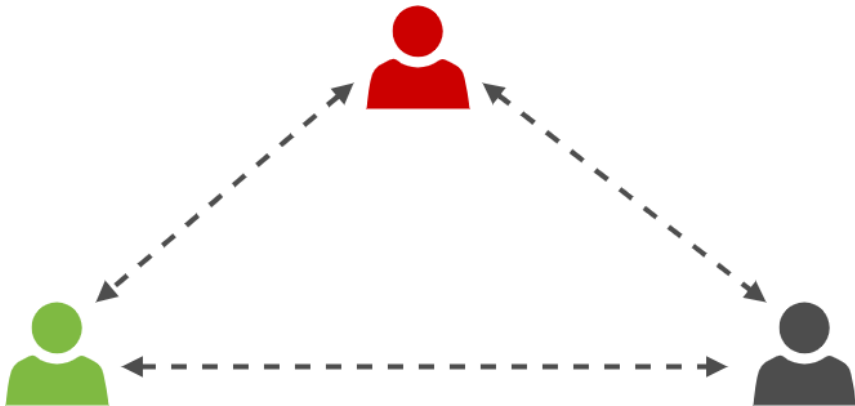


Fun fact: the first RTC server I ever wrote was an MCU 😊





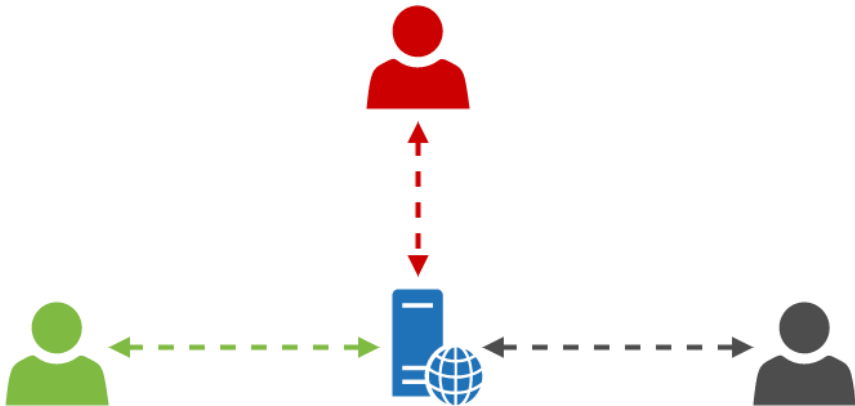
## WebRTC topologies: full-mesh



<https://webrtcchacks.com/webrtc-beyond-one-one/>



## WebRTC topologies: MCU (Multipoint Control Unit)

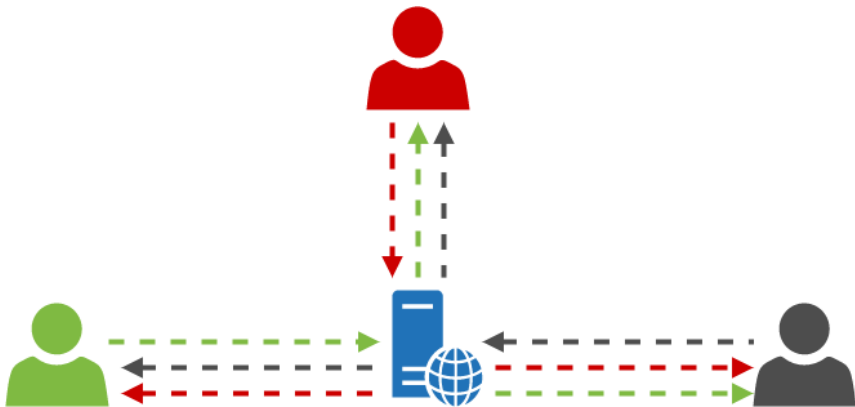


<https://webrtcchacks.com/webrtc-beyond-one-one/>





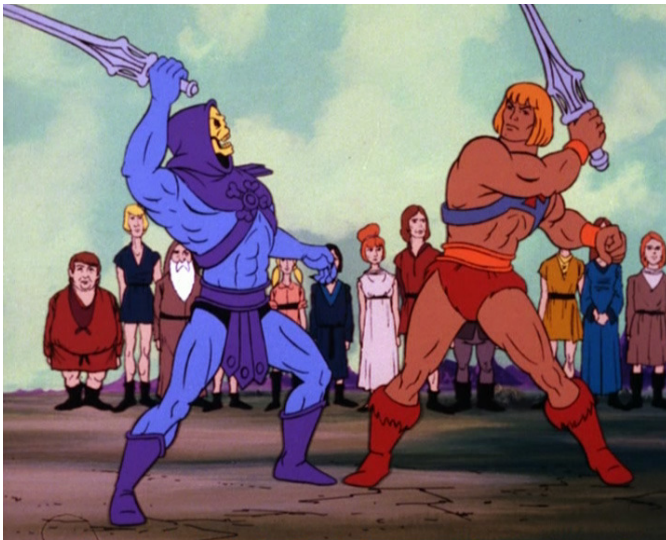
## WebRTC topologies: SFU (Selective Forwarding Unit)



<https://webrtcchacks.com/webrtc-beyond-one-one/>



How most “SFU vs. MCU” discussions look like





## A few words on MCUs

- Been around for a long time (e.g., legacy SIP/H.323 conferencing systems)
- To simplify, mixes multiple streams into one
  - Multiple participants send their streams to the MCU
  - MCU decodes each stream and mixes/composes them
  - Participants get a single encoded stream back

### Some Pros

- Limited/consistent bandwidth usage
- Good for constrained endpoints
- Can compose/show many streams
- Endpoints can use different codecs
- Easier “legacy” interoperability

### Some Cons

- CPU heavy on server (expensive)
- Needs smarter RTP buffering
- Can't *really* use Simulcast/SVC
- Little flexibility on UI
- Can't do end-to-end encryption



## A few words on MCUs

- Been around for a long time (e.g., legacy SIP/H.323 conferencing systems)
- To simplify, mixes multiple streams into one
  - Multiple participants send their streams to the MCU
  - MCU decodes each stream and mixes/composes them
  - Participants get a single encoded stream back

### Some Pros

- Limited/consistent bandwidth usage
- Good for constrained endpoints
- Can compose/show many streams
- Endpoints can use different codecs
- Easier “legacy” interoperability

### Some Cons

- CPU heavy on server (expensive)
- Needs smarter RTP buffering
- Can't *really* use Simulcast/SVC
- Little flexibility on UI
- Can't do end-to-end encryption



## A few words on MCUs

- Been around for a long time (e.g., legacy SIP/H.323 conferencing systems)
- To simplify, mixes multiple streams into one
  - Multiple participants send their streams to the MCU
  - MCU decodes each stream and mixes/composes them
  - Participants get a single encoded stream back

### Some Pros

- Limited/consistent bandwidth usage
- Good for constrained endpoints
- Can compose/show many streams
- Endpoints can use different codecs
- Easier “legacy” interoperability

### Some Cons

- CPU heavy on server (expensive)
- Needs smarter RTP buffering
- Can't *really* use Simulcast/SVC
- Little flexibility on UI
- Can't do end-to-end encryption



## A few words on MCUs

- Been around for a long time (e.g., legacy SIP/H.323 conferencing systems)
- To simplify, mixes multiple streams into one
  - Multiple participants send their streams to the MCU
  - MCU decodes each stream and mixes/composes them
  - Participants get a single encoded stream back

### Some Pros

- Limited/consistent bandwidth usage
- Good for constrained endpoints
- Can compose/show many streams
- Endpoints can use different codecs
- Easier “legacy” interoperability

### Some Cons

- CPU heavy on server (expensive)
- Needs smarter RTP buffering
- Can't *really* use Simulcast/SVC
- Little flexibility on UI
- Can't do end-to-end encryption



## A few words on SFUs

- Relatively recent, and more widespread thanks to WebRTC
- No mixing performed, packets are just relayed
  - Multiple participants send their streams to the SFU
  - Streams forwarded separately (and optionally) to other subscribers
  - No transcoding is performed on the media path

### Some Pros

- Much more lightweight on CPU
- Feedback between users preserved
- Can take advantage of Simulcast/SVC
- Allows for end-to-end encryption
- Very flexible on UI side

### Some Cons

- Higher bandwidth usage (expensive)
- Clients have to decode/render a lot
- They also need to support same codecs
- Harder to integrate with legacy systems



## A few words on SFUs

- Relatively recent, and more widespread thanks to WebRTC
- No mixing performed, packets are just relayed
  - Multiple participants send their streams to the SFU
  - Streams forwarded separately (and optionally) to other subscribers
  - No transcoding is performed on the media path

### Some Pros

- Much more lightweight on CPU
- Feedback between users preserved
- Can take advantage of Simulcast/SVC
- Allows for end-to-end encryption
- Very flexible on UI side

### Some Cons

- Higher bandwidth usage (expensive)
- Clients have to decode/render a lot
- They also need to support same codecs
- Harder to integrate with legacy systems





## A few words on SFUs

- Relatively recent, and more widespread thanks to WebRTC
- No mixing performed, packets are just relayed
  - Multiple participants send their streams to the SFU
  - Streams forwarded separately (and optionally) to other subscribers
  - No transcoding is performed on the media path

### Some Pros

- Much more lightweight on CPU
- Feedback between users preserved
- Can take advantage of Simulcast/SVC
- Allows for end-to-end encryption
- Very flexible on UI side

### Some Cons

- Higher bandwidth usage (expensive)
- Clients have to decode/render a lot
- They also need to support same codecs
- Harder to integrate with legacy systems



## A few words on SFUs

- Relatively recent, and more widespread thanks to WebRTC
- No mixing performed, packets are just relayed
  - Multiple participants send their streams to the SFU
  - Streams forwarded separately (and optionally) to other subscribers
  - No transcoding is performed on the media path

### Some Pros

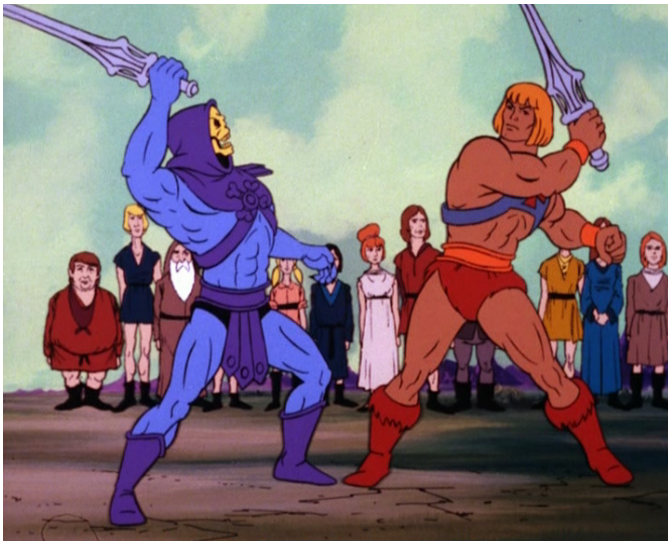
- Much more lightweight on CPU
- Feedback between users preserved
- Can take advantage of Simulcast/SVC
- Allows for end-to-end encryption
- Very flexible on UI side

### Some Cons

- Higher bandwidth usage (expensive)
- Clients have to decode/render a lot
- They also need to support same codecs
- Harder to integrate with legacy systems



So they're indeed very different...





But... can't SFUs and MCUs be friends?





## Audio MCU + Video SFU

- What we usually call the “hybrid approach”
  - Audio from participants is mixed (MCU)
  - Video from participants is relayed (SFU)
- Mixing only audio has a few advantages
  - 1 Audiomixing lighter than Videomixing, so relative impact
  - 2 Participants with constraints (CPU/BW) can stick to audio only
  - 3 In case audio is SIP-based, easy to hook to PSTN
  - 4 Decoupling audio from video allows for more options (e.g., interpreters)
  - 5 Easy to distribute/broadcast (audio already mixed)
- SFU mode keeps flexibility for video
  - Easy to subscribe to a subset of participants, or none at all



## Audio MCU + Video SFU

- What we usually call the “hybrid approach”
  - Audio from participants is mixed (MCU)
  - Video from participants is relayed (SFU)
- Mixing only audio has a few advantages
  - 1 Audiomixing lighter than Videomixing, so relative impact
  - 2 Participants with constraints (CPU/BW) can stick to audio only
  - 3 In case audio is SIP-based, easy to hook to PSTN
  - 4 Decoupling audio from video allows for more options (e.g., interpreters)
  - 5 Easy to distribute/broadcast (audio already mixed)
- SFU mode keeps flexibility for video
  - Easy to subscribe to a subset of participants, or none at all

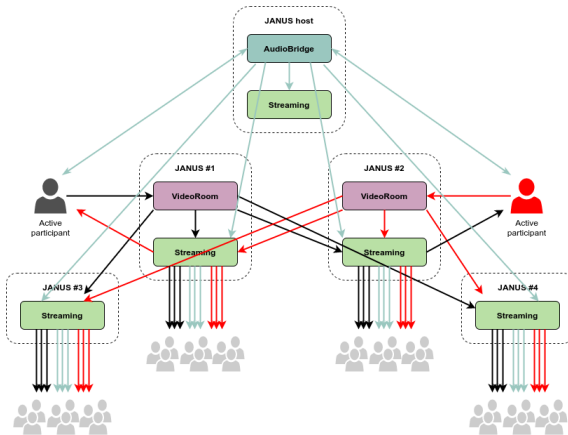


## Audio MCU + Video SFU

- What we usually call the “hybrid approach”
  - Audio from participants is mixed (MCU)
  - Video from participants is relayed (SFU)
- Mixing only audio has a few advantages
  - ① Audiomixing lighter than Videomixing, so relative impact
  - ② Participants with constraints (CPU/BW) can stick to audio only
  - ③ In case audio is SIP-based, easy to hook to PSTN
  - ④ Decoupling audio from video allows for more options (e.g., interpreters)
  - ⑤ Easy to distribute/broadcast (audio already mixed)
- SFU mode keeps flexibility for video
  - Easy to subscribe to a subset of participants, or none at all



# Audio MCU + Video SFU

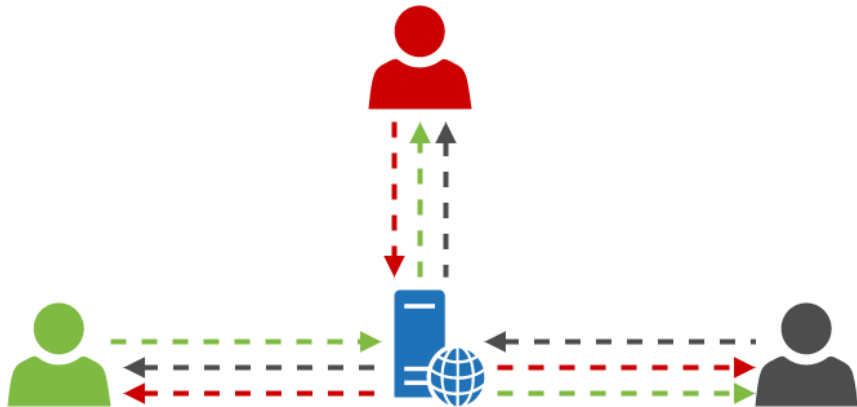


<https://commcon.xyz/session/turning-live-events-to-virtual-with-janus>



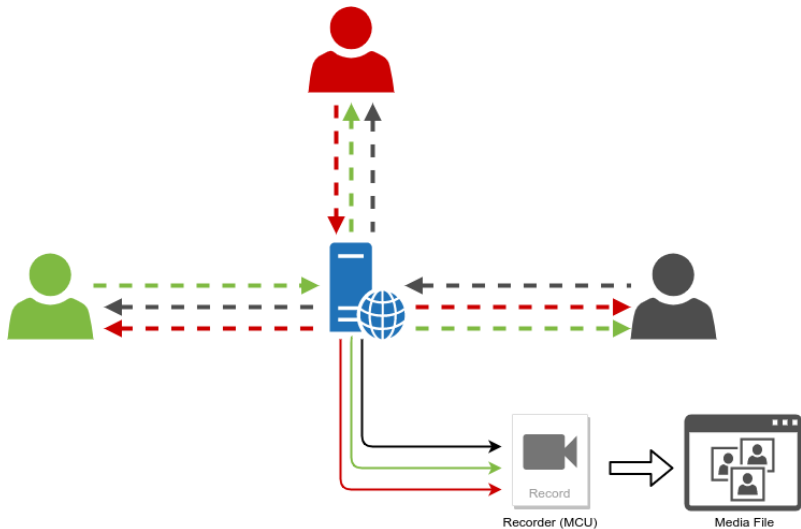


## Live Recording



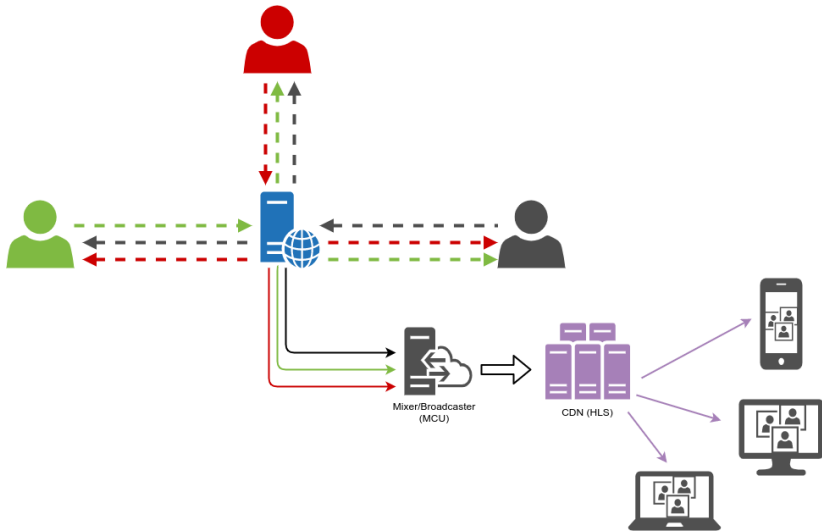


# Live Recording





# Broadcasting to a CDN





# A practical example: Meetecho @ IETF 108

The screenshot shows a web browser window displaying a Meetecho video conference. The browser's address bar shows the URL `gce.conf.meetecho.com/conference/?group=tdd`. The page header includes the IETF logo and the text "108 tdd". Below this, a list of participants is shown, including Dhruv Dhody, Pete Resnick, Jake Holland, and Peter van Dijk. The main area of the page displays three video feeds. The top-left feed shows a man with glasses and a dark shirt, identified as JOÃO DAMAS, with a video bitrate of 263 kbps. The bottom-left feed shows a man with a beard and a blue shirt, with a video bitrate of 158 kbps. The right-side feed shows a man with glasses and a blue shirt, with a video bitrate of 241 kbps. The bottom-right feed shows a man with glasses and a dark shirt, with a video bitrate of 252 kbps. The bottom of the page features the Meetecho logo, the text "Hosted by ERICSSON", and a status bar indicating "audio in: 17 kbps" and "recording".

Activities Google Chrome Fri 00:50

Board M x Path Cor x PCE Syn x IETF 108 x slides-10 x draft-iet x IETF 108 x Agenda x https x IETF 108 x

gce.conf.meetecho.com/conference/?group=tdd

Apps To Read Time Zone Co... SAA Yang INTC IETF tools YANG Data Mo...

108 tdd

**Dhruv Dhody**  
PARTICIPANT

114

AI MORTON

(number of "retries") 00:48:36

**Pete Resnick**

And we've been asking during the talk anyway. 00:49:30

**Jake Holland**

@Geoff: can you expand on "shouldn't run EDNS0"? or client subnet, rather 00:49:49

**Peter van Dijk**

not a question but it's what I have: this was very good, thank you! It might not be a deep dive for me or most people I talked to, but it was very good and this recording will be a valuable educational tool. 00:50:10

Write message...

263 kbps

JOÃO DAMAS

158 kbps

241 kbps

252 kbps

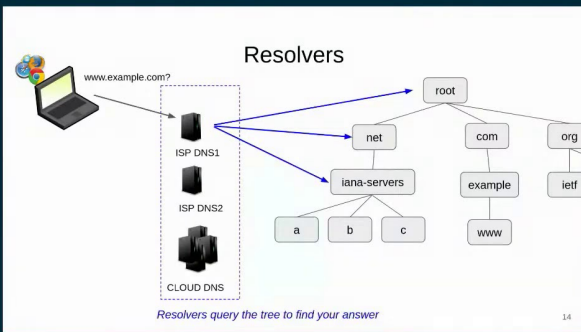
Meetecho Hosted by ERICSSON

audio in: 17 kbps recording



# A practical example: Meetecho @ IETF 108

## Technology Deep Dive: DNS



<https://www.youtube.com/watch?v=DV0q9s94RL8>



# A practical example: Meetecho @ IETF 108

Technology Deep Dive: DNS

**Come back for part 2!**  
Slide Subtitle

And in the meantime, please fill in the survey at

<https://www.surveymonkey.com/r/108TDD>

<https://www.youtube.com/watch?v=DV0q9s94RL8>





## A practical example: Meetecho @ IETF 108

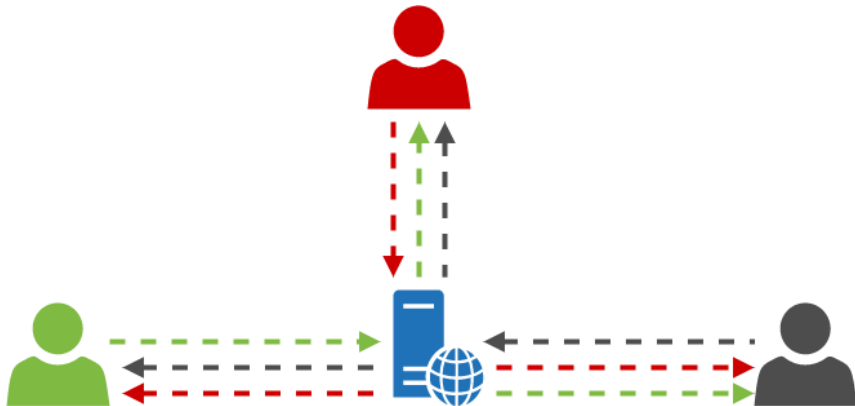


<https://www.youtube.com/watch?v=DV0q9s94RL8>



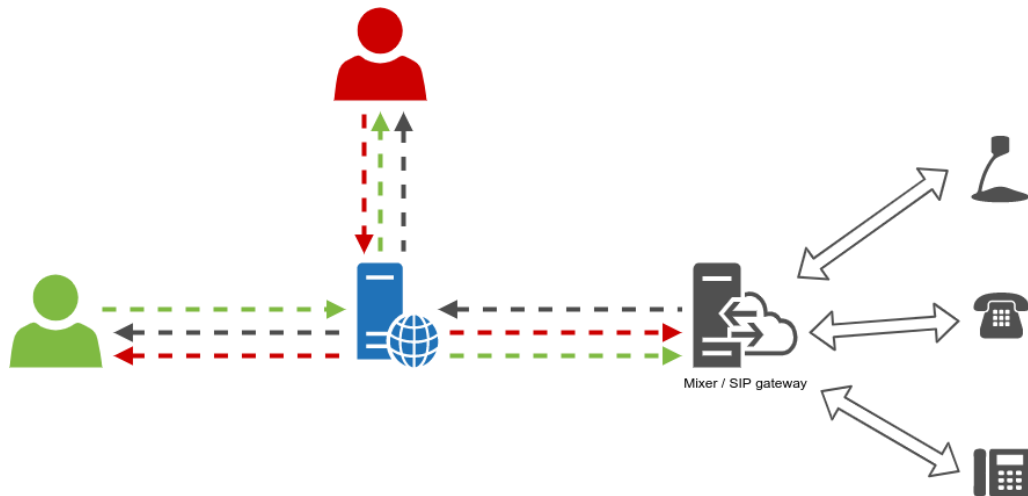


## Integrating with (legacy) conferencing systems



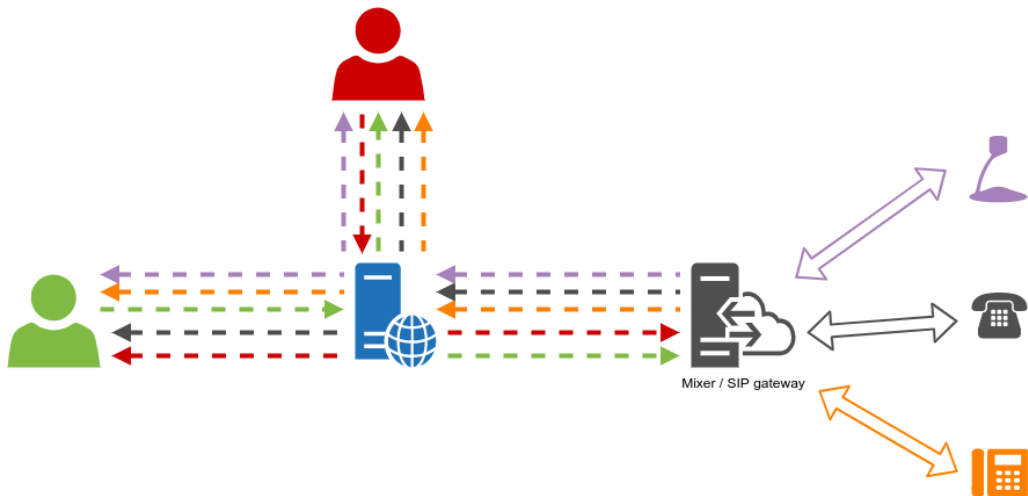


## Integrating with (legacy) conferencing systems



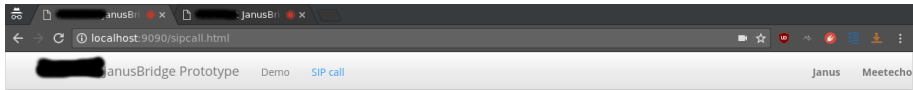


## Integrating with (legacy) conferencing systems

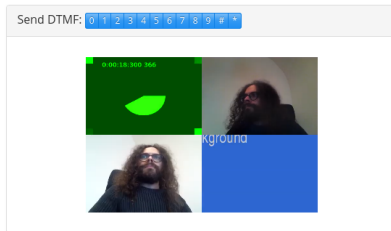
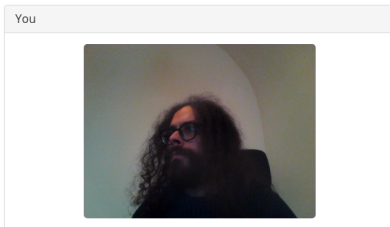




# A few practical examples

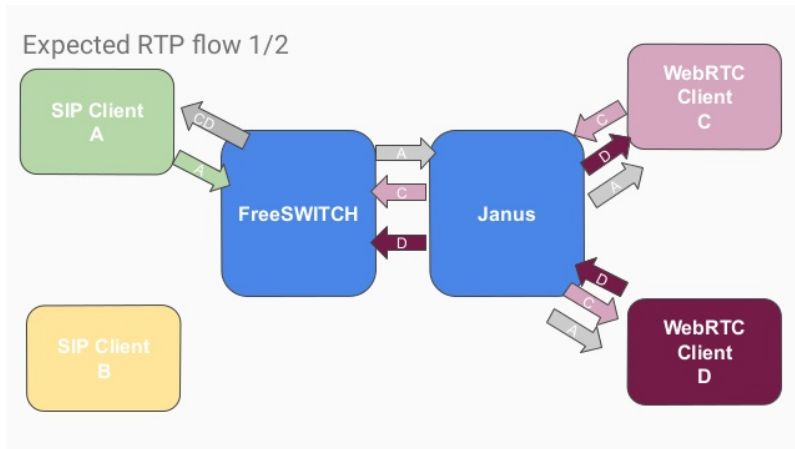


JanusBridge Prototype: SIP user Stop





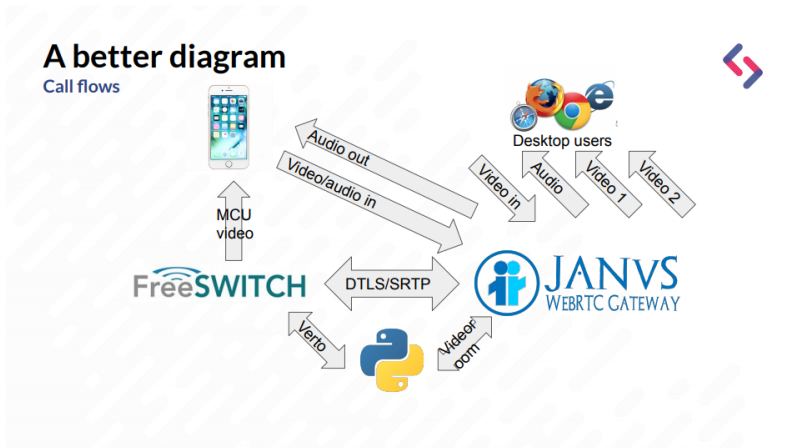
## A few practical examples



<http://www.januscon.it/2019/talk.php?t=nexmo> (Giacomo Vacca)



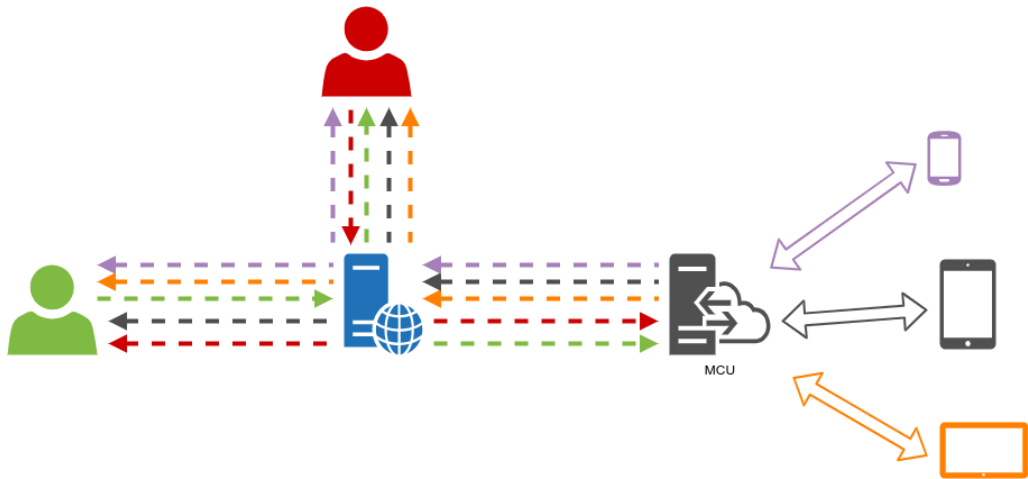
## A few practical examples



<http://www.januscon.it/2019/talk.php?t=mojolinga> (Luca Pradovera)



## Supporting less powerful devices





## What about using browsers as an MCU?

- Browsers often already used that way for recording/broadcasting
  - e.g., headless browser joining as participant
  - External tool (e.g., ffmpeg) captures audio/video from browser
- Canvas (and WebRTC) allow for a more integrated functionality, though
  - e.g., composition done on a canvas object (basically an MCU!)
    - `canvas.captureStream()` to turn it into a WebRTC stream
    - ...and, why not, WebAudio to do audio too!
  - Used by companies like Streamyard and Stage TEN for broadcasting

An ugly canvas+WebRTC demo

- <https://janus.conf.meetecho.com/canvas>





## What about using browsers as an MCU?

- Browsers often already used that way for recording/broadcasting
  - e.g., headless browser joining as participant
  - External tool (e.g., ffmpeg) captures audio/video from browser
- Canvas (and WebRTC) allow for a more integrated functionality, though
  - e.g., composition done on a canvas object (basically an MCU!)
    - `canvas.captureStream()` to turn it into a WebRTC stream
    - ...and, why not, WebAudio to do audio too!
  - Used by companies like Streamyard and Stage TEN for broadcasting

An ugly canvas+WebRTC demo

- <https://janus.conf.meetecho.com/canvas>



## What about using browsers as an MCU?

- Browsers often already used that way for recording/broadcasting
  - e.g., headless browser joining as participant
  - External tool (e.g., ffmpeg) captures audio/video from browser
- Canvas (and WebRTC) allow for a more integrated functionality, though
  - e.g., composition done on a canvas object (basically an MCU!)
    - `canvas.captureStream()` to turn it into a WebRTC stream
    - ...and, why not, WebAudio to do audio too!
  - Used by companies like Streamyard and Stage TEN for broadcasting

An ugly canvas+WebRTC demo

- <https://janus.conf.meetecho.com/canvas>



## What about using browsers as an MCU?

- Browsers often already used that way for recording/broadcasting
  - e.g., headless browser joining as participant
  - External tool (e.g., ffmpeg) captures audio/video from browser
- Canvas (and WebRTC) allow for a more integrated functionality, though
  - e.g., composition done on a canvas object (basically an MCU!)
    - `canvas.captureStream()` to turn it into a WebRTC stream
    - ...and, why not, WebAudio to do audio too!
  - Used by companies like Streamyard and Stage TEN for broadcasting

An ugly canvas+WebRTC demo

- <https://janus.conf.meetecho.com/canvas>



## What about using browsers as an MCU?

- Browsers often already used that way for recording/broadcasting
  - e.g., headless browser joining as participant
  - External tool (e.g., ffmpeg) captures audio/video from browser
- Canvas (and WebRTC) allow for a more integrated functionality, though
  - e.g., composition done on a canvas object (basically an MCU!)
    - `canvas.captureStream()` to turn it into a WebRTC stream
    - ...and, why not, WebAudio to do audio too!
  - Used by companies like Streamyard and Stage TEN for broadcasting

An ugly canvas+WebRTC demo

- <https://janus.conf.meetecho.com/canvas>



## What about using browsers as an MCU?

- Browsers often already used that way for recording/broadcasting
  - e.g., headless browser joining as participant
  - External tool (e.g., ffmpeg) captures audio/video from browser
- Canvas (and WebRTC) allow for a more integrated functionality, though
  - e.g., composition done on a canvas object (basically an MCU!)
    - `canvas.captureStream()` to turn it into a WebRTC stream
    - ...and, why not, WebAudio to do audio too!
  - Used by companies like Streamyard and Stage TEN for broadcasting

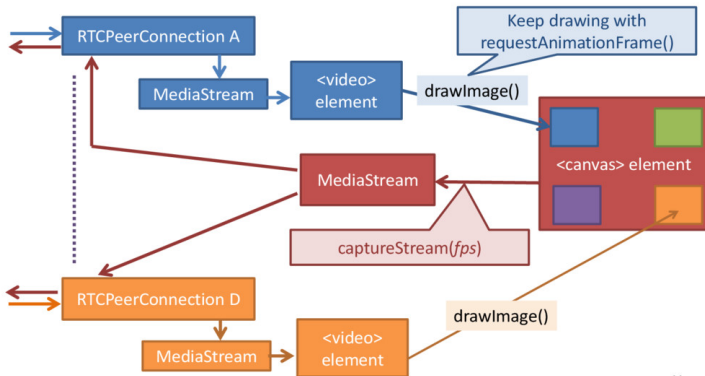
### An ugly canvas+WebRTC demo

- <https://janus.conf.meetecho.com/canvas>



# Masashi Ganeko's MCU in a Browser

## Inside MCU Server: Video

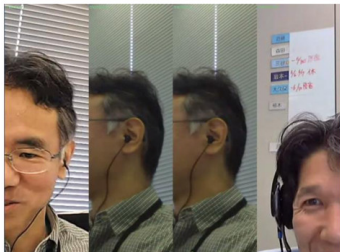




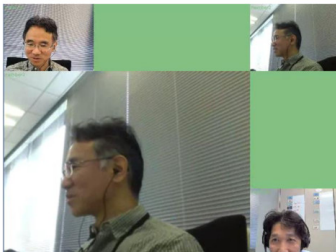
# Masashi Ganeko's MCU in a Browser

## Browser MCU DEMO 2

Stripe



Variable Zoom






# Tim Panton's Book Club!

Bookclub session

rendezvous.zone/?clubid=a3afed62fd9d22f5ab1f901447c1077f913537e2715ce35fc870366a85bfbdf

Bookclub session



We

### About We

*Out of the Russian revolution comes the inspiration for all futuristic dystopia novels, including Brave New World and Nineteen Eighty-Four.*

Yevgeny Zamyatin's *We* is set in a urban glass city called OneState, regulated by spies and secret police. Citizens of the tyrannical OneState wear identical clothing and are distinguished only by the number assigned to them at birth. The story follows a man called D-503, who dangerously begins to veer from the frame of society after meeting V-220.

in 1921, and was finally published in its home country over a half-century later.

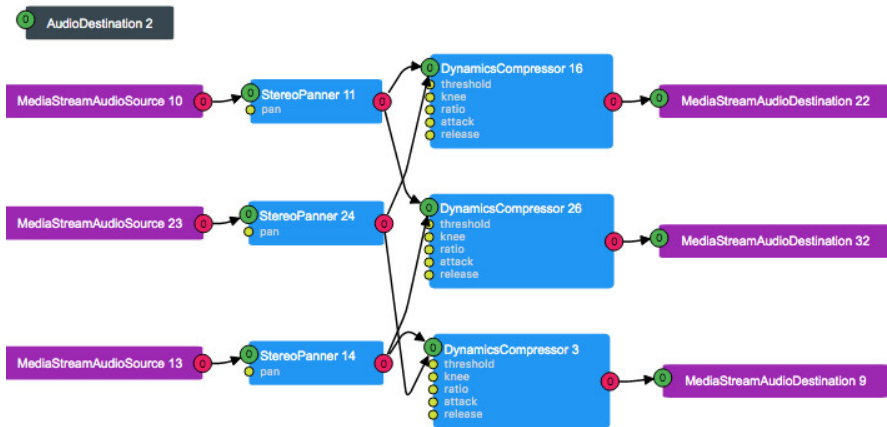
*We* is a part of Momentum's Classic Science Fiction series. It is heralded as "the best single work of science fiction yet written." (Ursula K. Le Guin)

<https://rendezvous.zone>





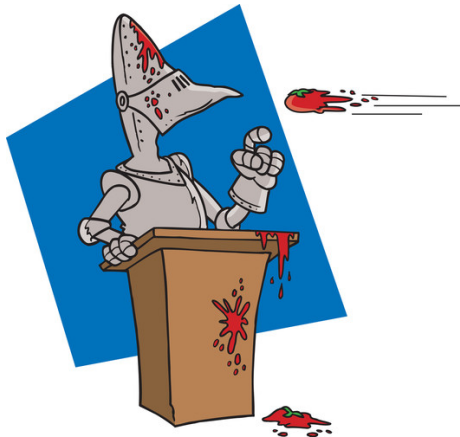
# Tim Panton's Book Club!



<https://rendezvous.zone>



Thanks! Questions? Comments?



### Get in touch!

-  <https://twitter.com/elminiero>
-  <https://twitter.com/meetecho>
-  <https://www.meetecho.com>