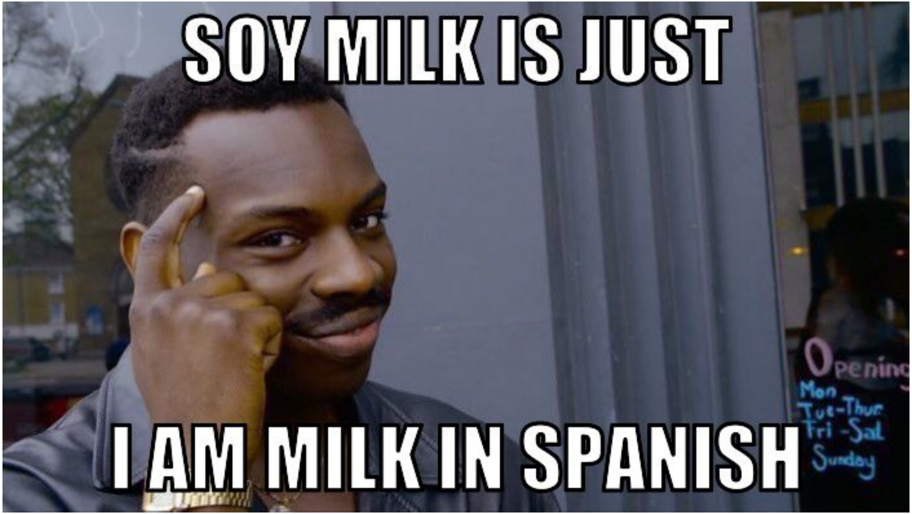
WebRTC-to-SIP and back
It's not all about audio and video!

Lorenzo Miniero
@lminiero@fosstodon.org

OpenSIPS Summit

May 15th 2024, Valencia

A chance to practice my broken Spanish!



Who am I?

Lorenzo Miniero

- Ph.D @ UniNA
- Chairman @ Meetecho
- Main author of Janus

Contacts and info

- lorenzo@meetecho.com
- https://fosstodon.org/@lminiero
- https://www.meetecho.com
- https://lminiero.it

Just a few words on Meetecho

- Co-founded in 2009 as an academic spin-off
  - University research efforts brought to the market
  - Completely independent from the University
- Focus on real-time multimedia applications
  - Strong perspective on standardization and open source
- Several activities
  - Consulting services
  - Commercial support and Janus licenses
  - Streaming of live events (IETF, ACM, etc.)
- Proudly brewed in sunny Napoli, Italy

A bit of context: Janus, WebRTC and SIP

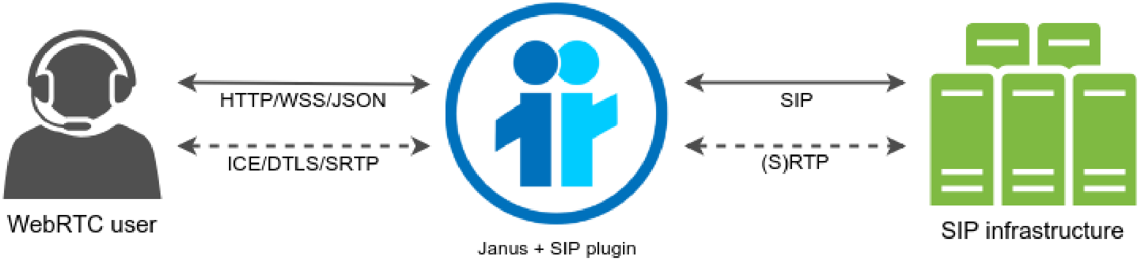
JANVS
WEBRTC SERVER

Janus

General purpose, open source WebRTC server

- https://github.com/meetecho/janus-gateway
- Demos and documentation: https://janus.conf.meetecho.com
- Community: https://janus.discourse.group/

SIP plugin in Janus



https://janus.conf.meetecho.com/docs/sip

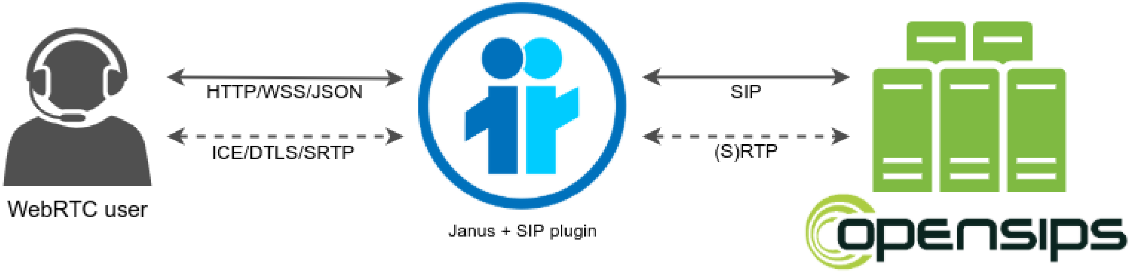
An endpoint of behalf of WebRTC users

- Janus SIP plugin acts as a collection of SIP endpoints, not a server/trunk
  - SIP stack implemented with Sofia-SIP
  - WebRTC users only see the Janus API (JSON), no SIP
  - No transcoding, media is only relayed
  - Built-in recording (separate media legs)
- Simplifies life for web developers
  - No need to worry about a SIP stack (only SIP URIs)
  - Basic methods/events to handle dialogs (call, answer, hangup, message, etc.)
  - Allows SIP headers injection/interception in many requests
  - Support for more advanced features too (e.g., hold, transfer, etc.)

An endpoint of behalf of WebRTC users

- Janus SIP plugin acts as a collection of SIP endpoints, not a server/trunk
  - SIP stack implemented with Sofia-SIP
  - WebRTC users only see the Janus API (JSON), no SIP
  - No transcoding, media is only relayed
  - Built-in recording (separate media legs)
- Simplifies life for web developers
  - No need to worry about a SIP stack (only SIP URIs)
  - Basic methods/events to handle dialogs (call, answer, hangup, message, etc.)
  - Allows SIP headers injection/interception in many requests
  - Support for more advanced features too (e.g., hold, transfer, etc.)

Works great with OpenSIPS!



Workshop on Janus and SIP (lesson/tutorial) at OpenSIPS 2020
https://www.youtube.com/watch?v=fv9KwrguR-4&t=3544s

Audio and video are “easy”

- Both SIP and WebRTC use SDP and RTP/RTCP
  - WebRTC uses SDP/RTP/RTCP on “steroids”
  - Apart from this, just differences in encryption (WebRTC mandates DTLS-SRTP)
- Media is basically encoded, packaged and sent the same way
  - As long as the same codec is used, they’re interoperable
  - When they aren’t, transcoding helps (but Janus won’t do it for you)
- WebRTC has mandatory-to-implement codecs
  - Opus and G.711 for audio, VP8 and H.264 (baseline) for video
  - G.711 and H.264 should guarantee compliance with legacy equipment

Audio and video are "easy"

- Both SIP and WebRTC use SDP and RTP/RTCP
  - WebRTC uses SDP/RTP/RTCP on "steroids"
  - Apart from this, just differences in encryption (WebRTC mandates DTLS-SRTP)
- Media is basically encoded, packaged and sent the same way
  - As long as the same codec is used, they're interoperable
  - When they aren't, transcoding helps (but Janus won't do it for you)
- WebRTC has mandatory-to-implement codecs
  - Opus and G.711 for audio, VP8 and H.264 (baseline) for video
  - G.711 and H.264 should guarantee compliance with legacy equipment

Audio and video are “easy”

- Both SIP and WebRTC use SDP and RTP/RTCP
  - WebRTC uses SDP/RTP/RTCP on “steroids”
  - Apart from this, just differences in encryption (WebRTC mandates DTLS-SRTP)
- Media is basically encoded, packaged and sent the same way
  - As long as the same codec is used, they’re interoperable
  - When they aren’t, transcoding helps (but Janus won’t do it for you)
- WebRTC has mandatory-to-implement codecs
  - Opus and G.711 for audio, VP8 and H.264 (baseline) for video
  - G.711 and H.264 should guarantee compliance with legacy equipment

SIP sometimes does more than that, though!

- A whole ecosystem of other protocols that could be used
  - Real-Time Text (T.140 over RTP)
  - Message Session Relay Protocol (MSRP)
  - Binary Floor Control Protocol (BFCP)
  - Fax (T.38 over RTP)
  - ...
- These protocols can't simply be gateway-ed to WebRTC
  - WebRTC supports RTP, but only for audio/video, not generic data
  - Custom protocols are not supported at all
- WebRTC-to-SIP gateways will in general strip them from the SDP
  - We can't rely on a WebRTC browser to simply reject unsupported media
  - An unsupported m-line will cause an exception in `setRemoteDescription`

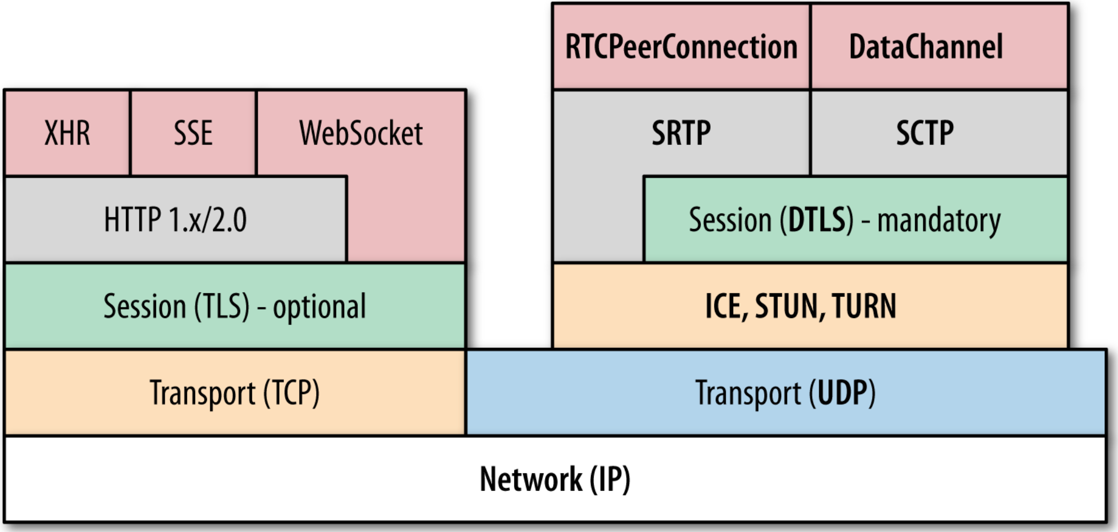
SIP sometimes does more than that, though!

- A whole ecosystem of other protocols that could be used
  - Real-Time Text (T.140 over RTP)
  - Message Session Relay Protocol (MSRP)
  - Binary Floor Control Protocol (BFCP)
  - Fax (T.38 over RTP)
  - ...
- These protocols can't simply be gateway-ed to WebRTC
  - WebRTC supports RTP, but only for audio/video, not generic data
  - Custom protocols are not supported at all
- WebRTC-to-SIP gateways will in general strip them from the SDP
  - We can't rely on a WebRTC browser to simply reject unsupported media
  - An unsupported m-line will cause an exception in `setRemoteDescription`

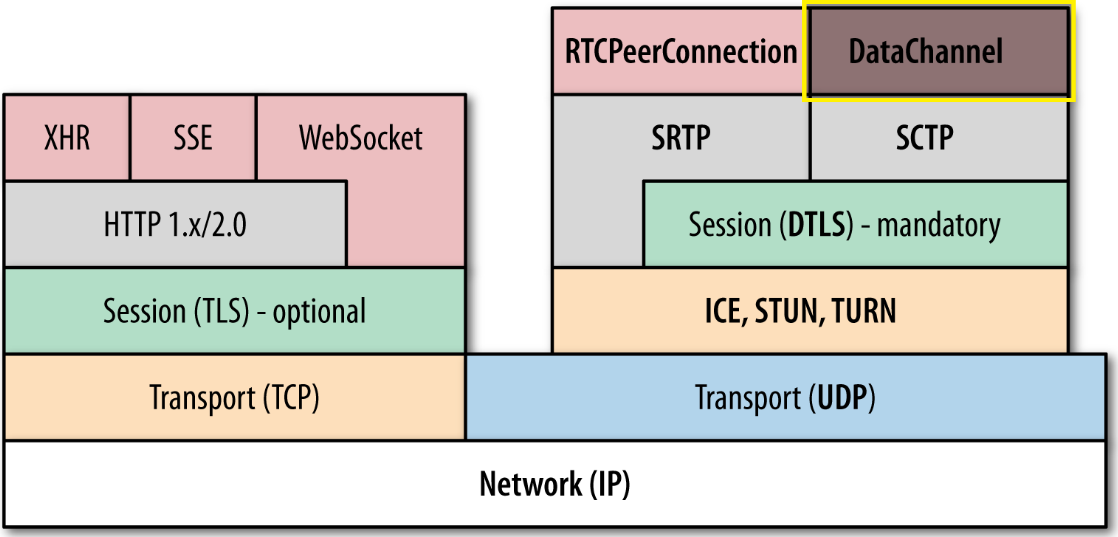
SIP sometimes does more than that, though!

- A whole ecosystem of other protocols that could be used
  - Real-Time Text (T.140 over RTP)
  - Message Session Relay Protocol (MSRP)
  - Binary Floor Control Protocol (BFCP)
  - Fax (T.38 over RTP)
  - ...
- These protocols can't simply be gateway-ed to WebRTC
  - WebRTC supports RTP, but only for audio/video, not generic data
  - Custom protocols are not supported at all
- WebRTC-to-SIP gateways will in general strip them from the SDP
  - We can't rely on a WebRTC browser to simply reject unsupported media
  - An unsupported m-line will cause an exception in **`setRemoteDescription`**

Data channels to the rescue!



Data channels to the rescue!



What are data channels?

- Arbitrary real-time connection for pretty much anything
  - Bidirectional data between two WebRTC peers
  - Support for multiple channels of different kinds
  - Supports ordered/unordered and reliable/unreliable
- Generic data sent via SCTP and encapsulated in DTLS
  - SCTP implements features, DTLS implements security
- Negotiated as an `application` in the SDP
  - `m=application 9 UDP/DTLS/SCTP webrtc-datachannel`

What are data channels?

- Arbitrary real-time connection for pretty much anything
  - Bidirectional data between two WebRTC peers
  - Support for multiple channels of different kinds
  - Supports ordered/unordered and reliable/unreliable
- Generic data sent via SCTP and encapsulated in DTLS
  - SCTP implements features, DTLS implements security
- Negotiated as an `application` in the SDP
  - `m=application 9 UDP/DTLS/SCTP webrtc-datachannel`

What are data channels?

- Arbitrary real-time connection for pretty much anything
  - Bidirectional data between two WebRTC peers
  - Support for multiple channels of different kinds
  - Supports ordered/unordered and reliable/unreliable
- Generic data sent via SCTP and encapsulated in DTLS
  - SCTP implements features, DTLS implements security
- Negotiated as an **`application`** in the SDP
  - **`m=application 9 UDP/DTLS/SCTP webrtc-datachannel`**

That's better, but still not enough!

- Non audio/video protocols have their own technical requirements
  - RTT and T.38 will have custom RTP packetization rules
  - BFCP/MSRP/others will require custom UDP or TCP transport
  - All this will be reflected in the SDP negotiation
- WebRTC only supports `m=application` + `UDP/DTLS/SCTP`
  - It's the only non audio/video thing it can negotiate
- A gateway will have more work to do than with RTP audio/video
  - Translate between SDP formats (e.g., custom protocol ↔ application)
  - Gateway the protocol itself (e.g., RTP/T.140 ↔ data channel)

That's better, but still not enough!

- Non audio/video protocols have their own technical requirements
  - RTT and T.38 will have custom RTP packetization rules
  - BFCP/MSRP/others will require custom UDP or TCP transport
  - All this will be reflected in the SDP negotiation
- WebRTC only supports **`m=application`** + **`UDP/DTLS/SCTP`**
  - It's the only non audio/video thing it can negotiate
- A gateway will have more work to do than with RTP audio/video
  - Translate between SDP formats (e.g., custom protocol ↔ application)
  - Gateway the protocol itself (e.g., RTP/T.140 ↔ data channel)

That's better, but still not enough!

- Non audio/video protocols have their own technical requirements
  - RTT and T.38 will have custom RTP packetization rules
  - BFCP/MSRP/others will require custom UDP or TCP transport
  - All this will be reflected in the SDP negotiation
- WebRTC only supports **`m=application`** + **`UDP/DTLS/SCTP`**
  - It's the only non audio/video thing it can negotiate
- A gateway will have more work to do than with RTP audio/video
  - Translate between SDP formats (e.g., custom protocol ↔ application)
  - Gateway the protocol itself (e.g., RTP/T.140 ↔ data channel)

A practical example: Real-Time Text (T.140)

- Text transmitted instantly, as it is typed or created
  - ITU-T T.140 (Protocol for multimedia application text conversation)
  - Allows for real-time editing (e.g., backspace, rewriting)
- T.140 messages transported over RTP
  - RFC 4103 (RTP Payload for Text Conversation)
  - Redundancy implemented via RED (RFC 2198)

Specification to use T.140 over data channels

- T.140 Real-Time Text Conversation over WebRTC Data Channels (RFC 8865)
- https://www.rfc-editor.org/rfc/rfc8865.html

A practical example: Real-Time Text (T.140)

- Text transmitted instantly, as it is typed or created
  - ITU-T T.140 (Protocol for multimedia application text conversation)
  - Allows for real-time editing (e.g., backspace, rewriting)
- T.140 messages transported over RTP
  - RFC 4103 (RTP Payload for Text Conversation)
  - Redundancy implemented via RED (RFC 2198)

Specification to use T.140 over data channels

- T.140 Real-Time Text Conversation over WebRTC Data Channels (RFC 8865)
- https://www.rfc-editor.org/rfc/rfc8865.html

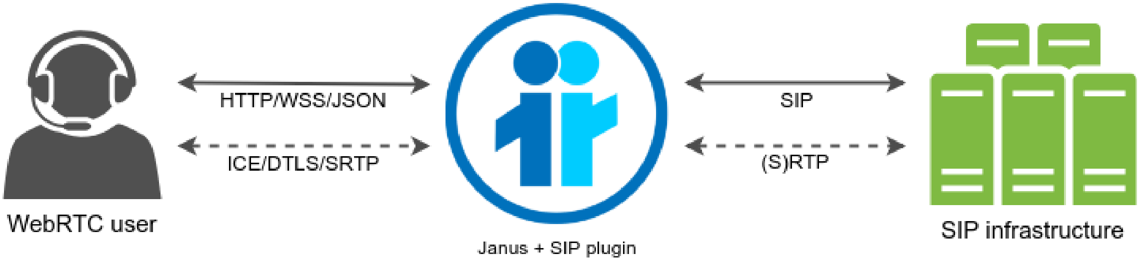
A practical example: Real-Time Text (T.140)

- Text transmitted instantly, as it is typed or created
  - ITU-T T.140 (Protocol for multimedia application text conversation)
  - Allows for real-time editing (e.g., backspace, rewriting)
- T.140 messages transported over RTP
  - RFC 4103 (RTP Payload for Text Conversation)
  - Redundancy implemented via RED (RFC 2198)

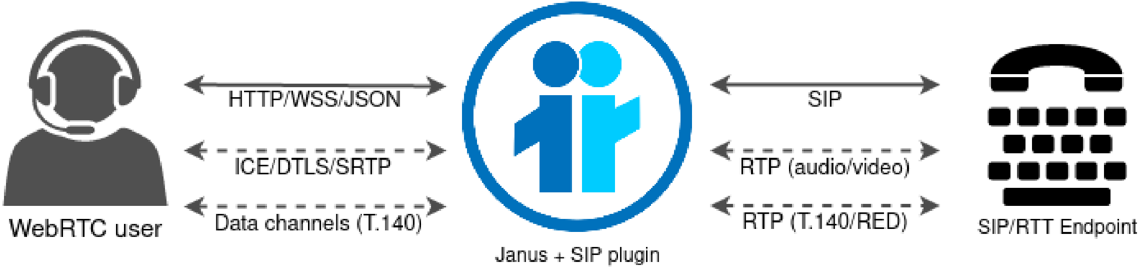
Specification to use T.140 over data channels

- T.140 Real-Time Text Conversation over WebRTC Data Channels (RFC 8865)
- https://www.rfc-editor.org/rfc/rfc8865.html

SIP plugin in Janus normally...



SIP plugin in Janus + RTT



https://github.com/meetecho/janus-gateway/pull/3231

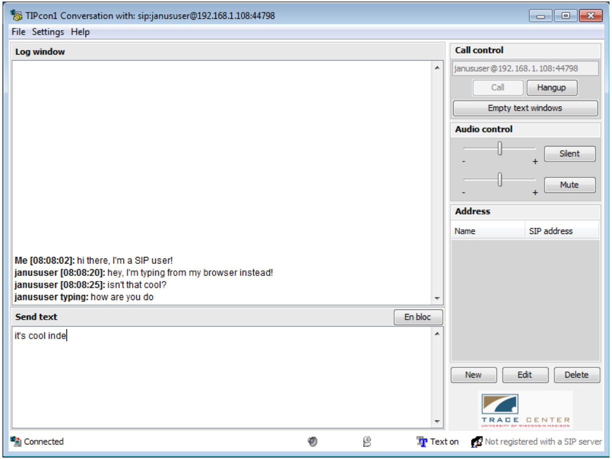
Example: SDP offer from SIP/RTT endpoint

```
v=0
o=Lorenzo_Miniero 1 1 IN IP4 192.168.1.74
s=Omnitor_SDP_v1.1
c=IN IP4 192.168.1.74
t=0 0
m=text 1024 RTP/AVP 99 98
a=rtpmap:99 red/1000
a=fmtp:99 98/98/98
a=rtpmap:98 t140/1000
```

Example: SDP offer to WebRTC endpoint

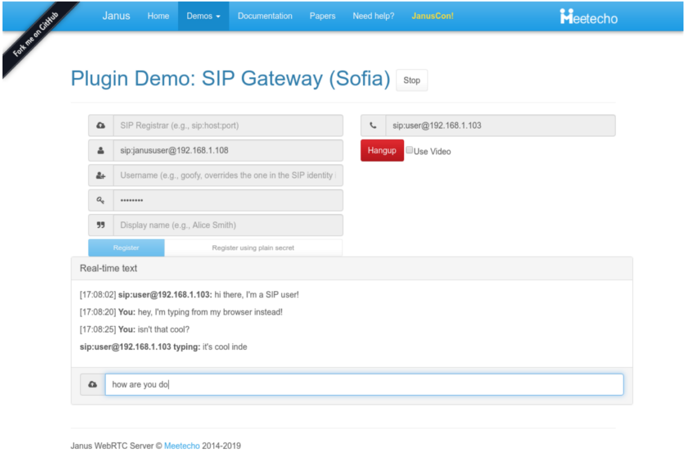
```
v=0
o=Lorenzo_Miniero 1 1 IN IP4 192.168.1.74
s=Omnitor_SDP_v1.1 t=0 0
a=group:BUNDLE data
a=msid-semantic: WMS janus
m=application 9 UDP/DTLS/SCTP webrtc-datachannel
c=IN IP4 192.168.1.74
a=sendrecv
a=sctp-port:5000
a=mid:0
[.. ICE/DTLS details follow ..]
```

Testing: TIPcon1 (SIP/RTT)



https://www.meetecho.com/blog/realtime-text-sip-and-webrtc/

Testing: Janus SIP demo (WebRTC)



https://www.meetecho.com/blog/realtime-text-sip-and-webrtc/

A more complex example: MSRP

- Instant Messaging protocol negotiated in SIP/SDP
  - RFC 4975 (Message Session Relay Protocol)
  - Requires a reliable transport (e.g., TCP)
- Used `m=message` m-line type for negotiation
  - Protocol must be `TCP/MSRP` or `TCP/TLS/MSRP`
  - `path` attribute defines endpoints

Specification to use MSRP over data channels

- Message Session Relay Protocol (MSRP) over Data Channels (RFC 8873)
- https://www.rfc-editor.org/rfc/rfc8873.html

A more complex example: MSRP

- Instant Messaging protocol negotiated in SIP/SDP
  - RFC 4975 (Message Session Relay Protocol)
  - Requires a reliable transport (e.g., TCP)
- Used **`m=message`** m-line type for negotiation
  - Protocol must be **`TCP/MSRP`** or **`TCP/TLS/MSRP`**
  - **`path`** attribute defines endpoints

Specification to use MSRP over data channels

- Message Session Relay Protocol (MSRP) over Data Channels (RFC 8873)
- https://www.rfc-editor.org/rfc/rfc8873.html

A more complex example: MSRP

- Instant Messaging protocol negotiated in SIP/SDP
  - RFC 4975 (Message Session Relay Protocol)
  - Requires a reliable transport (e.g., TCP)
- Used **`m=message`** m-line type for negotiation
  - Protocol must be **`TCP/MSRP`** or **`TCP/TLS/MSRP`**
  - **`path`** attribute defines endpoints

Specification to use MSRP over data channels

- Message Session Relay Protocol (MSRP) over Data Channels (RFC 8873)
- https://www.rfc-editor.org/rfc/rfc8873.html

Much more complex to handle than RTT

- T.140 still uses RTP (and UDP)
  - Easier to integrate in existing Janus SIP plugin bridging
  - Complexity mostly in SDP translation (and maybe RED)
- MSRP requires a reliable transport protocol
  - With TCP, who connects to who?
  - Integrating TCP with UDP poll loop can be tricky (head-of-line blocking)
- On the WebRTC side, we need to be able to provide MSRP path
  - It won't be in the SDP (attribute stripped) and `dcsa` unsupported by browsers
  - Needs out of band (Janus SIP plugin API?) mechanism for that

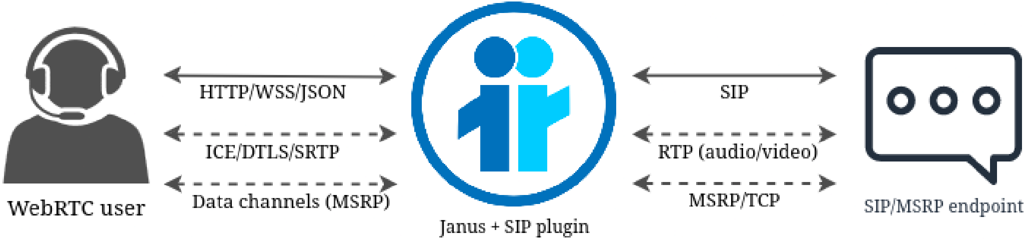
Much more complex to handle than RTT

- T.140 still uses RTP (and UDP)
  - Easier to integrate in existing Janus SIP plugin bridging
  - Complexity mostly in SDP translation (and maybe RED)
- MSRP requires a reliable transport protocol
  - With TCP, who connects to who?
  - Integrating TCP with UDP poll loop can be tricky (head-of-line blocking)
- On the WebRTC side, we need to be able to provide MSRP path
  - It won't be in the SDP (attribute stripped) and `dcsa` unsupported by browsers
  - Needs out of band (Janus SIP plugin API?) mechanism for that

Much more complex to handle than RTT

- T.140 still uses RTP (and UDP)
  - Easier to integrate in existing Janus SIP plugin bridging
  - Complexity mostly in SDP translation (and maybe RED)
- MSRP requires a reliable transport protocol
  - With TCP, who connects to who?
  - Integrating TCP with UDP poll loop can be tricky (head-of-line blocking)
- On the WebRTC side, we need to be able to provide MSRP path
  - It won't be in the SDP (attribute stripped) and `dcsa` unsupported by browsers
  - Needs out of band (Janus SIP plugin API?) mechanism for that

SIP plugin in Janus + MSRP



https://github.com/meetecho/janus-gateway/tree/sip-msrp

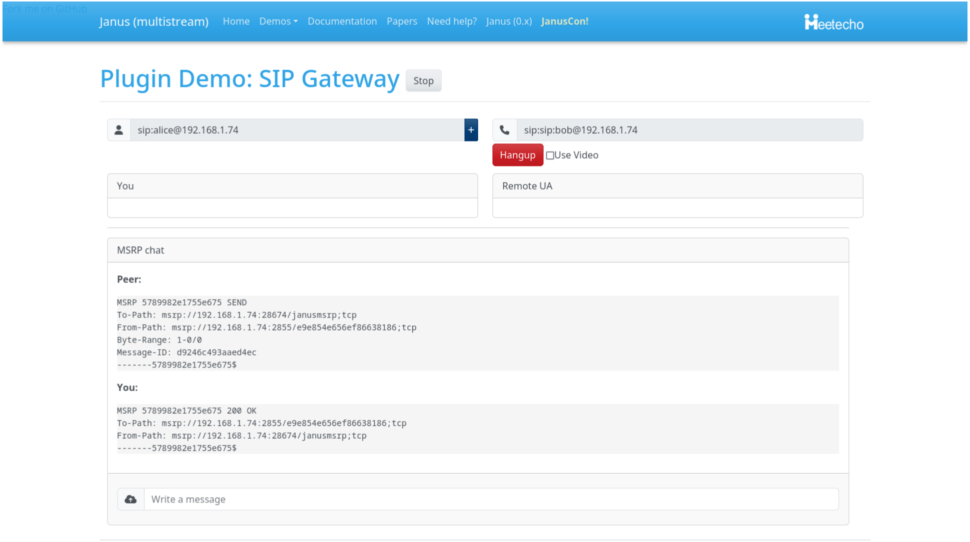
Example: SDP offer from Blink (SIP/MSRP)

```
v=0
o=- 3922178296 3922178296 IN IP4 192.168.1.74
s=Blink 5.5.1 (Linux)
t=0 0
m=message 2855 TCP/MSRP *
c=IN IP4 192.168.1.74
a=path:msrp://192.168.1.74:2855/59a83c96684d4fc5fa41;tcp
a=accept-types:message/cpim text/* image/* \\
        application/im-iscomposing+xml
a=accept-wrapped-types:text/* image/* \\
        application/im-iscomposing+xml
a=setup:active
```

Example: SDP offer to WebRTC endpoint

```
v=0
o=- 3922178296 3922178296 IN IP4 192.168.1.74
s=Blink 5.5.1 (Linux)
t=0 0
a=group:BUNDLE 0
a=extmap-allow-mixed
a=msid-semantic: WMS *
m=application 9 UDP/DTLS/SCTP webrtc-datachannel
c=IN IP4 192.168.1.74
a=sendrecv
a=sctp-port:5000
[.. ICE/DTLS details follow ..]
```

Testing: Janus SIP demo (WebRTC)



What's next?

- Both RTT and MSRP "work" in Janus, but there's a lot to do!
  - RTT is in a better shape, but needs testing with actual endpoints
  - MSRP in a more embrional stage, needs a lot of love
- Multiple protocols in same PeerConnection not supported yet either
  - Data channels allow for that (single m-line, multiple labels)
  - We need better mapping when dealing with SDP translation
- Maybe add support for other protocols?
  - Not sure what's really used and needed, out there
- Is this effort really worth it, though?
  - Probably yes for RTT (Emergency Services mandate it)
  - Other protocols may have better "solutions" already

What's next?

- Both RTT and MSRP "work" in Janus, but there's a lot to do!
  - RTT is in a better shape, but needs testing with actual endpoints
  - MSRP in a more embrional stage, needs a lot of love
- Multiple protocols in same PeerConnection not supported yet either
  - Data channels allow for that (single m-line, multiple labels)
  - We need better mapping when dealing with SDP translation
- Maybe add support for other protocols?
  - Not sure what's really used and needed, out there
- Is this effort really worth it, though?
  - Probably yes for RTT (Emergency Services mandate it)
  - Other protocols may have better "solutions" already

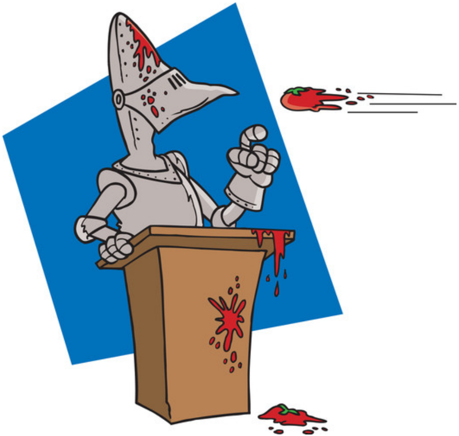
What's next?

- Both RTT and MSRP "work" in Janus, but there's a lot to do!
  - RTT is in a better shape, but needs testing with actual endpoints
  - MSRP in a more embrional stage, needs a lot of love
- Multiple protocols in same PeerConnection not supported yet either
  - Data channels allow for that (single m-line, multiple labels)
  - We need better mapping when dealing with SDP translation
- Maybe add support for other protocols?
  - Not sure what's really used and needed, out there
- Is this effort really worth it, though?
  - Probably yes for RTT (Emergency Services mandate it)
  - Other protocols may have better "solutions" already

What's next?

- Both RTT and MSRP "work" in Janus, but there's a lot to do!
  - RTT is in a better shape, but needs testing with actual endpoints
  - MSRP in a more embrional stage, needs a lot of love
- Multiple protocols in same PeerConnection not supported yet either
  - Data channels allow for that (single m-line, multiple labels)
  - We need better mapping when dealing with SDP translation
- Maybe add support for other protocols?
  - Not sure what's really used and needed, out there
- Is this effort really worth it, though?
  - Probably yes for RTT (Emergency Services mandate it)
  - Other protocols may have better "solutions" already

Thanks! Questions? Comments?



Contacts

- https://fosstodon.org/@lminiero
- https://twitter.com/elminiero
- https://twitter.com/meetecho
- https://www.meetecho.com/blog/