OpenSIPS'17
L. Miniero
Intro
WebRTC
SIP and WebRTC
Janus
Modules and APIs
Janus and SIP
Monitoring
Next steps

Ranch, Caesar or Olive Oil? Different dressings for your SIP salad with Janus

Lorenzo Miniero
@elminiero

OpenSIPS Summit 2017
2nd May 2017,

Outline

1. A brief introduction
2. Some context
   - The problem: getting SIP and WebRTC to like each other
3. Different dressings for your "SIP salad" with Janus
   - Modular architecture: Janus and its plugins
   - What can Janus do to help with SIP?
   - Monitoring and troubleshooting
4. Next steps

What's Meetecho?

- A company born in 2009 as an academic spin-off
  - University research efforts brought to the market
  - Proudly brewed in sunny Napoli, Italy ☺
- Focus on real-time multimedia applications
  - Web conferencing only, at first
  - Then widened the scope to multimedia in general
  - Strong perspective on standardization and open source
    - *WebRTC rulez!*
- Several activities
  - Consulting services
  - Commercial support & licenses
  - Streaming of live events (e.g., IETF, ACM SIGCOMM, ...)
  - Products (conferencing, webinar, ...)

What's Meetecho?

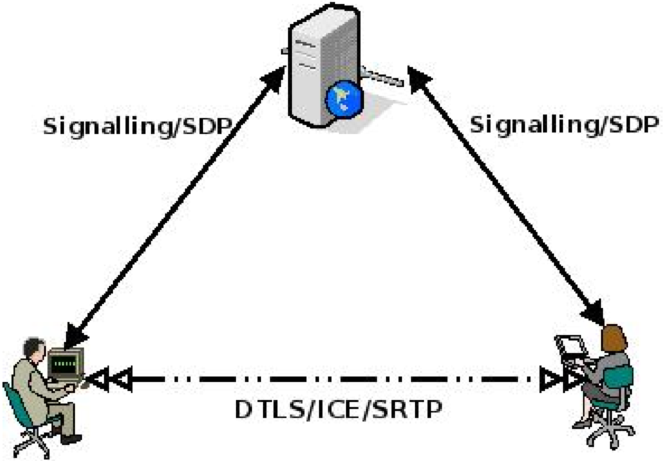
- A company born in 2009 as an academic spin-off
  - University research efforts brought to the market
  - Proudly brewed in sunny Napoli, Italy ☺
- Focus on real-time multimedia applications
  - Web conferencing only, at first
  - Then widened the scope to multimedia in general
  - Strong perspective on standardization and open source
    - *WebRTC rulez!*
- Several activities
  - Consulting services
  - Commercial support & licenses
  - Streaming of live events (e.g., IETF, ACM SIGCOMM, ...)
  - Products (conferencing, webinar, ...)

What's Meetecho?

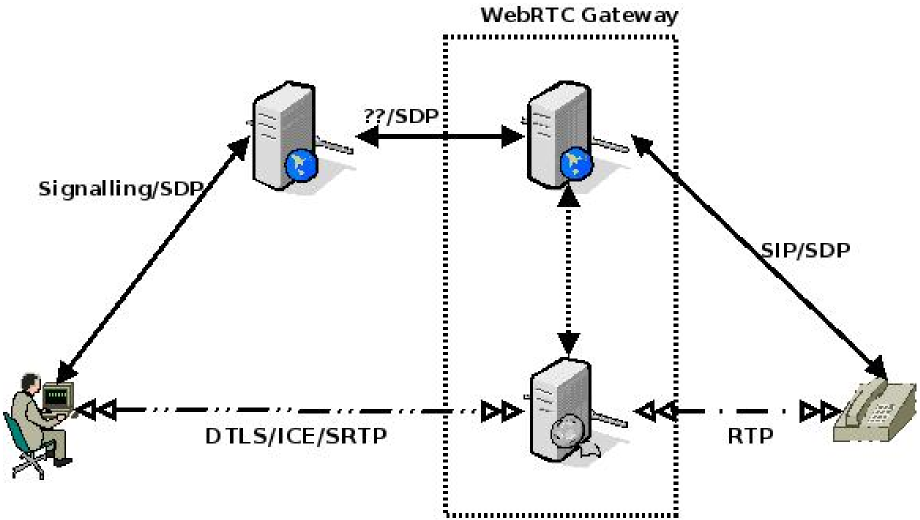
- A company born in 2009 as an academic spin-off
  - University research efforts brought to the market
  - Proudly brewed in sunny Napoli, Italy ☺
- Focus on real-time multimedia applications
  - Web conferencing only, at first
  - Then widened the scope to multimedia in general
  - Strong perspective on standardization and open source
    - *WebRTC rulez!*
- Several activities
  - Consulting services
  - Commercial support & licenses
  - Streaming of live events (e.g., IETF, ACM SIGCOMM, ...)
  - Products (conferencing, webinar, ...)

(*Napoli looks a bit like this...)

Getting SIP and WebRTC to like each other



Getting SIP and WebRTC to like each other



Bridging the gap: the WebRTC protocol suite

- Signalling (well, sort of) and Negotiation
  - Javascript Session Establishment Protocol (JSEP)
  - Session Description Protocol (SDP) adaptation
- Connection Establishment and NAT Traversal
  - Session Traversal Utilities for NAT (STUN)
  - Traversal Using Relay NAT (TURN)
  - Interactive Connectivity Establishment (ICE)
- Media Transport and Control
  - Real-time Transport (and Control) Protocol (RTP/RTCP)
  - Secure Extensions to RTP (SRTP)
  - Datagram Transport Layer Security (DTLS)
- Multimedia codecs
  - Opus audio codec (MTI, Mandatory-to-implement)
  - VP8 and H.264 video codecs (MTI, Mandatory-to-implement)
- Generic Data
  - WebRTC Data Channels (SCTP)

Bridging the gap: the WebRTC protocol suite

- Signalling (well, sort of) and Negotiation
  - Javascript Session Establishment Protocol (JSEP)
  - Session Description Protocol (SDP) adaptation
- Connection Establishment and NAT Traversal
  - Session Traversal Utilities for NAT (STUN)
  - Traversal Using Relay NAT (TURN)
  - Interactive Connectivity Establishment (ICE)
- Media Transport and Control
  - Real-time Transport (and Control) Protocol (RTP/RTCP)
  - Secure Extensions to RTP (SRTP)
  - Datagram Transport Layer Security (DTLS)
- Multimedia codecs
  - Opus audio codec (MTI, Mandatory-to-implement)
  - VP8 and H.264 video codecs (MTI, Mandatory-to-implement)
- Generic Data
  - WebRTC Data Channels (SCTP)

One way to handle this: remember Janus?

Check the recordings from last year!
https://youtu.be/SFeWYewoL7Q

Janus: a general purpose WebRTC gateway

- A door between the communications past and future
  - Legacy technologies (the "past")
  - WebRTC (the "future")

Janus

General purpose, open source WebRTC gateway

- https://github.com/meetecho/janus-gateway
- Demos and documentation: https://janus.conf.meetecho.com
- Community: https://groups.google.com/forum/#!forum/meetecho-janus

Modular architecture

- The core only implements the WebRTC stack
  - JSEP/SDP, ICE, DTLS-SRTP, Data Channels, ...
  - API over HTTP / WebSockets / RabbitMQ / Unix Sockets / MQTT
- Application logic implemented in server side plugins
  - Users attach to plugins via the gateway core
  - The gateway handles the WebRTC stuff
  - Plugins route/manipulate the media/data
- Some proof of concept plugins implemented
  - Echo Test
  - Streaming ($\rightarrow$ Live events!)
  - Video Room ($\rightarrow$ Selective Forwarding Unit!)
  - **SIP Gateway ($\rightarrow$ "Legacy" SIP!)**
  - ...

Modular architecture

- The core only implements the WebRTC stack
  - JSEP/SDP, ICE, DTLS-SRTP, Data Channels, ...
  - API over HTTP / WebSockets / RabbitMQ / Unix Sockets / MQTT
- Application logic implemented in server side plugins
  - Users attach to plugins via the gateway core
  - The gateway handles the WebRTC stuff
  - Plugins route/manipulate the media/data
- Some proof of concept plugins implemented
  - Echo Test
  - Streaming ($\rightarrow$ Live events!)
  - Video Room ($\rightarrow$ Selective Forwarding Unit!)
  - **SIP Gateway ($\rightarrow$ "Legacy" SIP!)**
  - ...

Modular architecture

- The core only implements the WebRTC stack
  - JSEP/SDP, ICE, DTLS-SRTP, Data Channels, ...
  - API over HTTP / WebSockets / RabbitMQ / Unix Sockets / MQTT
- Application logic implemented in server side plugins
  - Users attach to plugins via the gateway core
  - The gateway handles the WebRTC stuff
  - Plugins route/manipulate the media/data
- Some proof of concept plugins implemented
  - Echo Test
  - Streaming ($\rightarrow$ Live events!)
  - Video Room ($\rightarrow$ Selective Forwarding Unit!)
  - **SIP Gateway ($\rightarrow$ "Legacy" SIP!)**
  - ...

What can Janus do to help with SIP?

- Since day one, SIP support has been available as a Janus plugin
  - Demo: https://janus.conf.meetecho.com/siptest
- Basically a WebRTC-to-SIP gateway
  - WebRTC on one side, SIP(S)/(S)RTP on the other end
- Janus SIP plugin acts as a SIP endpoint
  - SIP stack implemented with Sofia-SIP
  - WebRTC users only see the Janus API (JSON)
  - No transcoding, media is only relayed
- Simplifies life for web developers
  - No need to worry about a SIP stack (only SIP URIs)
  - Basic methods/events to handle call (call, answer, hangup)
  - Allows headers injection, for ad-hoc cases
  - Recently added support for on-hold as well

What can Janus do to help with SIP?

- Since day one, SIP support has been available as a Janus plugin
  - Demo: https://janus.conf.meetecho.com/siptest
- Basically a WebRTC-to-SIP gateway
  - WebRTC on one side, SIP(S)/(S)RTP on the other end
- Janus SIP plugin acts as a SIP endpoint
  - SIP stack implemented with Sofia-SIP
  - WebRTC users only see the Janus API (JSON)
  - No transcoding, media is only relayed
- Simplifies life for web developers
  - No need to worry about a SIP stack (only SIP URIs)
  - Basic methods/events to handle call (call, answer, hangup)
  - Allows headers injection, for ad-hoc cases
  - Recently added support for on-hold as well

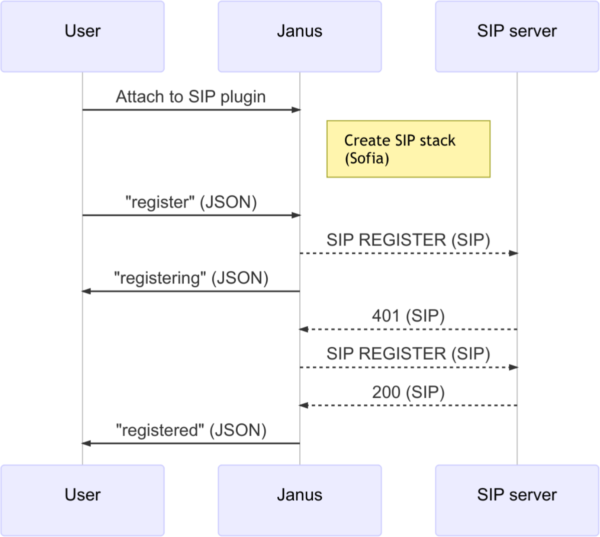
What can Janus do to help with SIP?

- Since day one, SIP support has been available as a Janus plugin
  - Demo: https://janus.conf.meetecho.com/siptest
- Basically a WebRTC-to-SIP gateway
  - WebRTC on one side, SIP(S)/(S)RTP on the other end
- Janus SIP plugin acts as a SIP endpoint
  - SIP stack implemented with Sofia-SIP
  - WebRTC users only see the Janus API (JSON)
  - No transcoding, media is only relayed
- Simplifies life for web developers
  - No need to worry about a SIP stack (only SIP URIs)
  - Basic methods/events to handle call (call, answer, hangup)
  - Allows headers injection, for ad-hoc cases
  - Recently added support for on-hold as well

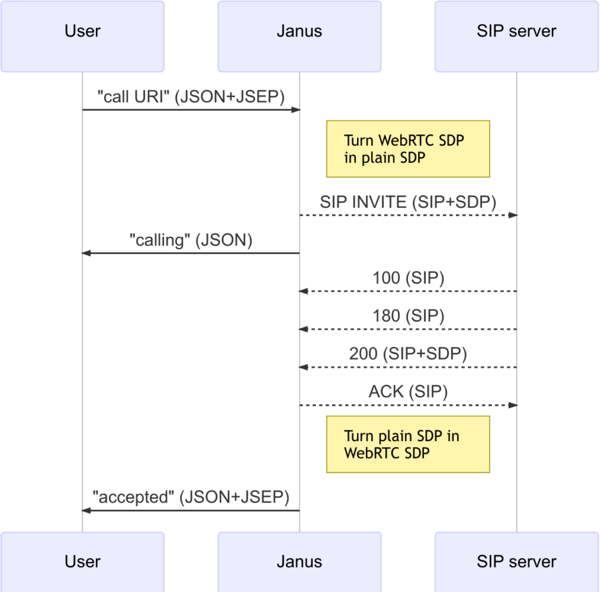
What can Janus do to help with SIP?

- Since day one, SIP support has been available as a Janus plugin
  - Demo: https://janus.conf.meetecho.com/siptest
- Basically a WebRTC-to-SIP gateway
  - WebRTC on one side, SIP(S)/(S)RTP on the other end
- Janus SIP plugin acts as a SIP endpoint
  - SIP stack implemented with Sofia-SIP
  - WebRTC users only see the Janus API (JSON)
  - No transcoding, media is only relayed
- Simplifies life for web developers
  - No need to worry about a SIP stack (only SIP URIs)
  - Basic methods/events to handle call (call, answer, hangup)
  - Allows headers injection, for ad-hoc cases
  - Recently added support for on-hold as well

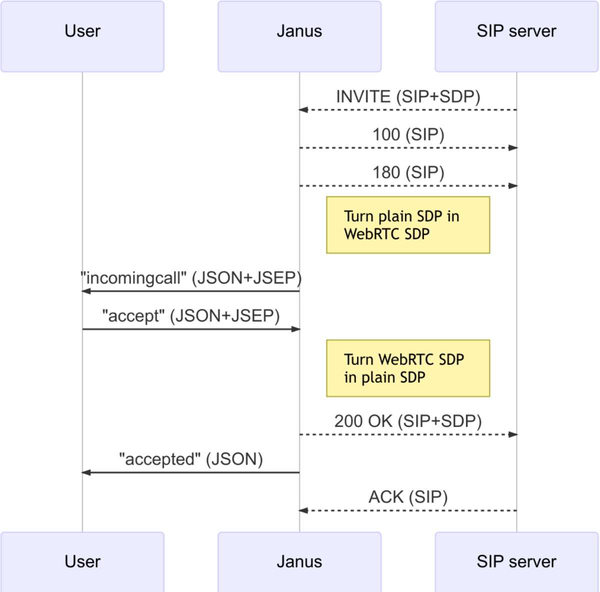
Sofia SIP plugin: a couple of sequence diagrams



Sofia SIP plugin: a couple of sequence diagrams



Sofia SIP plugin: a couple of sequence diagrams



Sofia SIP plugin: how do you use it?


```
{
    "janus": "message",
    "transaction": "rCM4XuZ37LuD",
    "body": {
        "request": "register",
        "username": "sip:janususer@192.168.1.80",
        "secret": "januspwd"
    }
}
```

Sofia SIP plugin: how do you use it?


```
{
    "janus": "event",
    "session_id": 3924816815510633,
    "sender": 3755395465070706,
    "plugindata": {
        "plugin": "janus.plugin.sip",
        "data": {
            "sip": "event",
            "result": {
                "event": "registered",
                "username": "janususer",
                "register_sent": true
            }
        }
    }
}
```

Sofia SIP plugin: how do you use it?


```
{
    "janus": "message",
    "transaction": "pdlWLrRnBbQA",
    "body": {
        "request": "call",
        "uri": "sip:600@192.168.1.80"
    },
    "jsep":{
        "type": "offer",
        "sdp": "v=0\r\n[..]"
    }
}
```

Sofia SIP plugin: how do you use it?


```
{
    "janus": "message",
    "transaction": "pdlWLrRnBbQA",
    "body": {
        "request": "call",
        "uri": "sip:600@192.168.1.80",
        "srtp": "sdes_optional"
    },
    "jsep":{
        "type": "offer",
        "sdp": "v=0\r\n[..]"
    }
}
```

Sofia SIP plugin: how do you use it?


```
{
    "janus": "message",
    "transaction": "pdlWLrRnBbQA",
    "body": {
        "request": "call",
        "uri": "sip:600@192.168.1.80",
        "srtp": "sdes_mandatory"
    },
    "jsep":{
        "type": "offer",
        "sdp": "v=0\r\n[..]"
    }
}
```

Sofia SIP plugin: how do you use it?


```
{
    "janus": "message",
    "transaction": "pdlWLrRnBbQA",
    "body": {
        "request": "call",
        "uri": "sip:600@192.168.1.80",
        "headers": {
            "My-Header": "value",
            "AnotherHeader": "another string"
        }
    },
    "jsep":{
        "type": "offer",
        "sdp": "v=0\r\n[..]"
    }
}
```

Sofia SIP plugin: how do you use it?


```
{
    "janus": "event",
    "session_id": 8037471822771254,
    "sender": 6140585515849261,
    "transaction": "pdlWLrRnBbQA",
    "plugindata": {
        "plugin": "janus.plugin.sip",
        "data": {
            "sip": "event",
            "result": {
                "event": "calling"
            }
        }
    }
}
```

Sofia SIP plugin: how do you use it?


```
{
    "janus": "event",
    "session_id": 8037471822771254,
    "sender": 6140585515849261,
    "transaction": "pdlWLrRnBbQA",
    "plugindata": {
        "plugin": "janus.plugin.sip",
        "data": {
            "sip": "event",
            "result": {
                "event": "accepted",
                "username": "sip:600@192.168.1.80"
            }
        }
    },
    "jsep": {
        "type": "answer",
        "sdp": "v=0\r\n[..]"
    }
}
```

Sofia SIP plugin: how do you use it?


```
{
    "janus": "message",
    "transaction": "uEaTHwcAWlGB",
    "body": {
        "request": "hangup"
    }
}
```

Sofia SIP plugin: how do you use it?


```
{
    "janus": "event",
    "session_id": 8037471822771254,
    "sender": 6140585515849261,
    "plugindata": {
        "plugin": "janus.plugin.sip",
        "data": {
            "sip": "event",
            "result": {
                "event": "incomingcall",
                "username": "sip:anonymous@anonymous.invalid",
                "displayname": "\"Anonymous\""
            }
        }
    },
    "jsep": {
        "type": "offer",
        "sdp": "v=0\r\n[..]"
    }
}
```

Sofia SIP plugin: how do you use it?


```
{
    "janus": "message",
    "transaction": "OBDRz5IB9Qk2",
    "body": {
        "request": "accept"
    },
    "jsep": {
        "type": "answer",
        "sdp": "v=0\r\n[..]"
    }
}
```

Sofia SIP plugin: how do you use it?


```
{
    "janus": "event",
    "session_id": 8037471822771254,
    "sender": 6140585515849261,
    "plugindata": {
        "plugin": "janus.plugin.sip",
        "data": {
            "sip": "event",
            "result": {
                "event": "hangup",
                "code": 200,
                "reason": "Session Terminated"
            }
        }
    }
}
```

Sofia SIP plugin: how do you use it?

```
{
    "janus": "message",
    "transaction": "rCM4XuZ37LuD",
    "body": {
        "request": "hold"
    }
}
```

Sofia SIP plugin: how do you use it?


```
{
    "janus": "message",
    "transaction": "rCM4XuZ37LuD",
    "body": {
        "request": "unhold"
    }
}
```

"But I don't like Sofia SIP!"

- Heard this a few times...
  - While simple and effective, not very flexible and has a few issues
  - Besides, hasn't been updated in a while (apart from Freeswitch)
  - May make it harder for contributors to add/fix things
- What about a similar plugin that uses a different C stack?
- Why not a libre based SIP plugin, then!
  - libre is a very efficient and used library, which should make it easier to use
- Still heavily WIP and incomplete, so help welcome ☺
  - https://github.com/meetecho/janus-gateway/pull/823
- The idea is to keep the API exactly as it is
  - "register", "call", "accept", "hangup", all the events remain the same
  - The plugin namespace is the only thing that changes (`janus.plugin.sipre`)

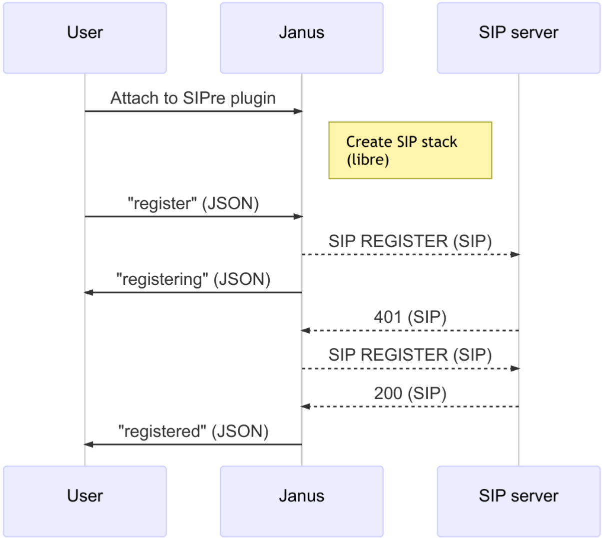
"But I don't like Sofia SIP!"

- Heard this a few times...
  - While simple and effective, not very flexible and has a few issues
  - Besides, hasn't been updated in a while (apart from Freeswitch)
  - May make it harder for contributors to add/fix things
- What about a similar plugin that uses a different C stack?
- Why not a libre based SIP plugin, then!
  - libre is a very efficient and used library, which should make it easier to use
- Still heavily WIP and incomplete, so help welcome ☺
  - https://github.com/meetecho/janus-gateway/pull/823
- The idea is to keep the API exactly as it is
  - "register", "call", "accept", "hangup", all the events remain the same
  - The plugin namespace is the only thing that changes (`janus.plugin.sipre`)

“But I don’t like Sofia SIP!”

- Heard this a few times...
  - While simple and effective, not very flexible and has a few issues
  - Besides, hasn’t been updated in a while (apart from Freeswitch)
  - May make it harder for contributors to add/fix things
- What about a similar plugin that uses a different C stack?
- Why not a libre based SIP plugin, then!
  - libre is a very efficient and used library, which should make it easier to use
- Still heavily WIP and incomplete, so help welcome ☺
  - https://github.com/meetecho/janus-gateway/pull/823
- The idea is to keep the API exactly as it is
  - "register", "call", "accept", "hangup", all the events remain the same
  - The plugin namespace is the only thing that changes (`janus.plugin.sipre`)

libre SIP plugin: sequence diagram



libre SIP plugin: what changes? (spoiler alert: not much)


```
{
    "janus": "attach",
    "plugin": "janus.plugin.sipre",
    "opaque_id": "sipretest-vT4puPukAkCG",
    "transaction":"rzRUwazTqOeW"
}
```

libre SIP plugin: what changes? (spoiler alert: not much)

```
{
    "janus": "message",
    "transaction": "DjVDuL84vUaa",
    "body": {
        "request": "register",
        "username": "sip:janususer@192.168.1.80",
        "secret": "januspwd"
    }
}
```

libre SIP plugin: what changes? (spoiler alert: not much)


```
{
    "janus": "event",
    "session_id": 1523283188775524,
    "sender": 2733688991415468,
    "transaction": "DjVDuL84vUaa",
    "plugindata": {
        "plugin": "janus.plugin.sipre",
        "data": {
            "sip": "event",
            "result": {
                "event": "registering"
            }
        }
    }
}
```

What if you DON'T want it simple?

- You may want to have more control on SIP messaging
  - e.g., to re-use stacks like JsSIP or SIP.js instead of JSON, or other reasons
- Neither the Sofia- nor the libre-based SIP plugins allow for that
  - Complexity hidden from users, on purpose
  - Only partial control (e.g., custom headers, INFO DTMF, negotiating security)
- **BUT!** Again, Janus is extensible, so why not a new plugin?
- @saghul's idea some time ago: "BoringSDP"!
  - A new plugin to only handle media gatewaying
    - WebRTC and SIP SDPs both available to web user
  - You handle SIP transactions yourself, and leave media to Janus
    - You still need to communicate with Janus as well, of course

What if you DON'T want it simple?

- You may want to have more control on SIP messaging
  - e.g., to re-use stacks like JsSIP or SIP.js instead of JSON, or other reasons
- Neither the Sofia- nor the libre-based SIP plugins allow for that
  - Complexity hidden from users, on purpose
  - Only partial control (e.g., custom headers, INFO DTMF, negotiating security)
- **BUT!** Again, Janus is extensible, so why not a new plugin?
- @saghul's idea some time ago: "BoringSDP"!
  - A new plugin to only handle media gatewaying
    - WebRTC and SIP SDPs both available to web user
  - You handle SIP transactions yourself, and leave media to Janus
    - You still need to communicate with Janus as well, of course

What if you DON'T want it simple?

- You may want to have more control on SIP messaging
  - e.g., to re-use stacks like JsSIP or SIP.js instead of JSON, or other reasons
- Neither the Sofia- nor the libre-based SIP plugins allow for that
  - Complexity hidden from users, on purpose
  - Only partial control (e.g., custom headers, INFO DTMF, negotiating security)
- **BUT!** Again, Janus is extensible, so why not a new plugin?
- @saghul's idea some time ago: "BoringSDP"!
  - A new plugin to only handle media gatewaying
    - WebRTC and SIP SDPs both available to web user
  - You handle SIP transactions yourself, and leave media to Janus
    - You still need to communicate with Janus as well, of course

What if you DON'T want it simple?

- You may want to have more control on SIP messaging
  - e.g., to re-use stacks like JsSIP or SIP.js instead of JSON, or other reasons
- Neither the Sofia- nor the libre-based SIP plugins allow for that
  - Complexity hidden from users, on purpose
  - Only partial control (e.g., custom headers, INFO DTMF, negotiating security)
- **BUT!** Again, Janus is extensible, so why not a new plugin?
- @saghul's idea some time ago: "BoringSDP"!
  - A new plugin to only handle media gatewaying
    - WebRTC and SIP SDPs both available to web user
  - You handle SIP transactions yourself, and leave media to Janus
    - You still need to communicate with Janus as well, of course

What if you DON'T want it simple?

- You may want to have more control on SIP messaging
  - e.g., to re-use stacks like JsSIP or SIP.js instead of JSON, or other reasons
- Neither the Sofia- nor the libre-based SIP plugins allow for that
  - Complexity hidden from users, on purpose
  - Only partial control (e.g., custom headers, INFO DTMF, negotiating security)
- **BUT!** Again, Janus is extensible, so why not a new plugin?
- @saghul's idea some time ago: "BoringSDP"!
  - A new plugin to only handle media gatewaying
    - WebRTC and SIP SDPs both available to web user
  - You handle SIP transactions yourself, and leave media to Janus
    - You still need to communicate with Janus as well, of course

BoringSDP brought to life: the NoSIP plugin!

- NoSIP plugin currently available as a PR
  - https://github.com/meetecho/janus-gateway/pull/799
- Still WIP, but already kinda works, so the plan is to merge soon
- Idea is to do exactly what explained in the previous slide
  - The plugin doesn't care about signalling, only SDP
    - You pass it a WebRTC SDP, it gives back a plain SDP
    - You pass it a plain SDP, it gives back a WebRTC SDP
  - If you use the gatewayed SDP in the signalling, media will go to Janus
    - Plugin bridges the media between WebRTC user and legacy client
  - No "register", "call", etc., that's up to you (e.g., SIP or others)

BoringSDP brought to life: the NoSIP plugin!

- NoSIP plugin currently available as a PR
  - https://github.com/meetecho/janus-gateway/pull/799
- Still WIP, but already kinda works, so the plan is to merge soon
- Idea is to do exactly what explained in the previous slide
  - The plugin doesn't care about signalling, only SDP
    - You pass it a WebRTC SDP, it gives back a plain SDP
    - You pass it a plain SDP, it gives back a WebRTC SDP
  - If you use the gatewayed SDP in the signalling, media will go to Janus
    - Plugin bridges the media between WebRTC user and legacy client
  - No "register", "call", etc., that's up to you (e.g., SIP or others)

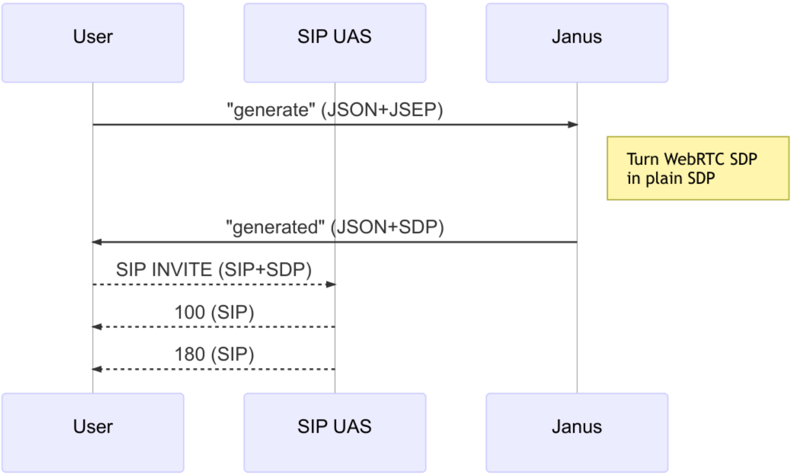
BoringSDP brought to life: the NoSIP plugin!

- NoSIP plugin currently available as a PR
  - https://github.com/meetecho/janus-gateway/pull/799
- Still WIP, but already kinda works, so the plan is to merge soon
- Idea is to do exactly what explained in the previous slide
  - The plugin doesn't care about signalling, only SDP
    - You pass it a WebRTC SDP, it gives back a plain SDP
    - You pass it a plain SDP, it gives back a WebRTC SDP
  - If you use the gatewayed SDP in the signalling, media will go to Janus
    - Plugin bridges the media between WebRTC user and legacy client
  - No "register", "call", etc., that's up to you (e.g., SIP or others)

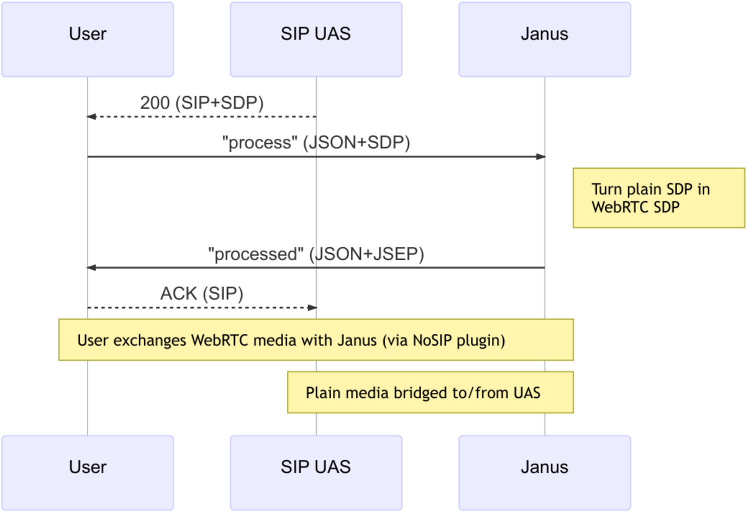
BoringSDP brought to life: the NoSIP plugin!

- NoSIP plugin currently available as a PR
  - https://github.com/meetecho/janus-gateway/pull/799
- Still WIP, but already kinda works, so the plan is to merge soon
- Idea is to do exactly what explained in the previous slide
  - The plugin doesn't care about signalling, only SDP
    - You pass it a WebRTC SDP, it gives back a plain SDP
    - You pass it a plain SDP, it gives back a WebRTC SDP
  - If you use the gatewayed SDP in the signalling, media will go to Janus
    - Plugin bridges the media between WebRTC user and legacy client
  - No "register", "call", etc., that's up to you (e.g., SIP or others)

NoSIP plugin explained: a sequence diagram



NoSIP plugin explained: a sequence diagram



NoSIP plugin: a look at the messaging


```
{
    "janus": "message",
    "session_id": 1596184216412857,
    "handle_id": 1626194946303257,
    "transaction":"l2hu5SBLn2bH",
    "body": {
        "request": "generate"
    },
    "jsep": {
        "type": "offer",
        "sdp": "v=0\r\n[..WebRTC SDP..]"
    }
}
```

NoSIP plugin: a look at the messaging


```
{
    "janus": "event",
    "session_id": 1596184216412857,
    "sender": 1626194946303257,
    "transaction": "l2hu5SBLn2bH",
    "plugindata": {
        "plugin": "janus.plugin.nosip",
        "data": {
            "nosip": "event",
            "result": {
                "event": "generated",
                "type": "offer",
                "sdp": "v=0\r\n[..non-WebRTC SDP..]"
            }
        }
    }
}
```

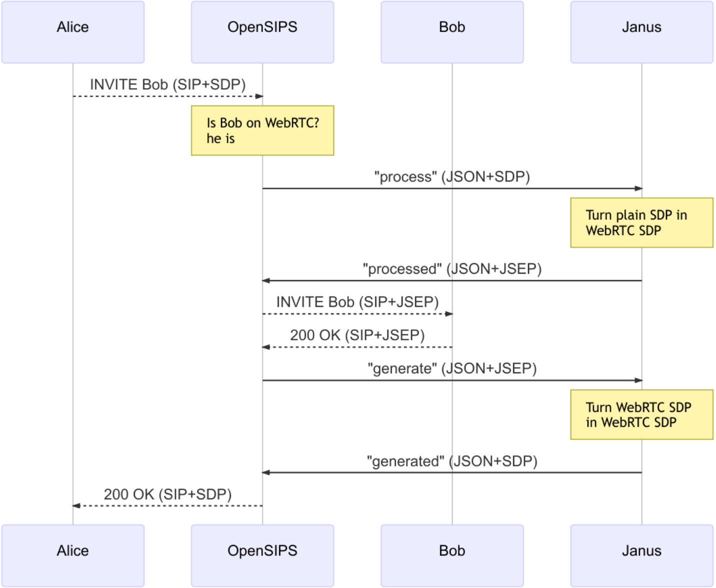
NoSIP plugin: a look at the messaging

```
{
    "janus": "message",
    "session_id": 1596184216412857,
    "sender": 1626194946303257,
    "transaction":"k5w2RAFLXsSP",
    "body": {
        "request": "process",
        "type": "answer",
        "sdp": "v=0\r\n[..non-WebRTC SDP..]"
    }
}
```

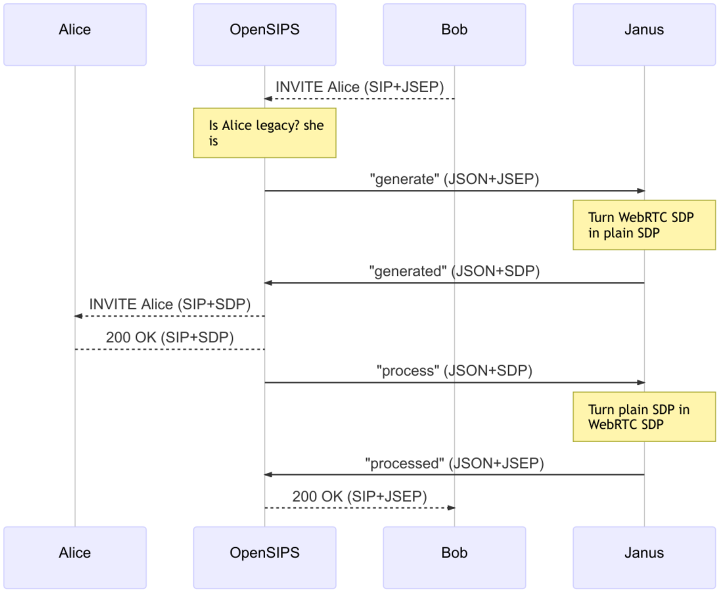
NoSIP plugin: a look at the messaging


```
{
   "janus": "event",
   "session_id": 1596184216412857,
   "sender": 1626194946303257,
   "transaction": "k5w2RAFLXsSP",
   "plugindata": {
      "plugin": "janus.plugin.nosip",
      "data": {
         "nosip": "event",
         "result": {
            "event": "processed"
         }
      }
   },
   "jsep": {
      "type": "answer",
      "sdp": "v=0\r\n[..WebRTC SDP..]"
   }
}
```

A possible use case: OpenSIPS “tool”



A possible use case: OpenSIPS “tool”

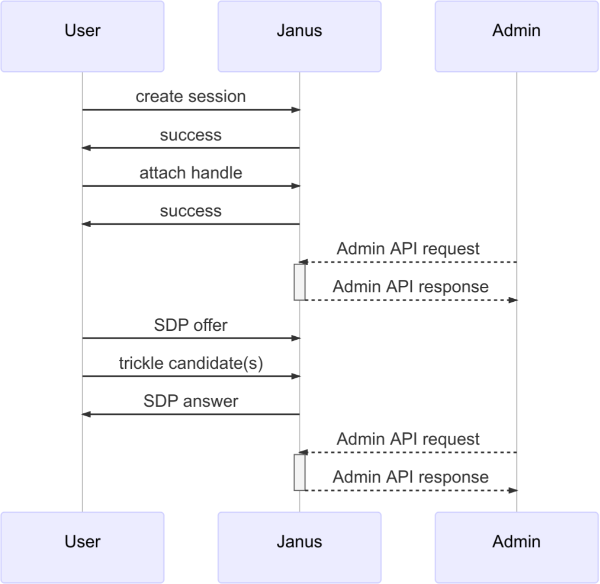


Monitoring/troubleshooting WebRTC/SIP calls: the Admin API

- Requests/response API to interrogate Janus
  - Query server capabilities
  - Control some aspects (e.g., enable/disable debugging)
  - Inspect handles and WebRTC "internals"
    - ... assuming you know the identifiers to query (session/handle)

http://www.meetecho.com/blog/understanding-the-janus-admin-api/

Admin API overview: polling for information



An asynchronous approach to monitoring/troubleshooting

- Admin API is cool, but is request/response...
  - Needs you to poll again to see changes
  - Information is lost when session/handle is gone
  - What about an asynchronous approach instead?
- A new mechanism: Event Handlers
  - Core and plugins generate events
    - Shared "header" (e.g., to identify target of event)
    - Different type of events allows for filtering
  - Custom modules can subscribe to and handle them
    - e.g., save to DB, send to external service, CDR, etc.
  - *Sample Event Handler* forwards JSON events via HTTP
    - Third-party plugins may provide integration with existing frameworks

An asynchronous approach to monitoring/troubleshooting

- Admin API is cool, but is request/response...
  - Needs you to poll again to see changes
  - Information is lost when session/handle is gone
  - What about an asynchronous approach instead?
- A new mechanism: Event Handlers
  - Core and plugins generate events
    - Shared “header” (e.g., to identify target of event)
    - Different type of events allows for filtering
  - Custom modules can subscribe to and handle them
    - e.g., save to DB, send to external service, CDR, etc.
  - *Sample Event Handler* forwards JSON events via HTTP
    - Third-party plugins may provide integration with existing frameworks

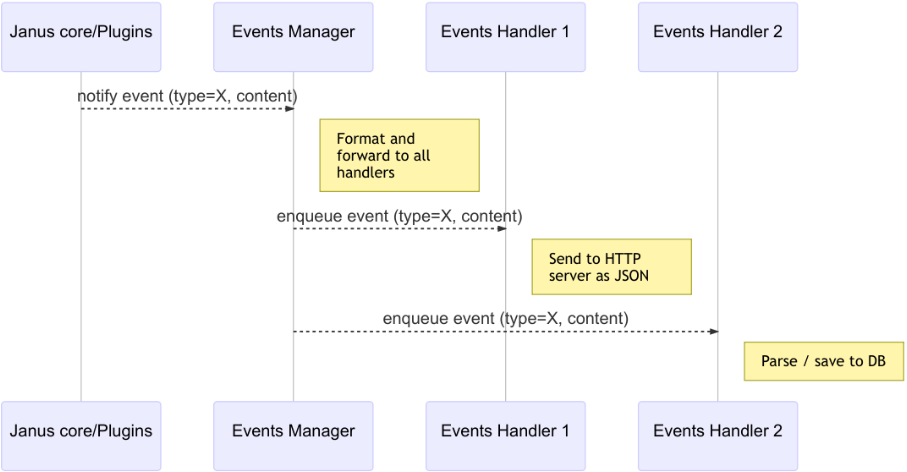
An asynchronous approach to monitoring/troubleshooting

- Admin API is cool, but is request/response...
  - Needs you to poll again to see changes
  - Information is lost when session/handle is gone
  - What about an asynchronous approach instead?
- A new mechanism: Event Handlers
  - Core and plugins generate events
    - Shared "header" (e.g., to identify target of event)
    - Different type of events allows for filtering
  - Custom modules can subscribe to and handle them
    - e.g., save to DB, send to external service, CDR, etc.
  - *Sample Event Handler* forwards JSON events via HTTP
    - Third-party plugins may provide integration with existing frameworks

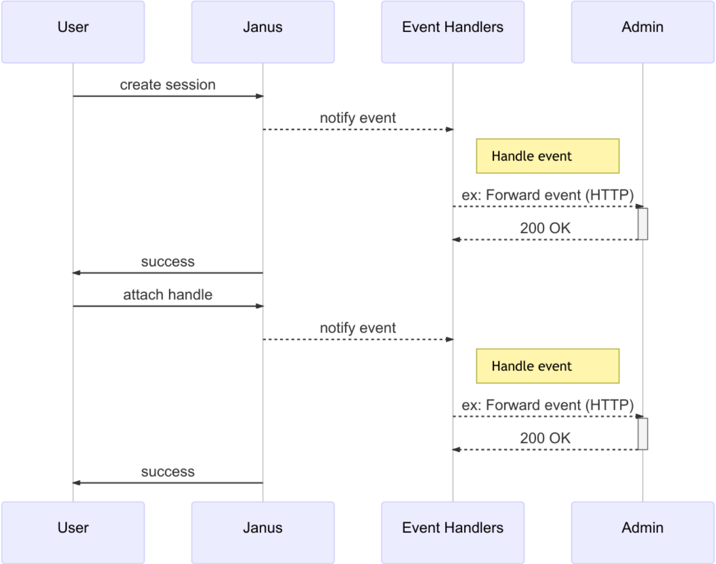
An asynchronous approach to monitoring/troubleshooting

- Admin API is cool, but is request/response...
  - Needs you to poll again to see changes
  - Information is lost when session/handle is gone
  - What about an asynchronous approach instead?
- A new mechanism: Event Handlers
  - Core and plugins generate events
    - Shared "header" (e.g., to identify target of event)
    - Different type of events allows for filtering
  - Custom modules can subscribe to and handle them
    - e.g., save to DB, send to external service, CDR, etc.
  - *Sample Event Handler* forwards JSON events via HTTP
    - Third-party plugins may provide integration with existing frameworks

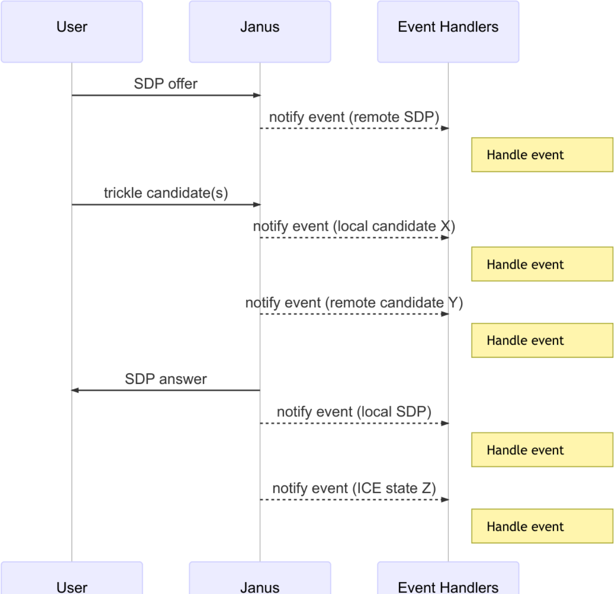
Event Handlers overview: routing and managing async events



Sample Event Handler example: notifying an Admin application



Sample Event Handler example: notifying an Admin application



Sofia SIP plugin: some event examples


```
{
    "type": 1,
    "timestamp": 1492609569131694,
    "session_id": 4285539655889579,
    "event": {
        "name": "created",
        "transport": {
            "transport": "janus.transport.http",
            "id": "0x7f06e40008c0"
        }
    }
}
```

Sofia SIP plugin: some event examples


```
{
    "type": 2,
    "timestamp": 1492609569145393,
    "session_id": 4285539655889579,
    "handle_id": 3904254637939378,
    "event": {
        "name": "attached",
        "plugin": "janus.plugin.sip",
        "opaque_id": "siptest-WvpckZtyzysY"
    }
}
```

Sofia SIP plugin: some event examples


```
{
    "type": 64,
    "timestamp": 1492609591790762,
    "session_id": 4285539655889579,
    "handle_id": 3904254637939378,
    "event": {
        "plugin": "janus.plugin.sip",
        "data": {
            "event": "sip-out",
            "sip": "REGISTER sip:192.168.1.80 SIP/2.0\r\n[..]"
        }
    }
}
```

Sofia SIP plugin: some event examples


```
{
    "type": 64,
    "timestamp": 1492609591791559,
    "session_id": 428553965588957 9,
    "handle_id": 3904254637939378,
    "event": {
        "plugin": "janus.plugin.sip",
        "data": {
            "event": "sip-in",
            "sip": "SIP/2.0 401 Unauthorized\r\n[..]"
        }
    }
}
```

Sofia SIP plugin: some event examples


```
{
    "type": 64,
    "timestamp": 1492609591791756,
    "session_id": 4285539655889579,
    "handle_id": 3904254637939378,
    "event": {
        "plugin": "janus.plugin.sip",
        "data": {
            "event": "sip-out",
            "sip": "REGISTER sip:192.168.1.80 SIP/2.0\r\n[..]"
        }
    }
}
```

Sofia SIP plugin: some event examples


```
{
    "type": 64,
    "timestamp": 1492609591792609,
    "session_id": 4285539655889579,
    "handle_id": 3904254637939378,
    "event": {
        "plugin": "janus.plugin.sip",
        "data": {
            "event": "sip-in",
            "sip": "SIP/2.0 200 OK\r\n[..]"
        }
    }
}
```

Sofia SIP plugin: some event examples


```
{
    "type": 64,
    "timestamp": 1492609591792635,
    "session_id": 4285539655889579,
    "handle_id": 3904254637939378,
    "event": {
        "plugin": "janus.plugin.sip",
        "data": {
            "event": "registered",
            "identity": "sip:janususer@192.168.1.80"
        }
    }
}
```

Sofia SIP plugin: some event examples


```
{
    "type": 8,
    "timestamp": 1492609620437498,
    "session_id": 4285539655889579,
    "handle_id": 3904254637939378,
    "event": {
        "owner": "remote",
        "jsep": {
            "type": "offer",
            "sdp": "v=0\r\no=- 2956402893317920386 2 IN IP4 127.0.0.1\r\n[...
        }
    }
}
```

Sofia SIP plugin: some event examples


```
{
    "type": 64,
    "timestamp": 1492609620438702,
    "session_id": 4285539655889579,
    "handle_id": 3904254637939378,
    "event": {
        "plugin": "janus.plugin.sip",
        "data": {
            "event": "calling",
            "callee": "sip:600@192.168.1.80",
            "call-id": "43wFB1UzBdAxm8blRExk92N",
            "sdp": "v=0\r\no=- 2956402893317920386 2 IN IP4 1.1.1.1\r\n[..]
        }
    }
}
```

Sofia SIP plugin: some event examples


```
{
    "type": 64,
    "timestamp": 1492609620439458,
    "session_id": 4285539655889579,
    "handle_id": 3904254637939378,
    "event": {
        "plugin": "janus.plugin.sip",
        "data": {
            "event": "sip-out",
            "sip": "INVITE sip:600@192.168.1.80 SIP/2.0\r\n[..]"
        }
    }
}
```

OpenSIPS'17

L. Miniero

Intro
WebRTC
SIP and WebRTC
Janus
Modules and APIs
Janus and SIP
Monitoring
Next steps

Sofia SIP plugin: some event examples


```
{
    "type": 64,
    "timestamp": 1492609620444829,
    "session_id": 4285539655889579,
    "handle_id": 3904254637939378,
    "event": {
        "plugin": "janus.plugin.sip",
        "data": {
            "event": "sip-in",
            "sip": "SIP/2.0 200 OK\r\n[..]"
        }
    }
}
```

Sofia SIP plugin: some event examples


```
{
    "type": 8,
    "timestamp": 1492609620445432,
    "session_id": 4285539655889579,
    "handle_id": 3904254637939378,
    "event": {
        "owner": "local",
        "jsep": {
            "type": "answer",
            "sdp": "v=0\r\no=root 2098992645 2098992645 IN IP4 192.168.1.80
        }
    }
}
```

Sofia SIP plugin: some event examples

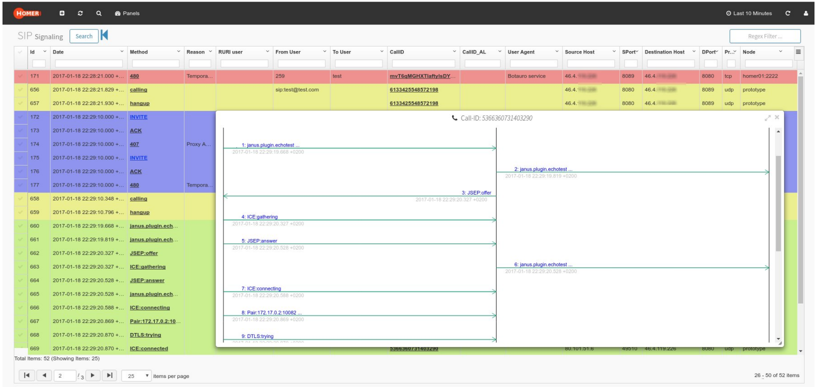

```
{
    "type": 64,
    "timestamp": 1492609620445448,
    "session_id": 4285539655889579,
    "handle_id": 3904254637939378,
    "event": {
        "plugin": "janus.plugin.sip",
        "data": {
            "event": "accepted",
            "call-id": "43wFB1UzBdAxm8blRExk92N",
            "username": "sip:600@192.168.1.80"
        }
    }
}
```

Sofia SIP plugin: some event examples


```
{
    "type": 16,
    "timestamp": 1492609620450203,
    "session_id": 4285539655889579,
    "handle_id": 3904254637939378,
    "event": {
        "ice": "connecting",
        "stream_id": 1,
        "component_id": 1
    }
}
```

A real-world example: Homer/HEP integration!

- Monitoring Janus sessions through a popular tool
  - Homer/HEP intercepts the events Janus sends
  - These events are correlated and displayed
- In case you want to know more...
  - https://fosdem.org/2017/schedule/event/janus/
  - https://fosdem.org/2017/schedule/event/homer/



What to do next?

- While the original SIP plugin is basically done, the others need love
  - The libre-based SIP plugin is still incomplete
  - The NoSIP plugin can definitely be improved as well
- What about SIP MESSAGE or INFO?
  - Could be done via DataChannels, or new requests
- Improve integration with HEPIC
  - Monitoring and troubleshooting are very important
  - A student is working exactly on that for his Master Thesis!

Help us improve this!

- Play with the plugins, more testing is important
- And, why not, help us improve plugins themselves!

What to do next?

- While the original SIP plugin is basically done, the others need love
    - The libre-based SIP plugin is still incomplete
    - The NoSIP plugin can definitely be improved as well
- What about SIP MESSAGE or INFO?
    - Could be done via DataChannels, or new requests
- Improve integration with HEPIC
    - Monitoring and troubleshooting are very important
    - A student is working exactly on that for his Master Thesis!

Help us improve this!

- Play with the plugins, more testing is important
- And, why not, help us improve plugins themselves!

What to do next?

- While the original SIP plugin is basically done, the others need love
  - The libre-based SIP plugin is still incomplete
  - The NoSIP plugin can definitely be improved as well
- What about SIP MESSAGE or INFO?
  - Could be done via DataChannels, or new requests
- Improve integration with HEPIC
  - Monitoring and troubleshooting are very important
  - A student is working exactly on that for his Master Thesis!

Help us improve this!

- Play with the plugins, more testing is important
- And, why not, help us improve plugins themselves!

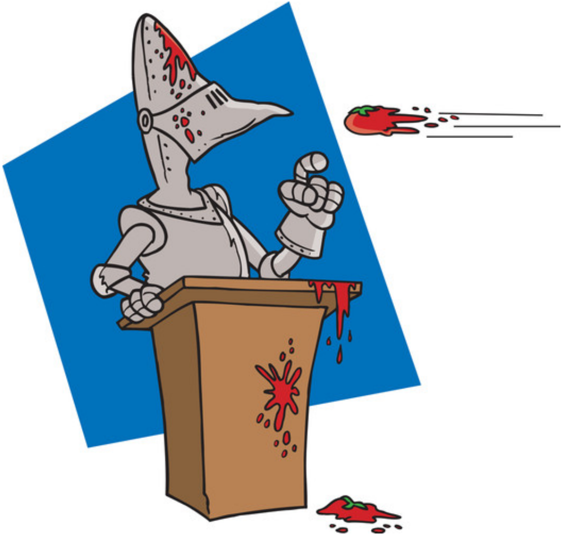
What to do next?

- While the original SIP plugin is basically done, the others need love
  - The libre-based SIP plugin is still incomplete
  - The NoSIP plugin can definitely be improved as well
- What about SIP MESSAGE or INFO?
  - Could be done via DataChannels, or new requests
- Improve integration with HEPIC
  - Monitoring and troubleshooting are very important
  - A student is working exactly on that for his Master Thesis!

Help us improve this!

- Play with the plugins, more testing is important
- And, why not, help us improve plugins themselves!

Questions? Comments?



https://twitter.com/elminiero