FOSDEM 2018

Writing a Janus plugin in Lua

C can be a scary world, let us come to the rescue!

Lorenzo Miniero

@elminiero

FOSDEM 2018 Real Time devroom

4th February 2018, Brussels

Remember Janus?

- 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

JANVS
WEBRTC GATEWAY

A quick recap: modular architecture

FOSDEM 2018

- The core only implements the WebRTC stack
  - JSEP/SDP, ICE, DTLS-SRTP, Data Channels, ...
- Plugins expose Janus API over different “transports”
  - Currently HTTP / WebSockets / RabbitMQ / Unix Sockets / MQTT
- “Application” logic implemented in plugins too
  - Users attach to plugins via the Janus core
  - The core handles the WebRTC stuff
  - Plugins route/manipulate the media/data
- Plugins can be combined on client side as “bricks”
  - Video SFU, Audio MCU, SIP gatewaying, broadcasting, etc.

“Pointers, pointers, everywhere...”

FOSDEM 2018

- Plugins a very powerful way to extend Janus, but...
  - ... everything in Janus is written in C! (well, except the web demos of course...)
- May be troublesome for some users to write their own (when really needed)

Let’s have a look at the Plugin API (1)

- Plugin initialization and information
    - **init()**: called when plugin is loaded
    - **destroy()**: called when Janus is shutting down
    - **get_api_compatibility()**: must return `JANUS_PLUGIN_API_VERSION`
    - **get_version()**: numeric version identifier (e.g., 3)
    - **get_version_string()**: verbose version identifier (e.g., “v1.0.1”)
    - **get_description()**: verbose description of the plugin (e.g., “This is my awesome plugin that does this and that”)
    - **get_name()**: short display name for your plugin (e.g., “My Awesome Plugin”)
    - **get_author()**: author of the plugin (e.g., “Meetecho s.r.l.”)
    - **get_package()**: unique package identifier for your plugin (e.g., “janus.plugin.myplugin”)

Let's have a look at the Plugin API (2)

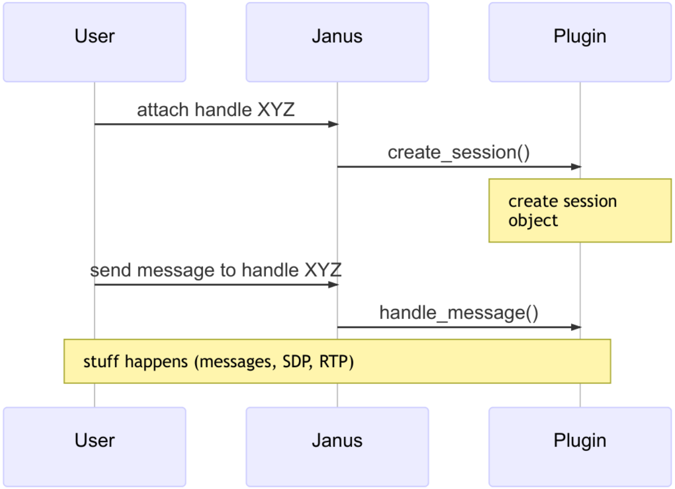
FOSDEM 2018

- Sessions management (callbacks invoked by the core)
  - **create_session()**: a user (session+handle) just attached to the plugin
  - **handle_message()**: incoming message/request (with or without a JSEP/SDP)
  - **setup_media()**: PeerConnection is now ready to be used
  - **incoming_rtp()**: incoming RTP packet
  - **incoming_rtcp()**: incoming RTCP message
  - **incoming_data()**: incoming DataChannel message
  - **slow_link()**: notification of problems on media path
  - **hangup_media()**: PeerConnection has been closed (e.g., DTLS alert)
  - **query_session()**: called to get plugin-specific info on a user session
  - **destroy_session()**: existing user gone (handle detached)

Let’s have a look at the Plugin API (3)

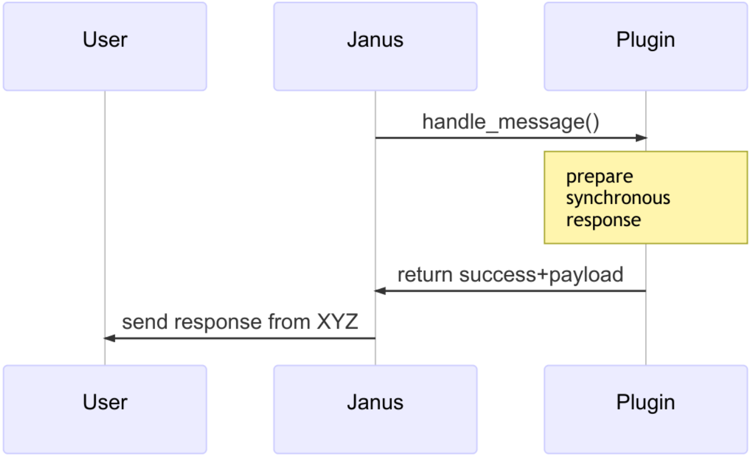
- Interaction with the core (methods invoked by the plugin)
  - **push_event()**: send the user a JSON message/event (with or without a JSEP/SDP)
  - **relay_rtp()**: send/relay the user an RTP packet
  - **relay_rtcp()**: send/relay the user an RTCP message
  - **relay_data()**: send/relay the user a DataChannel message
  - **close_pc()**: close the user’s PeerConnection
  - **end_session()**: close a user session (force-detach core handle)
  - **events_is_enabled()**: check whether the event handlers mechanism is enabled
  - **notify_event()**: notify an event to the registered and subscribed event handlers

Sequence diagrams (sessions mgmt)

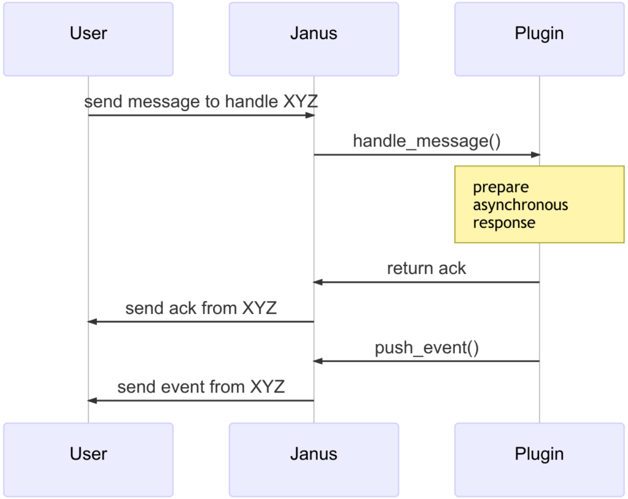


Sequence diagrams (sessions mgmt)

FOSDEM 2018

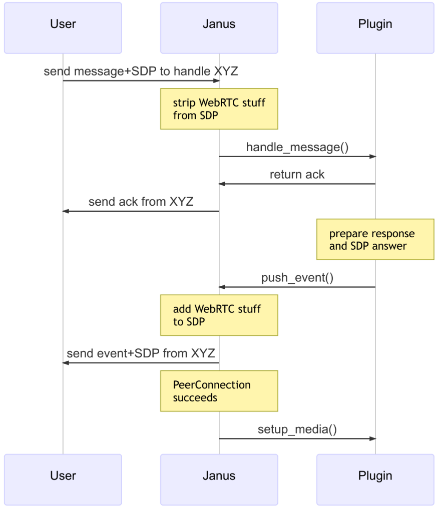


Sequence diagrams (sessions mgmt)

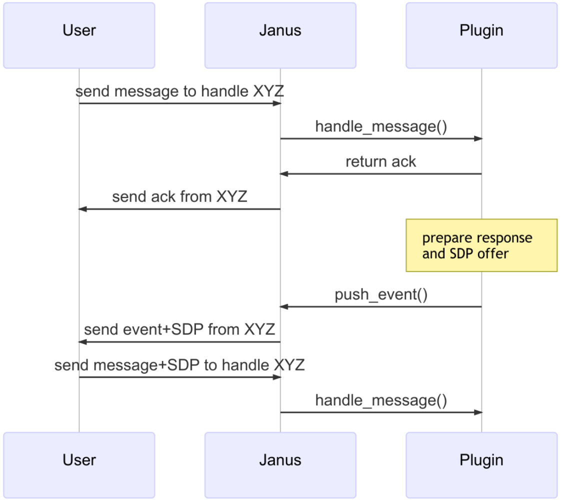


Sequence diagrams (sessions mgmt)

FOSDEM 2018

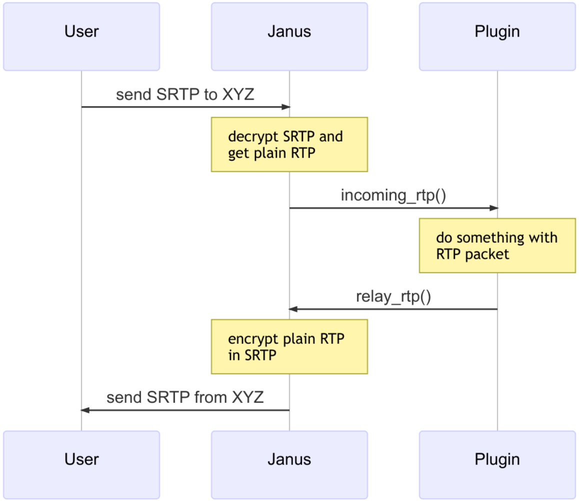


Sequence diagrams (sessions mgmt)



Sequence diagrams (sessions mgmt)

FOSDEM 2018



Writing a plugin in a different language

FOSDEM 2018

- All the above methods and callbacks need to be implemented in C
  - The core loads a shared module, and the core is written in C
- That said, does the logic really need to be written in C too?
  - As long as stubs are C, the core is happy
  - What these stubs do and return can be done in a different way
- All we need is provide hooks and bindings in C, and delegate the logic

Exactly what we did with the Lua plugin!

- https://github.com/meetecho/janus-gateway/pull/1033
- http://www.meetecho.com/blog/tutorial-writing-a-janus-video-call-plugin-in-lua/

Writing a plugin in a different language

FOSDEM 2018

- All the above methods and callbacks need to be implemented in C
  - The core loads a shared module, and the core is written in C
- That said, does the logic really need to be written in C too?
  - As long as stubs are C, the core is happy
  - What these stubs do and return can be done in a different way
- All we need is provide hooks and bindings in C, and delegate the logic

Exactly what we did with the Lua plugin!

- https://github.com/meetecho/janus-gateway/pull/1033
- http://www.meetecho.com/blog/tutorial-writing-a-janus-video-call-plugin-in-lua/

Writing a plugin in a different language

FOSDEM 2018

- All the above methods and callbacks need to be implemented in C
  - The core loads a shared module, and the core is written in C
- That said, does the logic really need to be written in C too?
  - As long as stubs are C, the core is happy
  - What these stubs do and return can be done in a different way
- All we need is provide hooks and bindings in C, and delegate the logic

Exactly what we did with the Lua plugin!

- https://github.com/meetecho/janus-gateway/pull/1033
- http://www.meetecho.com/blog/tutorial-writing-a-janus-video-call-plugin-in-lua/

Writing a plugin in a different language

- All the above methods and callbacks need to be implemented in C
  - The core loads a shared module, and the core is written in C
- That said, does the logic really need to be written in C too?
  - As long as stubs are C, the core is happy
  - What these stubs do and return can be done in a different way
- All we need is provide hooks and bindings in C, and delegate the logic

Exactly what we did with the Lua plugin!

- https://github.com/meetecho/janus-gateway/pull/1033
- http://www.meetecho.com/blog/tutorial-writing-a-janus-video-call-plugin-in-lua/

Janus Lua plugin: the basics

- Conceptually simple: C plugin, but with an embedded Lua state machine
  - Load a user-provided Lua script when initializing the plugin
  - Implement plugin callbacks in C, and have them call a Lua function
  - Implement core methods as Lua functions in C, that the Lua script can invoke
  - Track users/sessions via a unique ID that the C and Lua code share
- In theory, everything works (simple C↔Lua proxy)
  - The core sees a C plugin, but logic is handled in Lua
- In practice, that's not enough...
  1. Lua is single threaded (how to do things really asynchronously?)
  2. Handling RTP in Lua space would kill performance

Janus Lua plugin: the basics

FOSDEM 2018

- Conceptually simple: C plugin, but with an embedded Lua state machine
  - Load a user-provided Lua script when initializing the plugin
  - Implement plugin callbacks in C, and have them call a Lua function
  - Implement core methods as Lua functions in C, that the Lua script can invoke
  - Track users/sessions via a unique ID that the C and Lua code share
- In theory, everything works (simple C$\leftrightarrow$Lua proxy)
  - The core sees a C plugin, but logic is handled in Lua
- In practice, that's not enough...
  1. Lua is single threaded (how to do things really asynchronously?)
  2. Handling RTP in Lua space would kill performance

Janus Lua plugin: the basics

FOSDEM 2018

- Conceptually simple: C plugin, but with an embedded Lua state machine
  - Load a user-provided Lua script when initializing the plugin
  - Implement plugin callbacks in C, and have them call a Lua function
  - Implement core methods as Lua functions in C, that the Lua script can invoke
  - Track users/sessions via a unique ID that the C and Lua code share
- In theory, everything works (simple C↔Lua proxy)
  - The core sees a C plugin, but logic is handled in Lua
- In practice, that's not enough...
  1. Lua is single threaded (how to do things really asynchronously?)
  2. Handling RTP in Lua space would kill performance

Hooks and bindings (1)

- Plugin initialization and information

C		Lua
init()	⟶	**init()**
destroy()	⟶	**destroy()**
get_api_compatibility()	⟶	not needed
get_version()	⟶	**getVersion()**[1]
get_version_string()	⟶	**getVersionString()**[1]
get_description()	⟶	**getDescription()**[1]
get_name()	⟶	**getName()**[1]
get_author()	⟶	**getAuthor()**[1]
get_package()	⟶	**getPackage()**[1]

[1] Not really needed, so optional

Hooks and bindings (2)

- Sessions management (callbacks invoked by the core)

C		**Lua**
create_session()	$\longrightarrow$	createSession()
handle_message()	$\longrightarrow$	handleMessage()
setup_media()	$\longrightarrow$	setupMedia()
incoming_rtp()	$\longrightarrow$	incomingRtp()[2]
incoming_rtcp()	$\longrightarrow$	incomingRtcp()[2]
incoming_data()	$\longrightarrow$	incomingData()[2]
slow_link()	$\longrightarrow$	slowLink()
hangup_media()	$\longrightarrow$	hangupMedia()
query_session()	$\longrightarrow$	querySession()
destroy_session()	$\longrightarrow$	destroySession()

[2] Not the right way... more on this later!

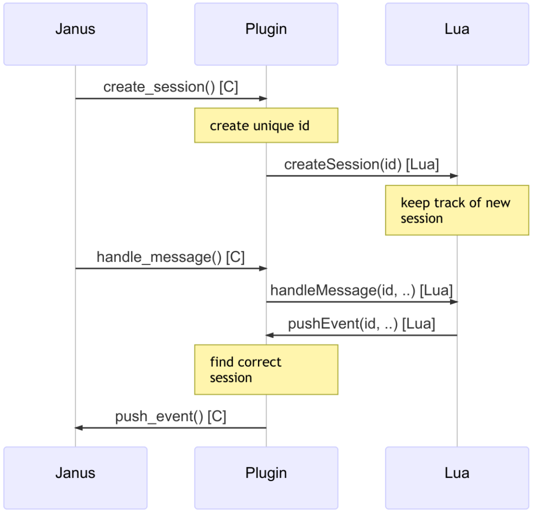
Hooks and bindings (3)

- Interaction with the core (methods invoked by the plugin)

C		Lua
push_event()	$\longleftarrow$	pushEvent()
relay_rtp()	$\longleftarrow$	relayRtp()[3]
relay_rtcp()	$\longleftarrow$	relayRtcp()[3]
relay_data()	$\longleftarrow$	relayData()[3]
close_pc()	$\longleftarrow$	closePc()
end_session()	$\longleftarrow$	endSession()
events_is_enabled()	$\longleftarrow$	eventsIsEnabled()[4]
notify_event()	$\longleftarrow$	notifyEvent()

[3]Not the right way... more on this later!
[4]Not really needed, so optional

Example of hooks and bindings



FOSDEM 2018

Asynchronous logic in the Lua plugin

FOSDEM 2018

- We've seen how asynchronous events are heavily used by plugins
  - Asynchronous message response, negotiations, etc.
  - Most out-of-the-box Janus plugins are thread based
- Lua is single threaded, though...
  - Coroutines can be seen as threads, but they aren't
  - Access to the Lua state isn't thread safe either

Solution: a C "scheduler"

A dedicated thread in the C code of the plugin acts as scheduler

- The Lua script queues tasks, and "pokes" the scheduler via `pokeScheduler()`
- `pokeScheduler()` is implemented in C, and wakes the scheduler (queue)
- The C scheduler calls `resumeScheduler()` in Lua as a coroutine

Asynchronous logic in the Lua plugin

- We've seen how asynchronous events are heavily used by plugins
  - Asynchronous message response, negotiations, etc.
  - Most out-of-the-box Janus plugins are thread based
- Lua is single threaded, though...
  - Coroutines can be seen as threads, but they aren't
  - Access to the Lua state isn't thread safe either

Solution: a C "scheduler"

A dedicated thread in the C code of the plugin acts as scheduler

- The Lua script queues tasks, and "pokes" the scheduler via `pokeScheduler()`
- `pokeScheduler()` is implemented in C, and wakes the scheduler (queue)
- The C scheduler calls `resumeScheduler()` in Lua as a coroutine

Asynchronous logic in the Lua plugin

- We've seen how asynchronous events are heavily used by plugins
  - Asynchronous message response, negotiations, etc.
  - Most out-of-the-box Janus plugins are thread based
- Lua is single threaded, though...
  - Coroutines can be seen as threads, but they aren't
  - Access to the Lua state isn't thread safe either

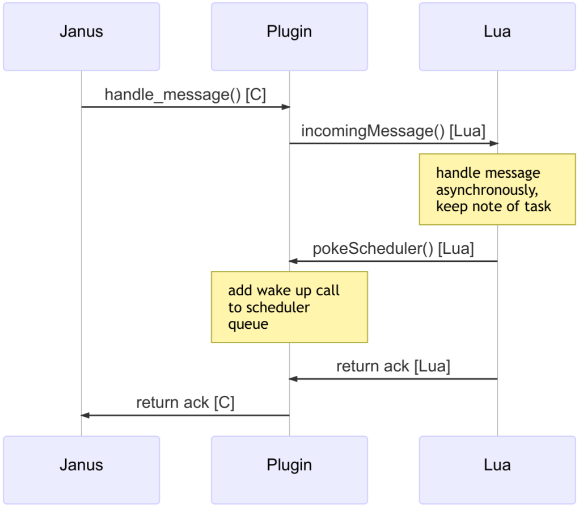
Solution: a C "scheduler"

A dedicated thread in the C code of the plugin acts as scheduler

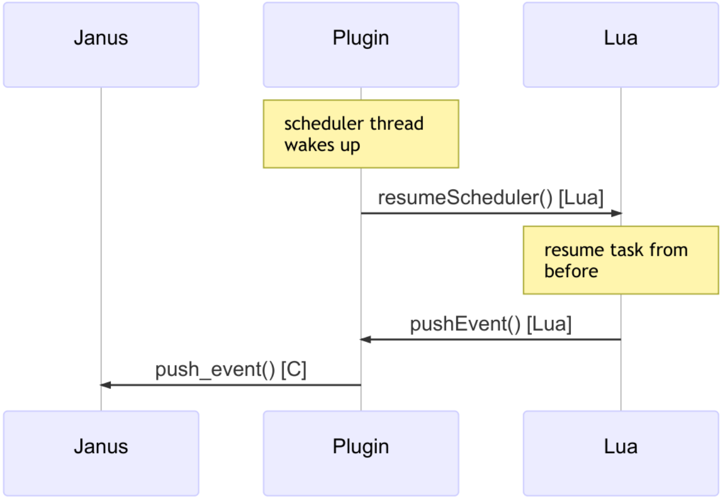
- The Lua script queues tasks, and "pokes" the scheduler via `pokeScheduler()`
- `pokeScheduler()` is implemented in C, and wakes the scheduler (queue)
- The C scheduler calls `resumeScheduler()` in Lua as a coroutine

Scheduler example: asynchronous reply

FOSDEM 2018



Scheduler example: asynchronous reply



Timed callbacks in the Lua plugin

FOSDEM 2018

- `pokeScheduler()` and `resumeScheduler()` are great but have limits
  - No arguments can be passed to the scheduler
  - You need to keep track of tasks yourself
  - The `resumeScheduler()` function is called as soon as possible
- You may want to trigger a callback (with a parameter?) after some time instead
  - e.g., “call secondsPassed(5) in 5 seconds”

Solution: a new `timeCallback` function as a C hook

A timed source in the C code of the plugin acts as triggerer

- The Lua script times a callback via `timeCallback()`
- `timeCallback()` is implemented in C, and creates a timed source
- The source fires and calls the specified callback in Lua as a coroutine

Timed callbacks in the Lua plugin

- `pokeScheduler()` and `resumeScheduler()` are great but have limits
  - No arguments can be passed to the scheduler
  - You need to keep track of tasks yourself
  - The `resumeScheduler()` function is called as soon as possible
- You may want to trigger a callback (with a parameter?) after some time instead
  - e.g., “call secondsPassed(5) in 5 seconds”

Solution: a new `timeCallback` function as a C hook

A timed source in the C code of the plugin acts as triggerer

- The Lua script times a callback via `timeCallback()`
- `timeCallback()` is implemented in C, and creates a timed source
- The source fires and calls the specified callback in Lua as a coroutine

Timed callbacks in the Lua plugin

FOSDEM 2018

- `pokeScheduler()` and `resumeScheduler()` are great but have limits
  - No arguments can be passed to the scheduler
  - You need to keep track of tasks yourself
  - The `resumeScheduler()` function is called as soon as possible
- You may want to trigger a callback (with a parameter?) after some time instead
  - e.g., “call secondsPassed(5) in 5 seconds”

Solution: a new `timeCallback` function as a C hook

A timed source in the C code of the plugin acts as triggerer

- The Lua script times a callback via `timeCallback()`
- `timeCallback()` is implemented in C, and creates a timed source
- The source fires and calls the specified callback in Lua as a coroutine

What about RTP/RTCP/data?

- As we pointed out, handling data in Lua drags performance down
    - While hooks are there, there's a cost in going from C to Lua and viceversa
    - Lua state is single threaded, meaning relaying would have a bottleneck
- Arguably this is more of an issue for RTP, less so for RTCP and data
    - ... unless RTCP and data messages are very frequent too

Solution: only configuring routing in Lua (actual relaying still in C)

The C code routes the media according to dynamic changes coming from Lua

- `addRecipient()` and `removeRecipient()` dictate who receives user's media
- `configureMedium()` opens/closes valves for outgoing/incoming media
- Helper methods (`setBitrate()`, `sendPli()`, etc.) do the rest

What about RTP/RTCP/data?

FOSDEM 2018

- As we pointed out, handling data in Lua drags performance down
  - While hooks are there, there's a cost in going from C to Lua and viceversa
  - Lua state is single threaded, meaning relaying would have a bottleneck
- Arguably this is more of an issue for RTP, less so for RTCP and data
  - ... unless RTCP and data messages are very frequent too

Solution: only configuring routing in Lua (actual relaying still in C)

The C code routes the media according to dynamic changes coming from Lua

- `addRecipient()` and `removeRecipient()` dictate who receives user's media
- `configureMedium()` opens/closes valves for outgoing/incoming media
- Helper methods (`setBitrate()`, `sendPli()`, etc.) do the rest

What about RTP/RTCP/data?

- As we pointed out, handling data in Lua drags performance down
  - While hooks are there, there's a cost in going from C to Lua and viceversa
  - Lua state is single threaded, meaning relaying would have a bottleneck
- Arguably this is more of an issue for RTP, less so for RTCP and data
  - ... unless RTCP and data messages are very frequent too

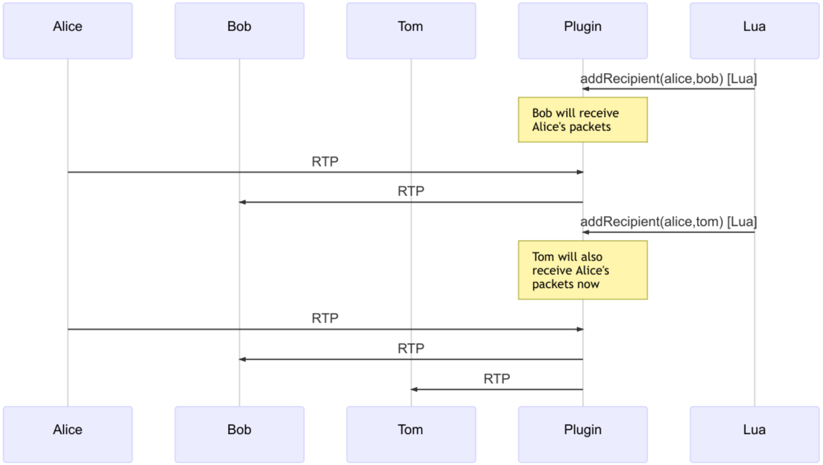
Solution: only configuring routing in Lua (actual relaying still in C)

The C code routes the media according to dynamic changes coming from Lua

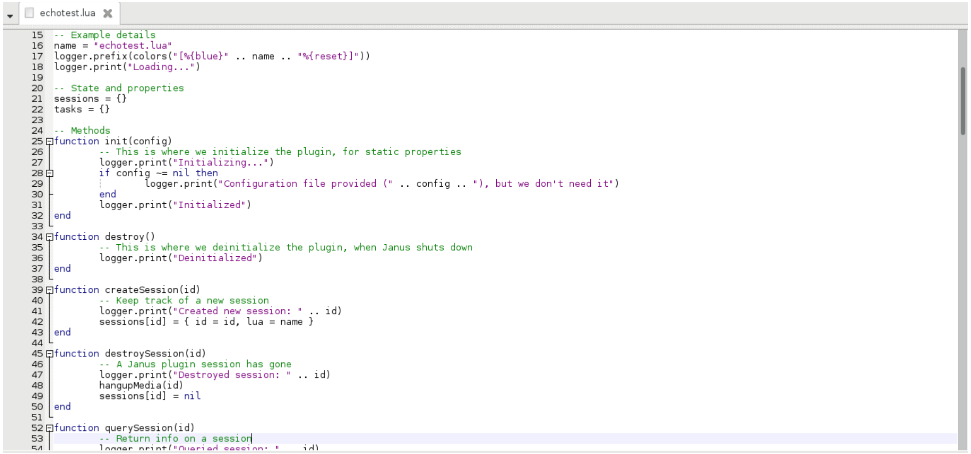
- `addRecipient()` and `removeRecipient()` dictate who receives user's media
- `configureMedium()` opens/closes valves for outgoing/incoming media
- Helper methods (`setBitrate()`, `sendPli()`, etc.) do the rest

Routing media (SFU example)

FOSDEM 2018


Alice
Bob
Tom
Plugin
Lua
addRecipient(alice,bob) [Lua]
Bob will receive Alice's packets
RTP
RTP
addRecipient(alice,tom) [Lua]
Tom will also receive Alice's packets now
RTP
RTP
RTP
Alice
Bob
Tom
Plugin
Lua

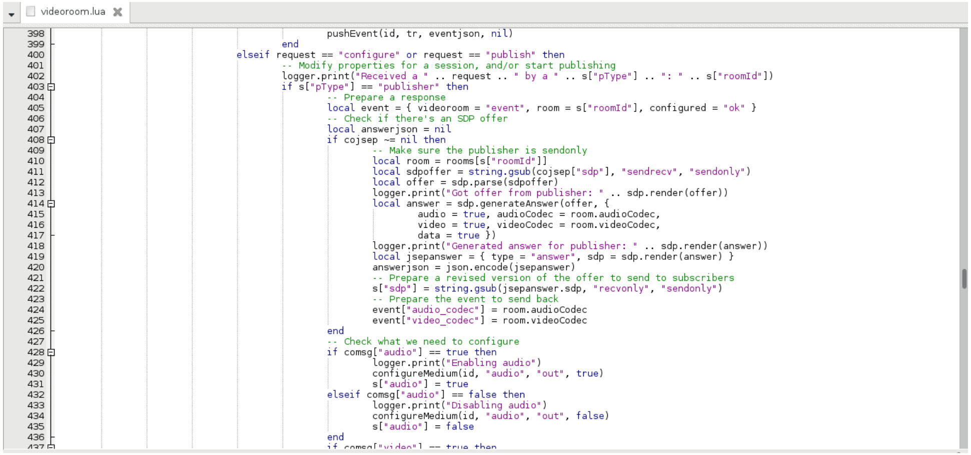
A few examples: EchoTest clone



```
-- Example details
name = "echotest.lua"
logger.prefix(colors("[%{blue}" .. name .. "%{reset}]"))
logger.print("Loading...")

-- State and properties
sessions = {}
tasks = {}

-- Methods
function init(config)
        -- This is where we initialize the plugin, for static properties
        logger.print("Initializing...")
        if config ~= nil then
                logger.print("Configuration file provided (" .. config .. "), but we don't need it")
        end
        logger.print("Initialized")
end

function destroy()
        -- This is where we deinitialize the plugin, when Janus shuts down
        logger.print("Deinitialized")
end

function createSession(id)
        -- Keep track of a new session
        logger.print("Created new session: " .. id)
        sessions[id] = { id = id, lua = name }
end

function destroySession(id)
        -- A Janus plugin session has gone
        logger.print("Destroyed session: " .. id)
        hangupMedia(id)
        sessions[id] = nil
end

function querySession(id)
        -- Return info on a session
        logger.print("Queried session: " .. id)
```

Something trickier: VideoRoom clone

FOSDEM 2018

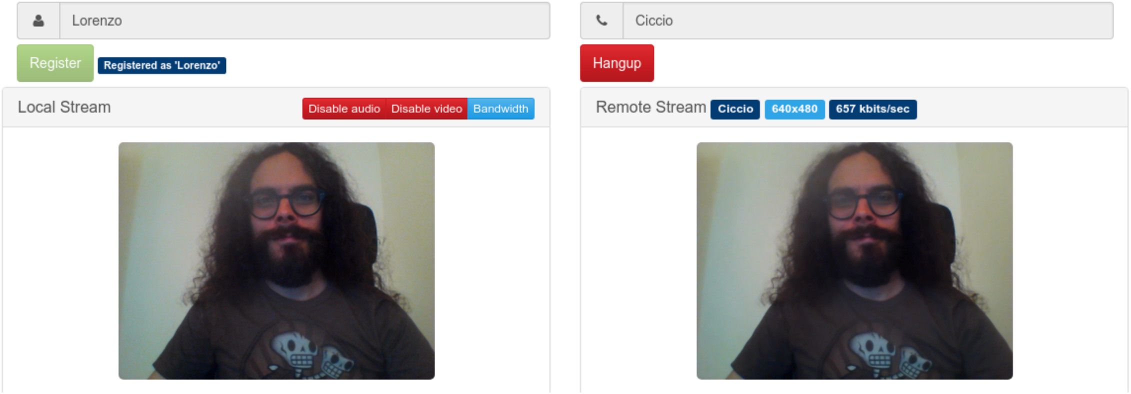


videoroom.lua

```lua
                        pushEvent(id, tr, eventjson, nil)
                    end
                elseif request == "configure" or request == "publish" then
                    -- Modify properties for a session, and/or start publishing
                    logger.print("Received a " .. request .. " by a " .. s["pType"] .. ": " .. s["roomId"])
                    if s["pType"] == "publisher" then
                        -- Prepare a response
                        local event = { videoroom = "event", room = s["roomId"], configured = "ok" }
                        -- Check if there's an SDP offer
                        local answerjson = nil
                        if cojsep ~= nil then
                            -- Make sure the publisher is sendonly
                            local room = rooms[s["roomId"]]
                            local sdpoffer = string.gsub(cojsep["sdp"], "sendrecv", "sendonly")
                            local offer = sdp.parse(sdpoffer)
                            logger.print("Got offer from publisher: " .. sdp.render(offer))
                            local answer = sdp.generateAnswer(offer, {
                                audio = true, audioCodec = room.audioCodec,
                                video = true, videoCodec = room.videoCodec,
                                data = true })
                            logger.print("Generated answer for publisher: " .. sdp.render(answer))
                            local jsepanswer = { type = "answer", sdp = sdp.render(answer) }
                            answerjson = json.encode(jsepanswer)
                            -- Prepare a revised version of the offer to send to subscribers
                            s["sdp"] = string.gsub(jsepanswer.sdp, "recvonly", "sendonly")
                            -- Prepare the event to send back
                            event["audio_codec"] = room.audioCodec
                            event["video_codec"] = room.videoCodec
                        end
                        -- Check what we need to configure
                        if comsg["audio"] == true then
                            logger.print("Enabling audio")
                            configureMedium(id, "audio", "out", true)
                            s["audio"] = true
                        elseif comsg["audio"] == false then
                            logger.print("Disabling audio")
                            configureMedium(id, "audio", "out", false)
                            s["audio"] = false
                        end
                        if comsg["video"] == true then
```

VideoCall clone: a tutorial

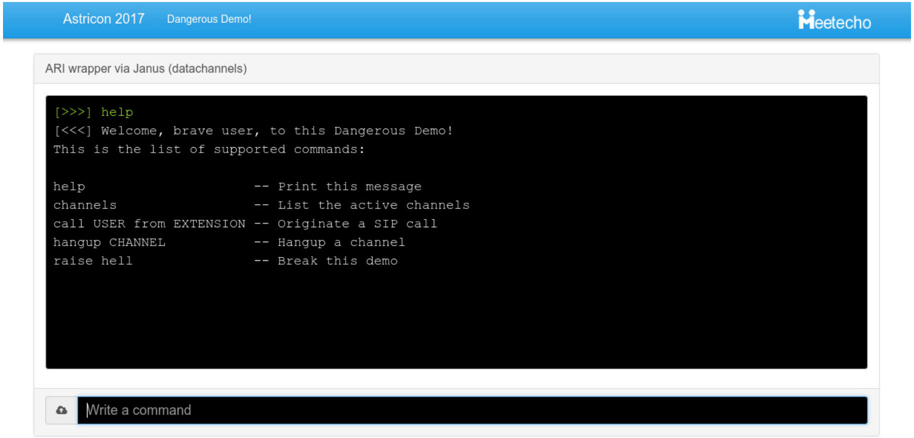
FOSDEM 2018



http://www.meetecho.com/blog/tutorial-writing-a-janus-video-call-plugin-in-lua/

Astricon 2017 Dangerous Demo

FOSDEM 2018

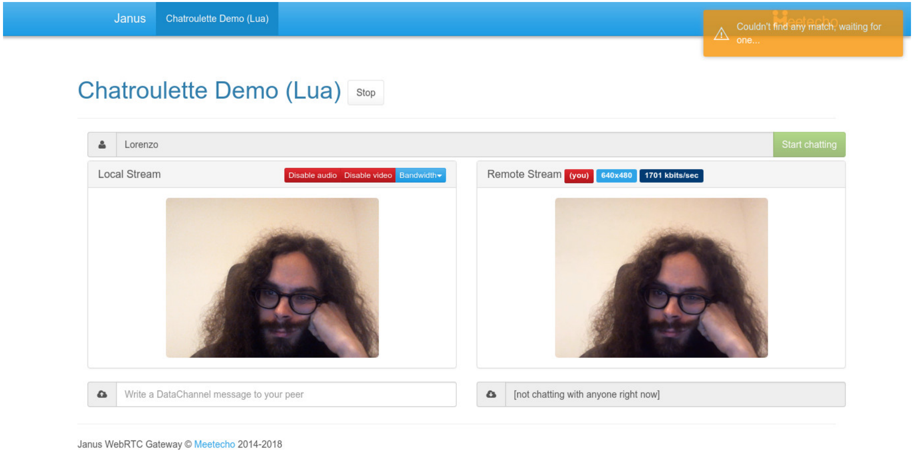


```
[>>>] help
[<<<] Welcome, brave user, to this Dangerous Demo!
This is the list of supported commands:

help                      -- Print this message
channels                  -- List the active channels
call USER from EXTENSION -- Originate a SIP call
hangup CHANNEL            -- Hangup a channel
raise hell                -- Break this demo
```

https://gist.github.com/lminiero/9aeeda1be501fb636cad0c8057c6e076

One more cool example... Chatroulette!

FOSDEM 2018


Janus
Chatroulette Demo (Lua)

Couldn't find any match, waiting for one...

Chatroulette Demo (Lua) Stop

Lorenzo
Start chatting

Local Stream
Disable audio Disable video Bandwidth

Remote Stream (you) 640x480 1701 kbits/sec

Write a DataChannel message to your peer

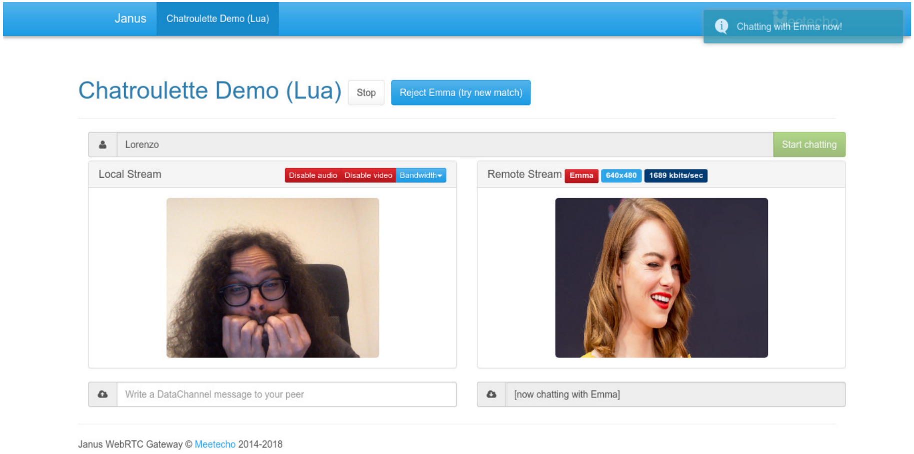
[not chatting with anyone right now]

Janus WebRTC Gateway © Meetecho 2014-2018

https://github.com/lminiero/fosdem18-januslua

One more cool example... Chatroulette!

FOSDEM 2018


Janus
Chatroulette Demo (Lua)
Chatting with Emma now!

Chatroulette Demo (Lua)
Stop
Reject Emma (try new match)

Lorenzo
Start chatting

Local Stream
Disable audio Disable video Bandwidth

Remote Stream Emma 640x480 1689 kbits/sec

Write a DataChannel message to your peer

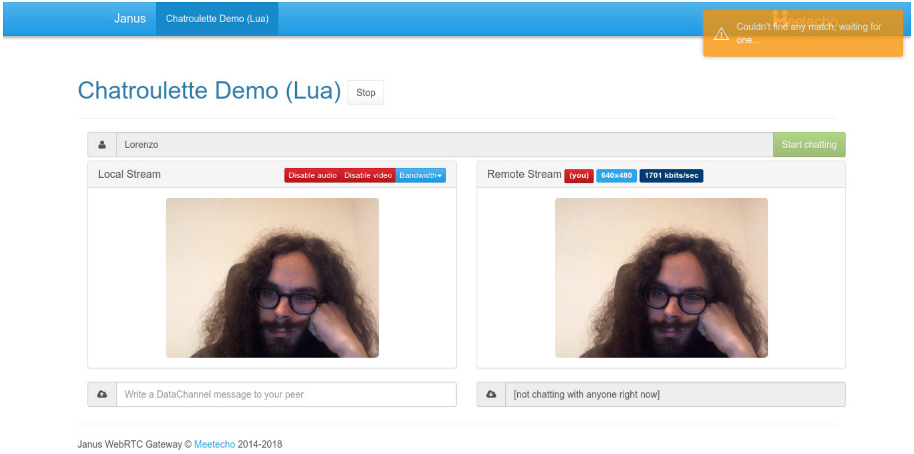
[now chatting with Emma]

Janus WebRTC Gateway © Meetecho 2014-2018

https://github.com/lminiero/fosdem18-januslua

One more cool example... Chatroulette!

FOSDEM 2018



Couldn't find any match, waiting for one...

Chatroulette Demo (Lua)

Lorenzo

Local Stream

Remote Stream

Write a DataChannel message to your peer

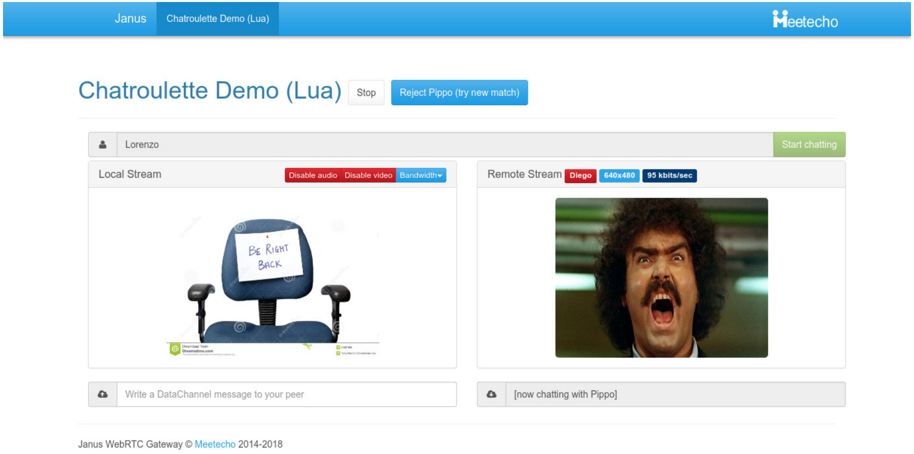
[not chatting with anyone right now]

Janus WebRTC Gateway © Meetecho 2014-2018

https://github.com/lminiero/fosdem18-januslua

One more cool example... Chatroulette!

FOSDEM 2018



Janus | Chatroulette Demo (Lua) | Meetecho

Chatroulette Demo (Lua)

Stop | Reject Pippo (try new match)

Lorenzo | Start chatting

Local Stream | Disable audio | Disable video | Bandwidth

Remote Stream | Diego | 640x480 | 95 kbits/sec

Write a DataChannel message to your peer

[now chatting with Pippo]

Janus WebRTC Gateway © Meetecho 2014-2018

What to do next?

FOSDEM 2018

- Integrate advanced features recently added to master
  - RTP injection/forwarding, simulcasting, VP9 SVC, ...
- General improvements may be needed once it's used more
  - Based on `refcount` branch, which is experimental itself
- Do Lua-based Transport plugins and Event Handlers make any sense?
  - They're plugins (shared objects) too, after all...
- Why not, write new plugins for other programming languages!
  - Most hooks are already there, after all, we only need bindings
  - A potential "candidate": JavaScript (e.g., with http://duktape.org/)

Help us improve this!

- Play with it, more testing is important
- Write your own applications, or help expand the Lua plugin itself!

What to do next?

FOSDEM 2018

- Integrate advanced features recently added to master
  - RTP injection/forwarding, simulcasting, VP9 SVC, ...
- General improvements may be needed once it’s used more
  - Based on `refcount` branch, which is experimental itself
- Do Lua-based Transport plugins and Event Handlers make any sense?
  - They’re plugins (shared objects) too, after all...
- Why not, write new plugins for other programming languages!
  - Most hooks are already there, after all, we only need bindings
  - A potential “candidate”: JavaScript (e.g., with http://duktape.org/)

Help us improve this!

- Play with it, more testing is important
- Write your own applications, or help expand the Lua plugin itself!

What to do next?

- Integrate advanced features recently added to master
  - RTP injection/forwarding, simulcasting, VP9 SVC, ...
- General improvements may be needed once it's used more
  - Based on `refcount` branch, which is experimental itself
- Do Lua-based Transport plugins and Event Handlers make any sense?
  - They're plugins (shared objects) too, after all...
- Why not, write new plugins for other programming languages!
  - Most hooks are already there, after all, we only need bindings
  - A potential "candidate": JavaScript (e.g., with http://duktape.org/)

Help us improve this!

- Play with it, more testing is important
- Write your own applications, or help expand the Lua plugin itself!

What to do next?

FOSDEM 2018

- Integrate advanced features recently added to master
  - RTP injection/forwarding, simulcasting, VP9 SVC, ...
- General improvements may be needed once it's used more
  - Based on `refcount` branch, which is experimental itself
- Do Lua-based Transport plugins and Event Handlers make any sense?
  - They're plugins (shared objects) too, after all...
- Why not, write new plugins for other programming languages!
  - Most hooks are already there, after all, we only need bindings
  - A potential "candidate": JavaScript (e.g., with http://duktape.org/)

Help us improve this!

- Play with it, more testing is important
- Write your own applications, or help expand the Lua plugin itself!

What to do next?

FOSDEM 2018

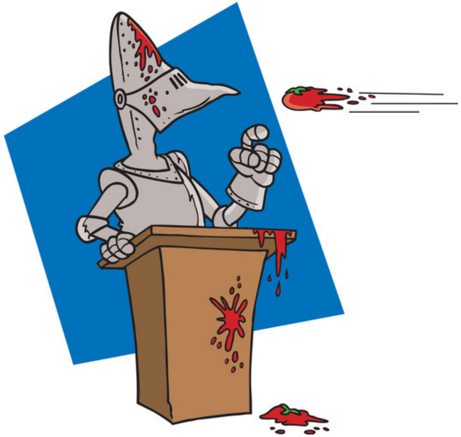
- Integrate advanced features recently added to master
  - RTP injection/forwarding, simulcasting, VP9 SVC, ...
- General improvements may be needed once it's used more
  - Based on `refcount` branch, which is experimental itself
- Do Lua-based Transport plugins and Event Handlers make any sense?
  - They're plugins (shared objects) too, after all...
- Why not, write new plugins for other programming languages!
  - Most hooks are already there, after all, we only need bindings
  - A potential "candidate": JavaScript (e.g., with http://duktape.org/)

Help us improve this!

- Play with it, more testing is important
- Write your own applications, or help expand the Lua plugin itself!

Thanks! Questions? Comments?

FOSDEM 2018



Get in touch!

- https://twitter.com/elminiero
- https://twitter.com/meetecho
- http://www.meetecho.com