SA8 T1 – Meeting 3

JANUS Basics and Applications

Rui Ribeiro

WebRTC Task Member

IP Video Services Manager, FCT|FCCN

Stockolm, 27 Oct 2015

28/10/2015

Meetecho JANUS

https://vimeo.com/102232226

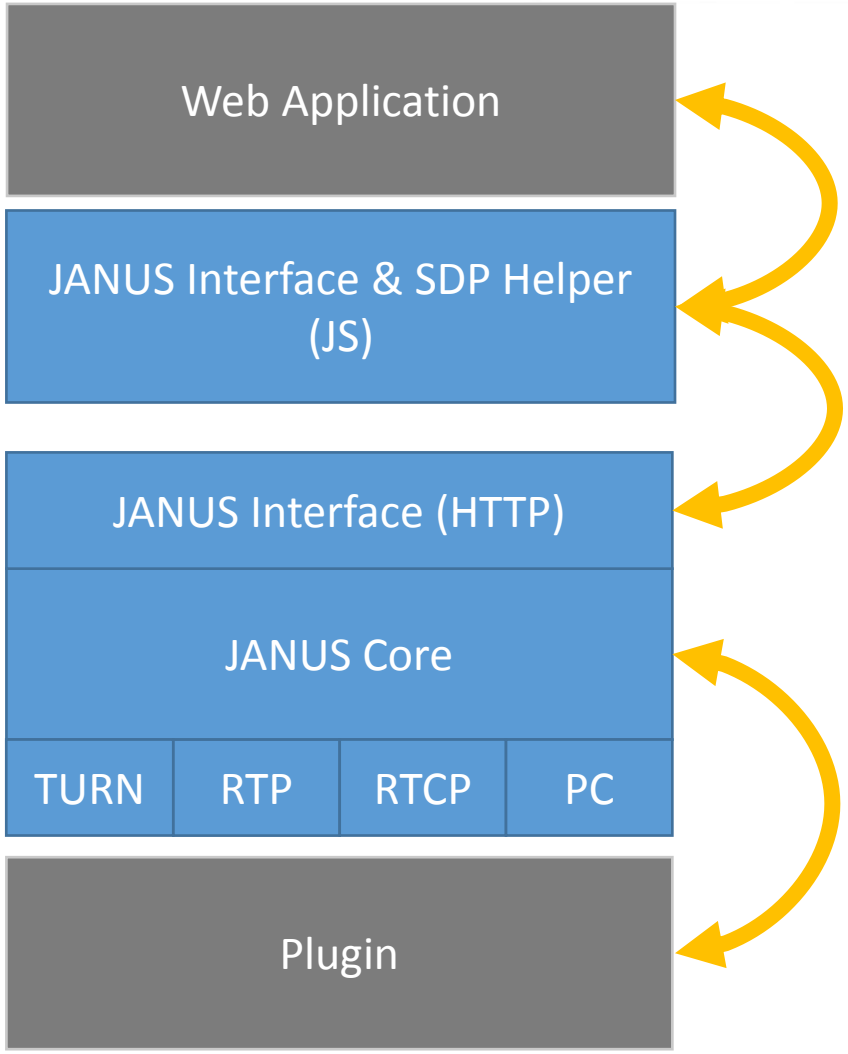
JANUS - General purpose WebRTC Gateway

- The Janus WebRTC Gateway has been conceived as a general purpose gateway.
- It doesn't provide any functionality *per se* other than implementing the means to set up a WebRTC media communication with a browser, exchanging JSON messages with it, and relaying RTP/RTCP and messages between browsers and the server-side application logic they're attached to.
- Any specific feature/application needs to be implemented in server side plugins, that browsers can then contact via the gateway to take advantage of the functionality they provide.
- Example of such plugins can be implementations of applications like echo tests, conference bridges, media recorders, SIP gateways and the like.
- Small footprint (C implementation)
- Pluggable modules philosophy

Demo Applications

Echo Test	A simple Echo Test demo, with knobs to control the bitrate.
Streaming	A media Streaming demo, with sample live and on-demand streams.
Video Call	A Video Call demo, a bit like AppRTC but with media passing through the gateway.
SIP Gateway	A SIP Gateway demo, allowing you to register at a SIP server and start/receive calls.
Video Room	A videoconferencing demo, allowing you to join a video room with up to six users.
Audio Room	An audio mixing/bridge demo, allowing you join an Audio Room room.
Voice Mail	A simple audio recorder demo, returning an .opus file after 10 seconds.
Recorder/Playout	A demo to record audio/video messages, and subsequently replay them through WebRTC.
Screen Sharing	A webinar-like screen sharing session, based on the Video Room plugin.

Architecture



Javascript API

https://github.com/meetecho/janus-gateway/blob/master/html/janus.js

REST API / JSON

https://github.com/meetecho/janus-gateway/blob/master/janus.c
@ janus_process_incoming_request

Janus Plugin API

https://github.com/meetecho/janus-gateway/blob/master/plugins/plugin.h

Javascript JANUS API

```
janus = new Janus(
  {
    server: janus_api_url
    iceServers: [{url: "turn:yourturnserver.com:3478"}],
    success: function() {
      janus.attach( {
        plugin: "janus.plugin.SERVICE",
        success: function(pluginHandle) { … },
        error: function(error) { … },
        consentDialog: function(on) { … },
        onmessage: function(msg, jsep) { … },
        onlocalstream: function(stream) { … },
        onremotestream: function(stream) { … },
        ondataopen: function(data) { … },
        ondata: function(data) { … },
        oncleanup: function() { … }
      } )
    error: function(error) { … },
    destroyed: function() { … }
  }
```

Event Handlers
(receiving events from JANUS)

Javascript JANUS API

JANUS Methods
(sending messages to JANUS)

- getId()
  - returns the unique handle identifier;
- getPlugin()
  - returns the unique package name of the attached plugin;
- send(parameters)
  - sends a message (with or without a jsep to negotiate a PeerConnection) to the plugin;
- createOffer(callbacks)
  - asks the library to create a WebRTC compliant OFFER;
- createAnswer(callbacks)
  - asks the library to create a WebRTC compliant ANSWER;
- handleRemoteJsep(callbacks)
  - asks the library to handle an incoming WebRTC compliant session description;
- dtmf(parameters)
  - sends a DTMF tone on the PeerConnection;
- data(parameters)
  - sends data through the Data Channel, if available;
- getBitrate()
  - gets a verbose description of the currently received stream bitrate (only available on Chrome, for now);
- hangup()
  - tells the library to close the PeerConnection;
- detach(parameters)
  - detaches from the plugin and destroys the handle, tearing down the related PeerConnection if it exists.

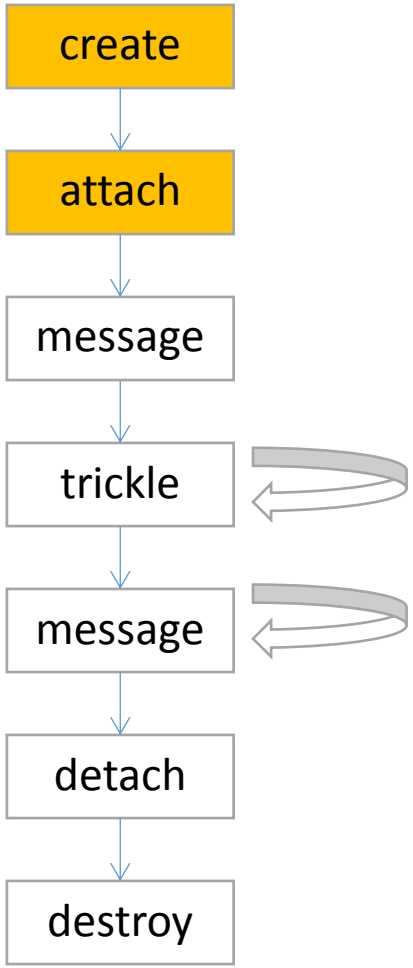
Javascript JANUS API

```
application = null

janus = new Janus(
  {
    server: janus_api_url
    iceServers: [{url: "turn:yourturnserver.com:3478"}],
    success: function() {
      janus.attach( {
        plugin: "janus.plugin.SERVICE",
        success: function(pluginHandle) { application = pluginHandle; … },
    […]
}

application.getPlugin();
application.getId();
application.send({"message": body});
…
```

JANUS REST API

- **info** returns info about janus
- **ping** responds pong
- **keepalive** keep session alive
- *rmqtest* *(Rabbit MQ Test)*
- **create** create session
- **attach** service/plugin
- **destroy** session
- **detach** seervice/plugin
- **message** multipurpose message transfer
- **trickle** find connectivity solution

JANUS REST API Sequence 1/5



▼General
Remote Address: 143.225.229.138:443
Request URL: https://janus.conf.meetecho.com/janus
Request Method: POST
Status Code: 200 OK

▶Response Headers (6)
▶Request Headers (12)
▼Request Payload view source

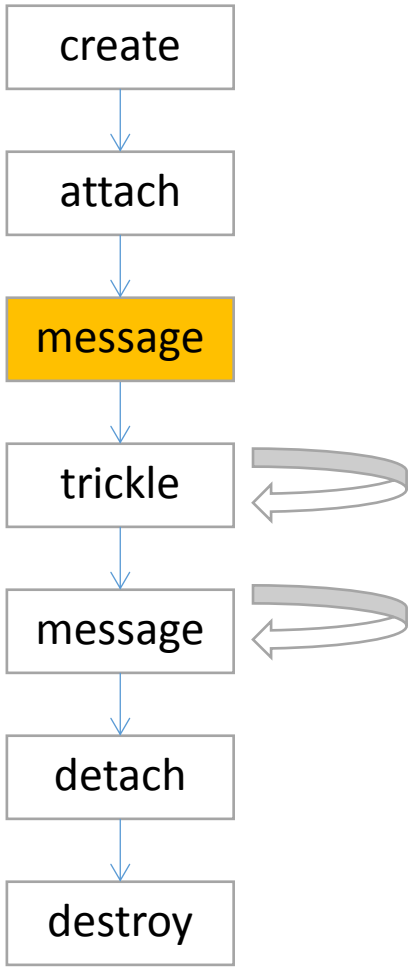
```
▼{janus: "create", transaction: "DBNLQFDW0c3v"}
    janus: "create"
    transaction: "DBNLQFDW0c3v"
```

▼General
Remote Address: 143.225.229.138:443
Request URL: https://janus.conf.meetecho.com/janus/724131890
Request Method: POST
Status Code: 200 OK

▶Response Headers (6)
▶Request Headers (12)
▼Request Payload view source

```
▼{janus: "attach", plugin: "janus.plugin.echotest", transaction: "NFU079nRYZQc"}
    janus: "attach"
    plugin: "janus.plugin.echotest"
    transaction: "NFU079nRYZQc"
```

JANUS REST API Sequence 2/5



▼General
Remote Address: 143.225.229.138:443
Request URL: https://janus.conf.meetecho.com/janus/724131890/4226221616
Request Method: POST
Status Code: 200 OK

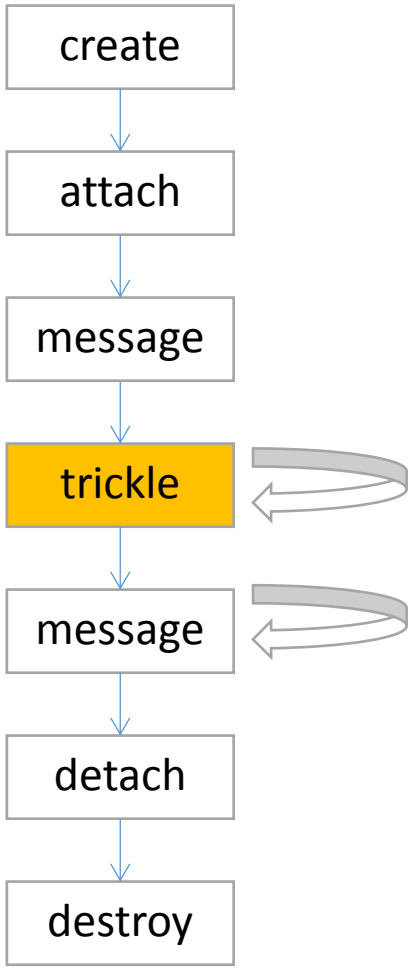
▶Response Headers (6)

▶Request Headers (12)

▼Request Payload view source

```
▼{janus: "message", body: {audio: true, video: true}, transaction: "wCV9ip71Zui7"}
  ▼body: {audio: true, video: true}
      audio: true
      video: true
    janus: "message"
    transaction: "wCV9ip71Zui7"
```

JANUS REST API Sequence 3/5



General
Remote Address: 143.225.229.138:443
Request URL: https://janus.conf.meetecho.com/janus/724131890/4226221616
Request Method: POST
Status Code: 200 OK

Response Headers (6)
Request Headers (12)
Request Payload view source

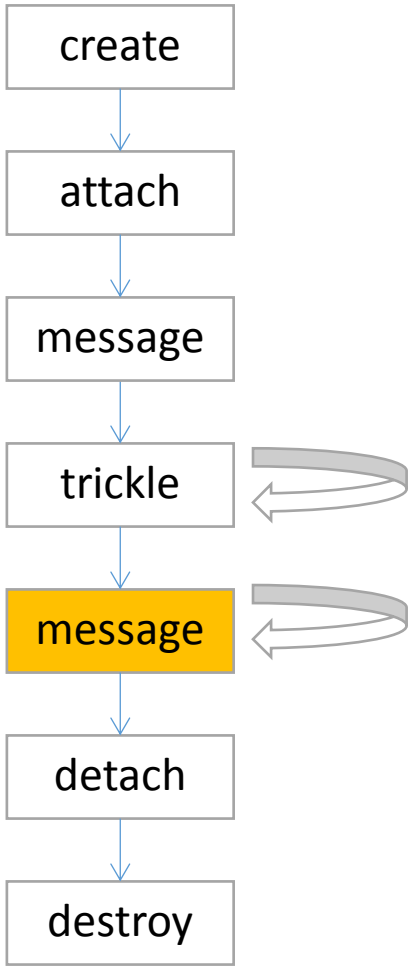
```
{janus: "trickle",…}
  candidate: {candidate: "candidate:2294684747 1 udp 2122260223 192.168.58.1 54882 typ host generation 0",…}
    candidate: "candidate:2294684747 1 udp 2122260223 192.168.58.1 54882 typ host generation 0"
    sdpMLineIndex: 1
    sdpMid: "video"
  janus: "trickle"
  transaction: "OCmoJhSSrZAu"
```

General
Remote Address: 143.225.229.138:443
Request URL: https://janus.conf.meetecho.com/janus/724131890/4226221616
Request Method: POST
Status Code: 200 OK

Response Headers (6)
Request Headers (12)
Request Payload view source

```
{janus: "trickle", candidate: {completed: true}, transaction: "WfTe9H2v6jEc"}
  candidate: {completed: true}
    completed: true
  janus: "trickle"
  transaction: "WfTe9H2v6jEc"
```

JANUS REST API Sequence 4/5



General
Remote Address: 143.225.229.138:443
Request URL: https://janus.conf.meetecho.com/janus/724131890/4226221616
Request Method: POST
Status Code: 200 OK

Response Headers (6)
Request Headers (12)
Request Payload view source

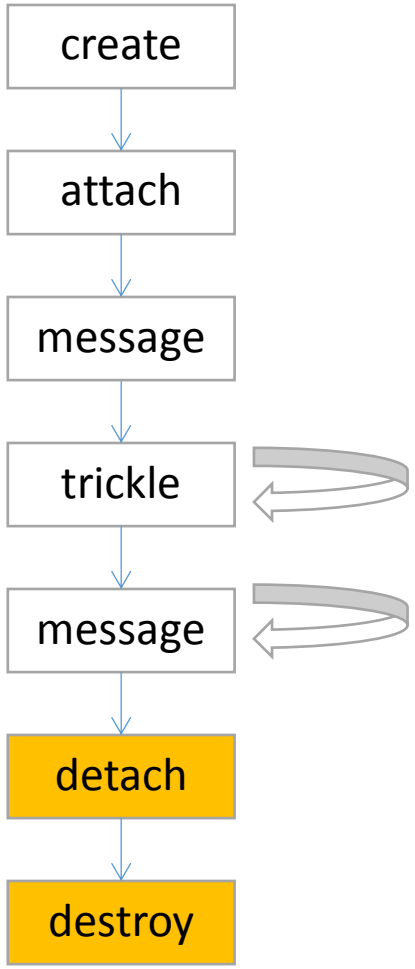
```
{janus: "message", body: {audio: true, video: true}, transaction: "HCEEtrH6UaOg",…}
  body: {audio: true, video: true}
    audio: true
    video: true
  janus: "message"
  jsep: {type: "offer",…}
    sdp: "v=0↵o=- 22269021171182658 2 IN IP4 127.0.0.1↵s=-↵t=0 0↵a=group:BUNDLE audio video data↵a=msid-semantic:…
    type: "offer"
  transaction: "HCEEtrH6UaOg"
```

General
Remote Address: 143.225.229.138:443
Request URL: https://janus.conf.meetecho.com/janus/724131890/4226221616
Request Method: POST
Status Code: 200 OK

Response Headers (6)
Request Headers (12)
Request Payload view source

```
{janus: "message", body: {audio: true, video: true}, transaction: "wCV9ip71Zui7"}
  body: {audio: true, video: true}
    audio: true
    video: true
  janus: "message"
  transaction: "wCV9ip71Zui7"
```

JANUS REST API Sequence 5/5



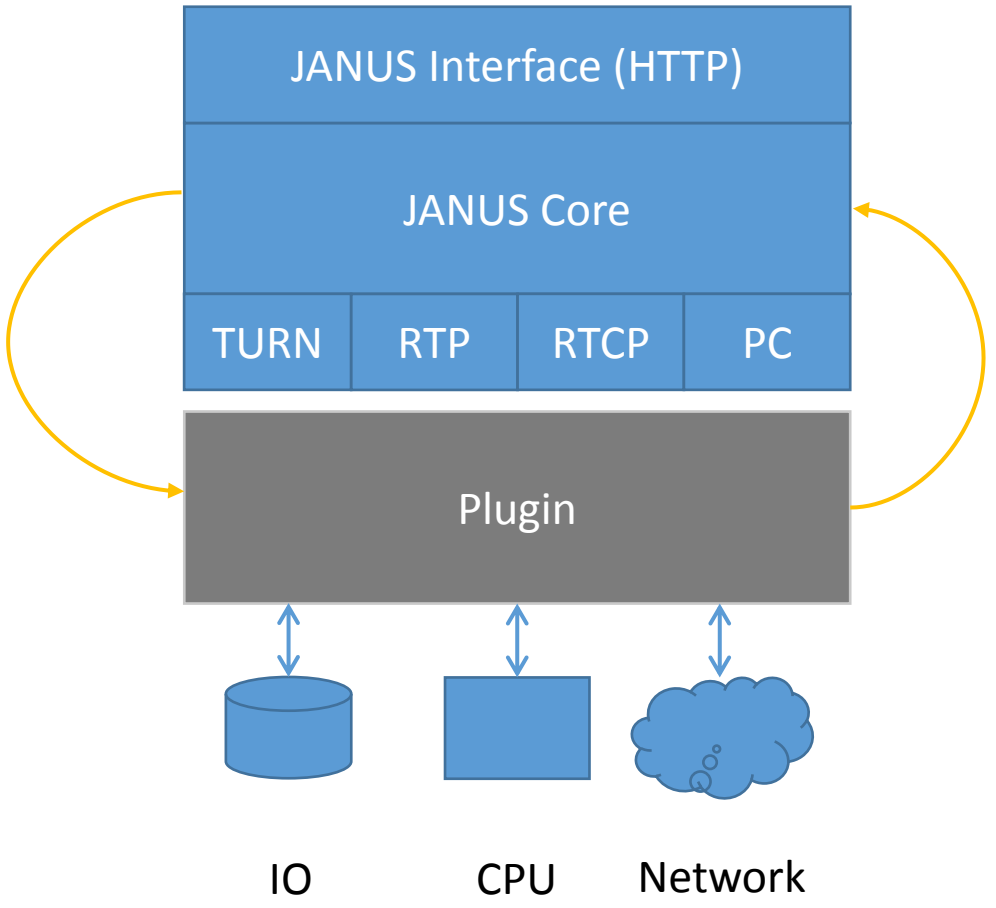
▼General
Request URL: https://janus.conf.meetecho.com/janus/724131890/4226221616
Request Method: POST
Status Code: 200 OK
▶Response Headers (6)
▶Request Headers (12)
▼Request Payload view source
▼{janus: "detach", transaction: "cEX20D8vuOh8"}
janus: "detach"
transaction: "cEX20D8vuOh8"

▼General
Request URL: https://janus.conf.meetecho.com/janus/724131890
Request Method: POST
Status Code: 200 OK
▶Response Headers (6)
▶Request Headers (12)
▼Request Payload view source
▼{janus: "destroy", transaction: "vhyWoUUmuwVF"}
janus: "destroy"
transaction: "vhyWoUUmuwVF"

JANUS Plugin

`struct janus_plugin`

Structure where the plugin must register all the mandatory callbacks.



`struct janus_callbacks`

Structure exposed to plugin with callbacks to the core
(runtime binding?)

JANUS Plugin Interface

```
struct janus_plugin {
 init…
 destroy…
 get_api_compatible…
 get_version…
 get_version_string…
 get_description…
 get_name…
 get_author…
 get_package…
 create_session…
 destroy_session…
 query_session…
 handle_message
 setup_media
 hangup_media…
 incoming_rtp…
 incoming_rtcp…
 incoming_data…
 slow_link…
}
```

```
struct janus_callbacks {
 end_session…
 push_event…
 close_pc…
 relay_rtp…
 relay_rtcp…
 relay_data…
}
```

Plugin Init & Destroy

```
/*! \brief Plugin initialization/constructor
 * @param[in] callback The callback instance the plugin can use to
 *  contact the gateway
 * @param[in] config_path Path of the folder where the configuration for
 *  this plugin can be found
 * @returns 0 in case of success, a negative integer in case of error */
int (* const init)(janus_callbacks *callback, const char *config_path);

/*! \brief Plugin deinitialization/destructor */
void (* const destroy)(void);
```

Plugin Information

```
/*! \brief Informative method to request the API version this plugin was compiled against
 *  \note This was added in version 0.0.7 of the gateway, to address changes to the API that might break existing plugin or
 * the core itself. All plugins MUST implement this method and return JANUS_PLUGIN_API_VERSION to make this work, or they will
 * be rejected by the core. Do NOT try to launch a <= 0.0.7 plugin on a >= 0.0.7 gateway or it will crash. */
int (* const get_api_compatibility)(void);

/*! \brief Informative method to request the numeric version of the plugin */
int (* const get_version)(void);

/*! \brief Informative method to request the string version of the plugin */
const char *(* const get_version_string)(void);

/*! \brief Informative method to request a description of the plugin */
const char *(* const get_description)(void);

/*! \brief Informative method to request the name of the plugin */
const char *(* const get_name)(void);

/*! \brief Informative method to request the author of the plugin */
const char *(* const get_author)(void);

/*! \brief Informative method to request the package name of the plugin (what will be used in web applications to refer to it) */
const char *(* const get_package)(void);
```

Session Handling

```
/*! \brief Method to create a new session/handle for a peer
 * @param[in] handle The plugin/gateway session that will be used for this peer
 * @param[out] error An integer that may contain information about any error */
void (* const create_session)(janus_plugin_session *handle, int *error);

/*! \brief Method to handle an incoming message/request from a peer
 * @param[in] handle The plugin/gateway session used for this peer
 * @param[in] transaction The transaction identifier for this message/request
 * @param[in] message The stringified version of the message/request JSON
 * @returns A janus_plugin_result instance that may contain a response (for
 *  immediate/synchronous replies), an ack
 * (for asynchronously managed requests) or an error */
void (* const destroy_session)(janus_plugin_session *handle, int *error);

/*! \brief Method to get plugin-specific info of a session/handle
 *  \note This was added in version 0.0.7 of the gateway. Janus assumes
 * the string is always allocated, so don't return constants here
 * @param[in] handle The plugin/gateway session used for this peer
 * @returns A JSON-formatted string with the requested info */
char *(* const query_session)(janus_plugin_session *handle);
```

Message Handling

```
/*! \brief Method to handle an incoming message/request from a peer
 * @param[in] handle The plugin/gateway session used for this peer
 * @param[in] transaction The transaction identifier for this
 * message/request
 * @param[in] message The stringified version of the message/request JSON
 * @returns A janus_plugin_result instance that may contain a response
 * (for immediate/synchronous replies), an ack
 * (for asynchronously managed requests) or an error
 */
struct janus_plugin_result * (* const handle_message)
 (janus_plugin_session *handle,
  char *transaction,
  char *message,
  char *sdp_type,
  char *sdp);
```

Media Setup

```
/*! \brief Callback to be notified when the associated PeerConnection is
 * up and ready to be used
 * @param[in] handle The plugin/gateway session used for this peer */
void (* const setup_media)(janus_plugin_session *handle);

/*! \brief Callback to be notified about DTLS alerts from a peer (i.e.,
 * the PeerConnection is not valid any more)
 * @param[in] handle The plugin/gateway session used for this peer */
void (* const hangup_media)(janus_plugin_session *handle);
```

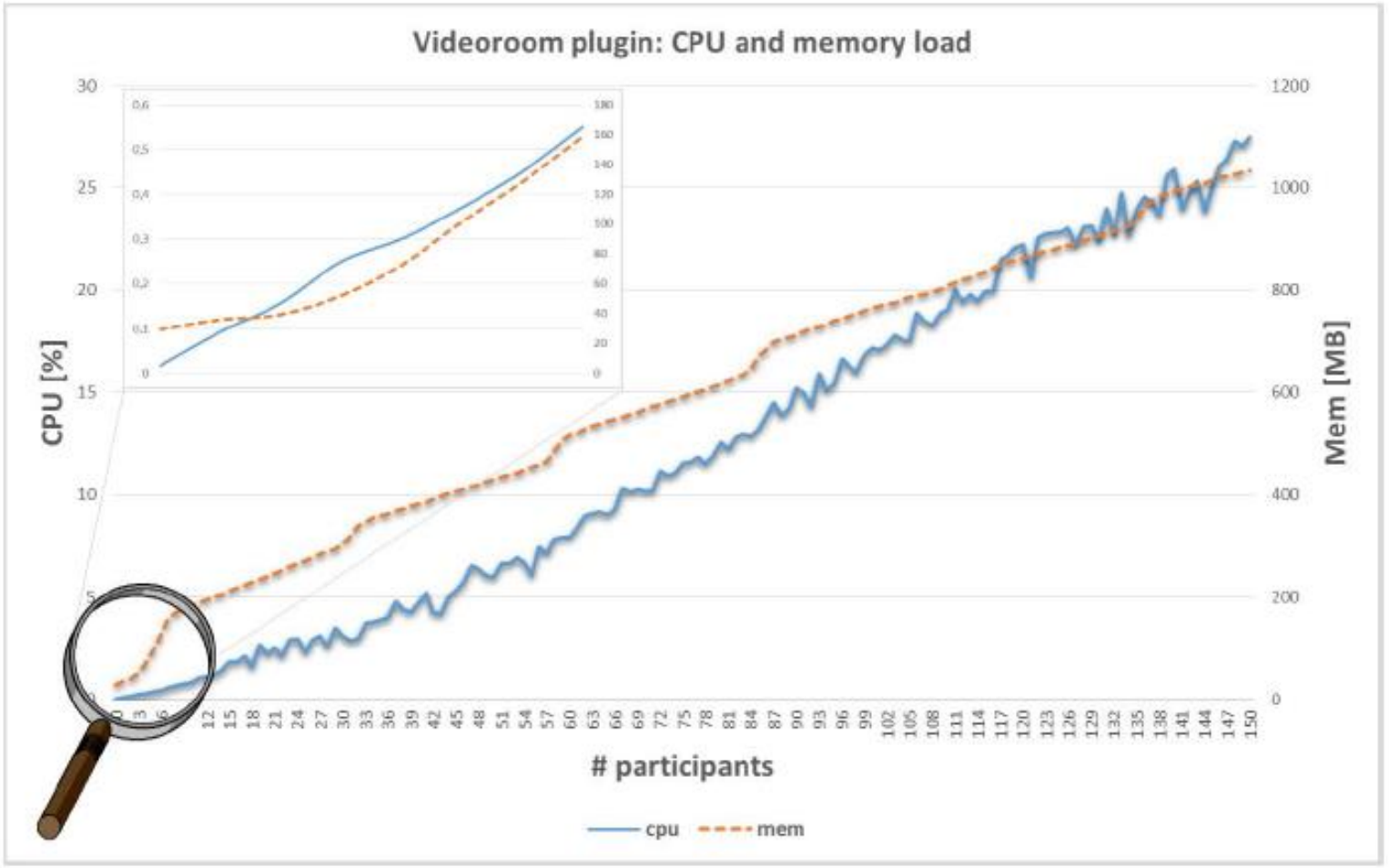
Incoming Data

```
/*! \brief Method to handle an incoming RTP packet from a peer
 * @param[in] handle The plugin/gateway session used for this peer
 * @param[in] video Whether this is an audio or a video frame
 * @param[in] buf The packet data (buffer)
 * @param[in] len The buffer lenght */
void (* const incoming_rtp)(janus_plugin_session *handle, int video, char *buf, int len);

/*! \brief Method to handle an incoming RTCP packet from a peer
 * @param[in] handle The plugin/gateway session used for this peer
 * @param[in] video Whether this is related to an audio or a video stream
 * @param[in] buf The message data (buffer)
 * @param[in] len The buffer lenght */
void (* const incoming_rtcp)(janus_plugin_session *handle, int video, char *buf, int len);

/*! \brief Method to handle incoming SCTP/DataChannel data from a peer (text only, for the moment)
 * \note We currently only support text data, binary data will follow... please also notice that DataChannels send
 * unterminated strings, so you'll have to terminate them with a \0 yourself to use them.
 * @param[in] handle The plugin/gateway session used for this peer
 * @param[in] buf The message data (buffer)
 * @param[in] len The buffer lenght */
void (* const incoming_data)(janus_plugin_session *handle, char *buf, int len);
```

Slow Link Handling

```
/*! \brief Method to be notified by the core when too many NACKs have
 * been received or sent by Janus, and so a slow or potentially
 * unreliable network is to be expected for this peer
 * \note Beware that this callback may be called more than once in a row,
 * (even though never more than once per second), until things go better for that
 * PeerConnection. You may or may not want to handle this callback and
 * act on it, considering you can get bandwidth information from REMB
 * feedback sent by the peer if the browser supports it. Besides, your
 * plugin may not have access to encoder related settings to slow down
 * or decreae the bitrate if required after the callback is called.
 * Nevertheless, it can be useful for debugging, or for informing your
 * users about potential issues that may be happening media-wise.
 * @param[in] handle The plugin/gateway session used for this peer
 * @param[in] uplink Whether this is related to the uplink (Janus to peer)
 * or downlink (peer to Janus)
 * @param[in] video Whether this is related to an audio or a video stream */
void (* const slow_link)(janus_plugin_session *handle, int uplink, int video);
```

Performance



10 Publishers
- Opus Audio
- VP8 Video

140 Subscribers

SFU Mode

1500 Peer Connections

Paper: Performance analysis of the Janus WebRTC gateway

http://dl.acm.org/citation.cfm?doid=2749215.2749223

Conclusion

Pros

- Multi-purpose WebRTC Gateway
- Small Footprint
- Plugin Philosophy
- Scalable
- Full source available
- “easy to use”

Cons

- Almost no documentation
- Small groups of developers
- No wide use (yet?)
- No media libraries to handle/process media streams
- No message router built-in

Janus Gateway Homepage & Demos
https://janus.conf.meetecho.com/

Source Code
https://github.com/meetecho/janus-gateway

rui.ribeiro@fccn.pt

Thank you and any questions

Networks · Services · People
www.geant.org