« Rendez-vous » Web(RTC) Conferencing as a Service

Franck Rupin
RENATER
23-25 rue Daviel
75013 Paris

Abstract

Over the last decade, videoconferencing services were often deployed in the form of Multipoint Control Units (MCUs) using the SIP or H.323 protocols. These deployments have reached a moderate level of popularity and have satisfied some common use cases within the Research and Education (R&E) community. Most of them however have been sharing the same constraints:

- *High cost,*
- *Problematic usability,*
- *Deployment complexity,*
- *Lack of diversity.*

The Jitsi Videobridge Selective Forwarding Unity and the accompanying Jitsi Meet web application appealed to RENATER as a unique opportunity to achieve one specific objective: Bringing affordable video conferencing services to the masses!

Key words

Webrtc, Jitsi, cloud, xmpp.

1 Introduction

The “Rendez-vous” project started in an effort to build a scalable web conferencing service for the French research and education community. This goal involved many issues such as cost optimizations, cloud versus hardware choices, plug-in management, etc.

In 2013 while both BlueJimp and RENATER took part in the biennial French conference ”JRES”, we could share our common views on the conferencing problems and solutions.

When trying to achieve scalability in video conferencing there are multiple factors to take into account. First, if one wants to base a desktop solution on a legacy infrastructure one should consider that such services are complex for the service provider but also for the user and hiding this complexity is not easy. As soon as a solution emerges one should also worry about interoperability and how it can contribute to higher costs due to hardware or licensing.

From a user perspective maintaining a H323/SIP infrastructure often requires a dedicated team in addition to the increased load on the network and system teams. Trying to resolve that various NAT and firewall traversal issues may also

sometimes feel as a security concern, requiring interventions from yet another team. This makes such solutions difficult to scale or adapt to the needs of a research or an education institution.

Second, many manufacturers provide Web conferencing services in the cloud. They often bring with them fee considerations and the need to manage plug-in installation. There are also other questions regarding confidentiality. Also, since the Snowden and WikiLeaks scandals, many of us would rather avoid external hosting. This has turned out to be particularly important to the members of our French community.

Obviously there are cases where both cloud and on-premises SIP/H.323 installations can actually be satisfactory and there are many using them. Still, neither of them is truly suitable for a scalable cost-effective deployment, which makes it very important to look for alternative technologies.

Until recently, a huge part of our users were excluded from our MCU-based conferencing services. Students for example had no access to them, so they switched to other tools such as Microsoft Skype and Google Hangouts. So when at RENATER we met Jitsi we were quite excited because we immediately recognized it as a viable alternative that perfectly matched our needs.

The first thing we did was to create a demo deployment and involve our community in testing it. We then asked everyone if the sort of service could solve their day-to-day conferencing needs. The feedback we received was very enthusiastic and many saw the application as a Skype challenger (even though we had never tried to present it as such).

Yet, Jitsi Videobridge and Jitsi Meet did not seem to have everything that we needed in order to put them in production. They were a promising WebRTC application but from an operator's perspective we still needed quite a few things that were essential to our future "rendez-vous" service:

- Stability
- Access control
- Logging
- High availability
- Load balancing

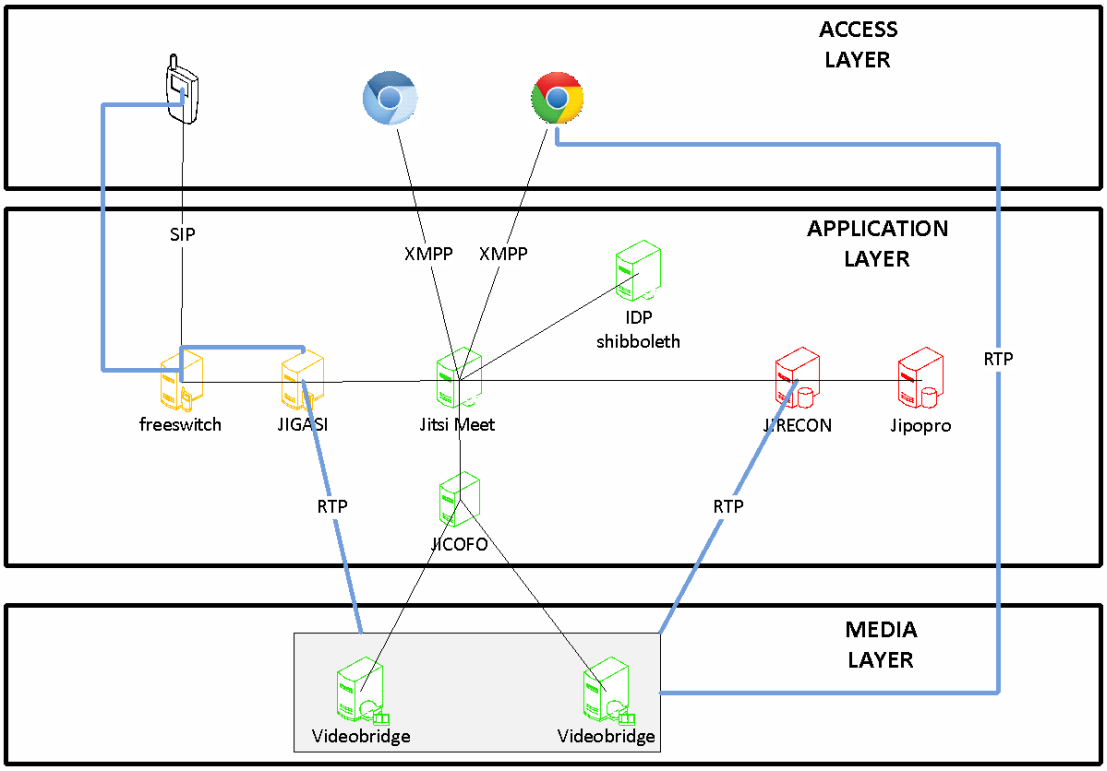
The rest of the article describes how we resolved the above constraints both within Jitsi Videobridge and with the architecture of RENATER's new infrastructure.

2 Jitsi Videobridge design

2.1 Overview

The design of the Rendez-Vous service can be viewed as a WebRTC stack in which each layer involves different stakeholders. When setting up the only thing users need to do is to open their browser and type a URL. We didn't want it to be any more complicated than that.

The figure below shows our service stack as it was deployed at RENATER :



Application Stack

We enter into the WebRTC world with the access layer. This is the area involving browsers or SIP audio devices (for interoperability). The browser downloads and executes the Jitsi Meet application. This is a Javascript application embedding WebRTC capabilities. The Jitsi Meet application calls the getUserMedia (gUM) and the peer connection API. It also establishes signaling and managed media sessions with XMPP Jingle.

The application layer is the area where signaling is operated and different processes are involved to provide services such as load balancing, authentication and so on.

When the whole process ends to relate the browser with the Jisti Videobridge (Media Layer), media streams are directly sent between browser and Videobridge without relay while signaling is still held by Jicofo and Prosody services.

2.2 Jitsi Meet

This component provides the javascript application and establishes the BOSH transport layer for XMPP signaling.

Then once application is executed and signaling is handled, User always communicate via Jicofo because it is responsible for signaling and relays to the videobridge. Jitsi Meet is useful application with many features such as screen sharing, etherpad for editing documents, multi user chatroom… This application is simple to use and doesn't require technical skills to be used.

2.3 Jicofo: JItsi COnference FOcus

A Conference Focus is a signaling entity that orchestrates a session. Formerly this role was held by the first participant who joined the room. The process was embedded into the browser, which caused some amount of unreliability. Indeed, it was frequent that browser holding the focus crashed and all participants into the room were kicked off.

As the client-side focus was identified as a source of instability, the role was implemented server side as the Jicofo application. Jicofo currently plays the following roles:

1. Managing Colibri channels for participants and establishing media flow to/from the JVB.

2. Bridge load balancing based on conference count.

3. Can handle client authentication (optionally)

4. Takes part in some features implementation like audio muting, recording, SIP calls. It is processing and forwarding client requests to corresponding components. Verifies user permissions: only moderators are allowed to start new SIP call or mute others.

Jicofo is the central signaling component in the system.

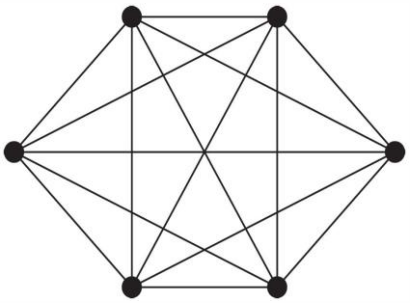
2.4 Jitsi Videobridge

Jitsi Videobridge acts as media server in the Jitsi infrastructure. This is the core of the system. It implements many features such as simulcast, DTLS/SRTP, Last N, ICE and others.

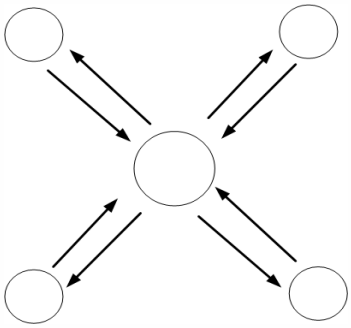
Before we dive deeper into details let us make sure we are all on the same page with regard to the basics. Different kinds of models are available to stream media, but some of them are more reliable than others.

Full mesh

In this model each point is connected to all other points. The main problem with this is that it does not scale. In the figure below each segment is bidirectional. So, possible combinations increase significantly when adding a node.

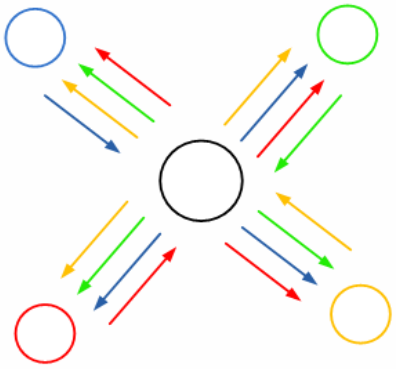


full-mesh



Star (MCU)

Traditional MCUs use a star model. End-points send their streams to the central point and it sends to each of them a stream containing a mix of everyone else's content.



Full-star (SFU)

Selective Forwarding Units (SFUs) are a third way of approaching media distribution. Similarly to full-mesh architectures, all participants receive independently all streams from everyone else. Contrary to full-mesh apps though, participants in SFU hosted conferences only need to send their outgoing streams once. It is the responsibility of the SFU to then deliver it to everyone else (or at least the participants that are considered appropriate receivers).

Also, contrary to an MCU, an SFU does not operate with media. It may inspect and analyse packets but it would never need to implement any encoding, decoding or compositing features.

This helps SFUs perform orders of magnitude better than MCUs in terms of CPU consumption. It also eliminates the need for any codec licenses for the SFU.

2.5 LastN

Use of resources in typical full-star topologies increases quadratically with the number of participants. This also implies a usability problem: while it makes sense for everyone to see everyone else in a conference with several participants, a call with a few hundred attendees is obviously a different story.

In order to solve this Jitsi Videobridge adds a Last N feature that only redistributes a limited number of recent active speakers to everyone in the call.

Additional details on the operation and implementation of Last N can be found in [2]

2.6 Simulcast

An important characteristic of many MCUs is their ability to modify the bandwidth of the streams they are sending to participating endpoints. This allows them to support diverse network conditions and devices.

Given that encoding and decoding are never performed on an SFU, it is important that bandwidth adaptivity be achieved in a different way. SFUs achieve that by subcontracting it to endpoints: rather than sending a single high resolution, all endpoints encode their video streams multiple times. Depending on their available upstream bandwidth, they can then send all or some of the encoded layers to the SFU. The SFU itself can make the same choice when forwarding media to the rest of the connected devices.

This feature is known as Simulcast and Jitsi Videobridge supports it in its latest versions.

2.7 ICE

Traversing NATs has been one of the most widely prevalent problems for audio/video real-time communication. The Interactive Connectivity Establishment protocol that the IETF standardized in 2010 as RFC 5245. Jitsi Videobridge uses ICE in a particularly advantageous way: given that Jitsi Videobridge deployments are always guaranteed to have a public address they do not require use of neither TURN relays or not even STUN servers. All candidate IP addresses are discovered by the videobridge during session negotiation.

In its latest versions Jitsi Videobridge even supports direct TCP connections (ICE-TCP) which makes it usable even from within corporate networks that only authorize outgoing connections on port 443.

3 Operator point of view

3.1 Technical context

The technical context for running Jitsi Videobridge is a virtualization environment. Since Jitsi Videobridge is a SFU, it is easy to virtualize the media server without performance issues. This is important because operating in the cloud is the key for cost optimization.

As virtualization is in the DNA of Jitsi Videobridge, scalability is easily allowed for by the videobridge.

3.2 High Availability

While Jitsi Videobridge handles scalability at the transport layer, it is important to also consider the operational side of things. Providing such a service in production makes availability a crucial matter. The first pilot version did not support a high availability feature, it means if a videobridge failed, service failed as well. To resolve that Jicofo had to support multiple videobridge instances simultaneously.

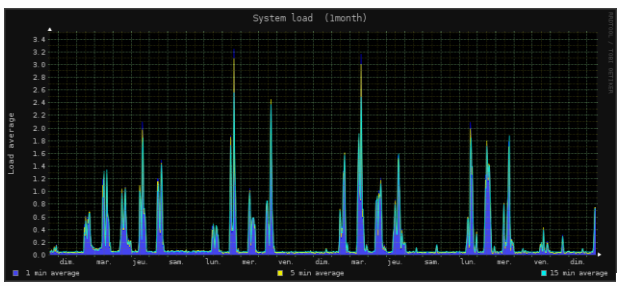
Blue Jimp and RENATER worked together on implementing this feature. Improving the service started by videobridge and should continue by improving the Jicofo service and then the Jitsi Meet front-end.

3.3 Load balancing

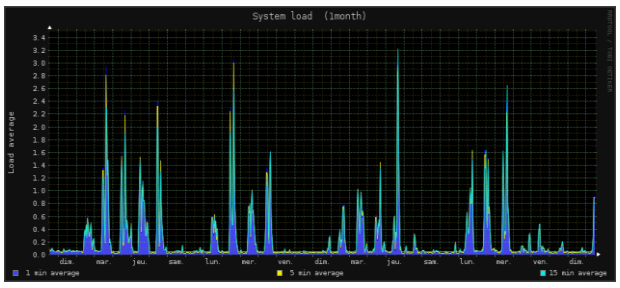
This feature is associated with High Availability

Once again this is the Jicofo service which is in charge of this feature. Jicofo implements a load balancing algorithm, based on the number of conferences, participants and cpu workload.

The two following figures below show the workload sharing for two videobridges running simultaneously on the “Rendez-Vous” service. We can see the similar shapes between the two figures which proved the load balancing works well and reaches the goal.



Load



Load balancing : Videobridge 2 workload

3.4 Monitoring

- Jitsi Meet-Torture

Traditional monitoring systems can be used to monitor infrastructure components (e.g. nagios), but in order to monitor the service itself one should add more features. The Jitsi Meet-Torture application helps detect videobridge failures. Note that monitoring the Videobridge alone is not sufficient and one also needs to continually check its ability to create media rooms and to actually stream content.

Jitsi Meet-Torture is based on selenium chrome driver to test the room creation process.

- Real time monitoring with Grafana dashboard

An easy way monitor activity in real-time, consists in deploying a Grafana dashboard and query the Colibri REST API provided by each Jitsi videobridge. It requires developing some short pieces of code to run, but it is reliable and easy (welcome to the devops world). The Figure below shows participants and conferences on a Videobridge. But the colibri REST API provide also cpu workload, packet loss and other interesting numbers.

Real time monitoring: Grafana dashboard

3.5 Statistics and Logging

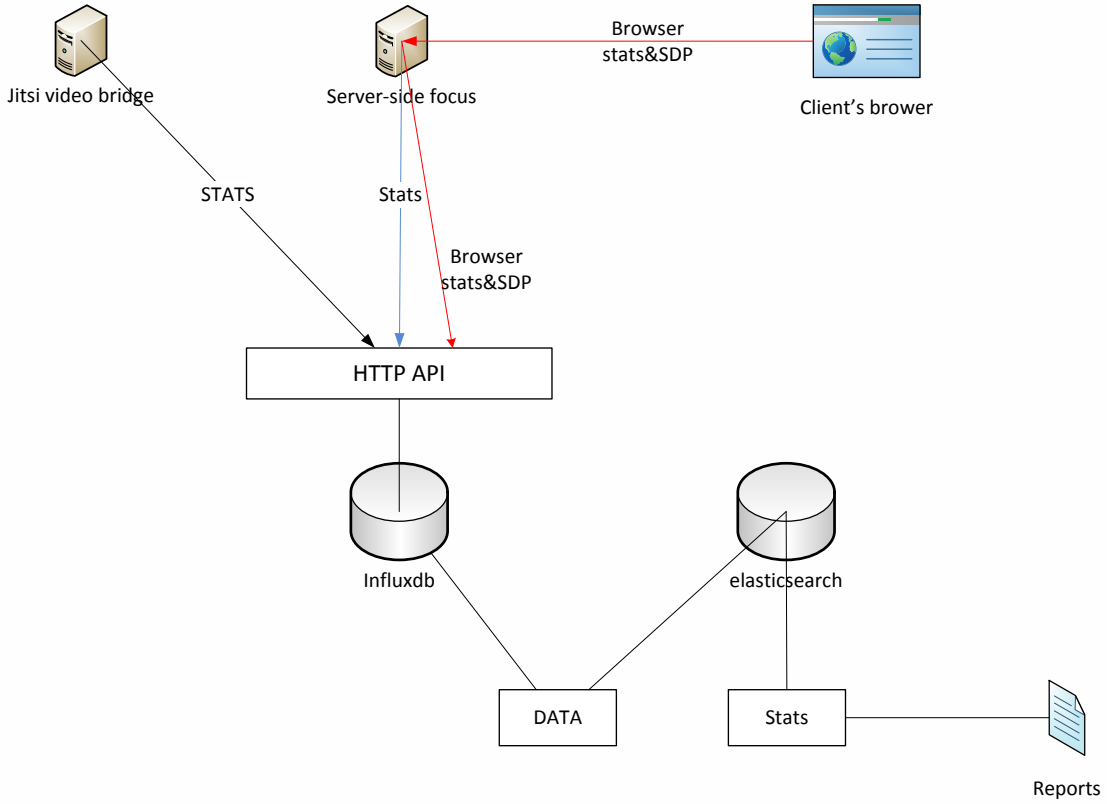
The figure below shows the different sources which collect logs and statistics. There 3 main sources:

- Jitsi-Videobridge
- Jicofo (Server side focus)
- Browser through Jicofo

The whole data are pushed into an influxDB backend in real time. So a conference is splitted into multiple schemes into influxDB related by common informations. But retrace a conference is a big challenge regarding the amount of data we need to parse. We developed a program to refactor all conferences and insert them into a elasticsearch backend as coherent documents. Thereby we are faster to compute metrics based on these refactored data. The figure below shows the workflow in which the “DATA” element is the program in charge for refactoring data and store document into

elasticsearch, and “stats” is the program which perform elasticsearch request based on the full text search engine and produce a report.

Reports could be enhanced by adding charts for example and also by programming much requests to provide new vision over the service: IDP top user, Mean Opinion Score of conference and so on.



Statistics workflow

Here an example of a report generated to provide the usages of the service:

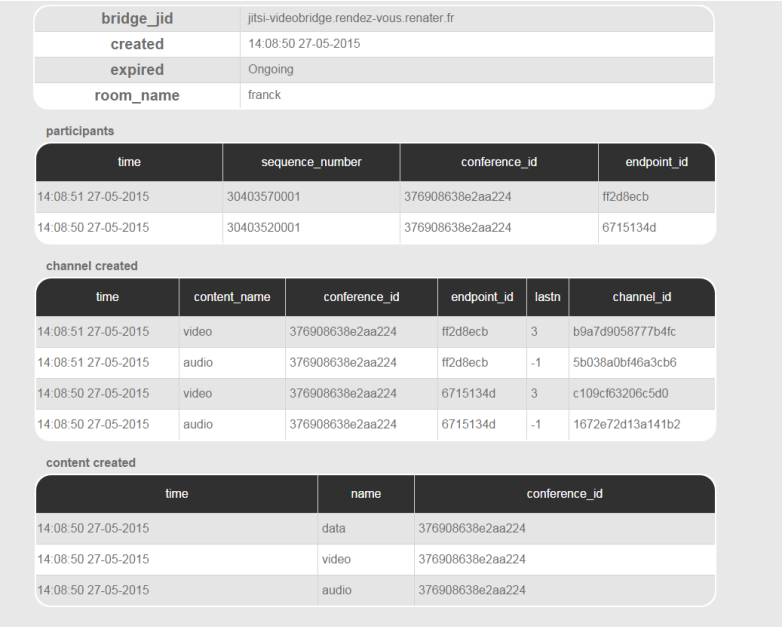
	2015-03 (16 to 31)	2015-04	2015-05	2015-06	2015-07	2015-08	2015-09	SUMS
Total Conf	1280	1955	1714	2520	1413	611	2071	11564
Total Cnx	5113	8190	7109	10277	5458	2290	9098	47535
Total duration in days	37,2	76,8	74,5	110,3	63,4	27,1	95,4	484,7
Average Cnx	3,9	4,1	4,1	4	3,8	3,7	4,3	3,98571429
Average Duration in minutes	41,8	56,5	62,6	63	64,7	63,8	66,3	59,8142857

Report example

- Logging which is required for troubleshooting:

Browser WebRTC internals are collected by Jicofo.

So Jiloin is a simple interface which allows investigating when users report issues. This already help us to solve issues, it was really appreciate while upgrading the platform with major updates. InfluxDB also provides its own interface to query data.



bridge_jid	jitsi-videobridge.rendez-vous.renater.fr
created	14:08:50 27-05-2015
expired	Ongoing
room_name	franck

participants

time	sequence_number	conference_id	endpoint_id
14:08:51 27-05-2015	30403570001	376908638e2aa224	ff2d8ecb
14:08:50 27-05-2015	30403520001	376908638e2aa224	6715134d

channel created

time	content_name	conference_id	endpoint_id	lastn	channel_id
14:08:51 27-05-2015	video	376908638e2aa224	ff2d8ecb	3	b9a7d9058777b4fc
14:08:51 27-05-2015	audio	376908638e2aa224	ff2d8ecb	-1	5b038a0bf46a3cb6
14:08:50 27-05-2015	video	376908638e2aa224	6715134d	3	c109cf63206c5d0
14:08:50 27-05-2015	audio	376908638e2aa224	6715134d	-1	1672e72d13a141b2

content created

time	name	conference_id
14:08:50 27-05-2015	data	376908638e2aa224
14:08:50 27-05-2015	video	376908638e2aa224
14:08:50 27-05-2015	audio	376908638e2aa224

Logging JID general view

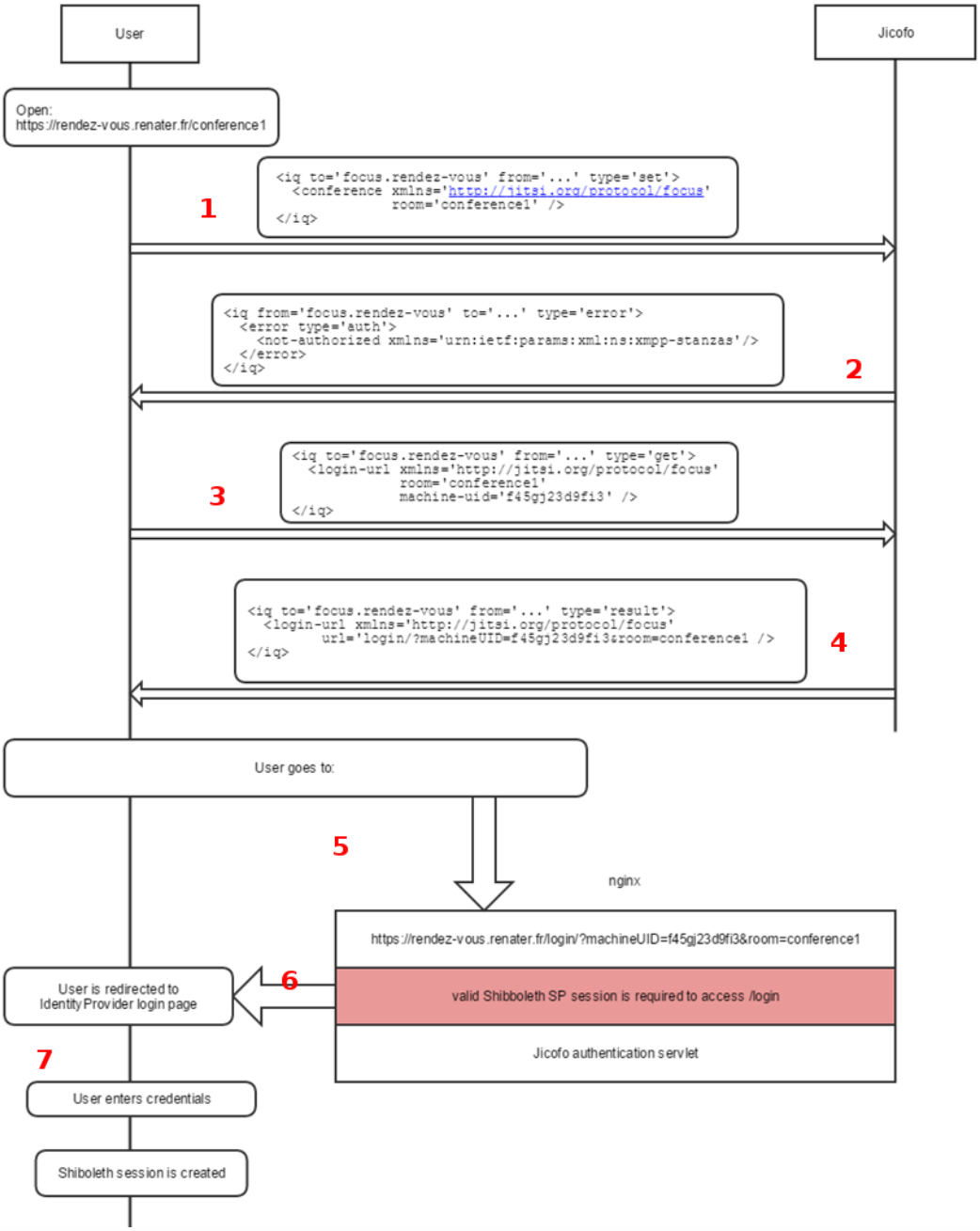
Authentication headers are also stored, it makes possible to identified users by their Identity Provider which allows the operator to provide usages reporting by site. This is a really useful feature.

3.6 Authentication

When RENATER ran its first pilot platform there was no access control mechanism available. This point was identified as necessary to provide the service, SAMLv2 support was required to be able to authenticate via Shibboleth and eduGAIN. Then it was implemented as shown by the commented flows below.

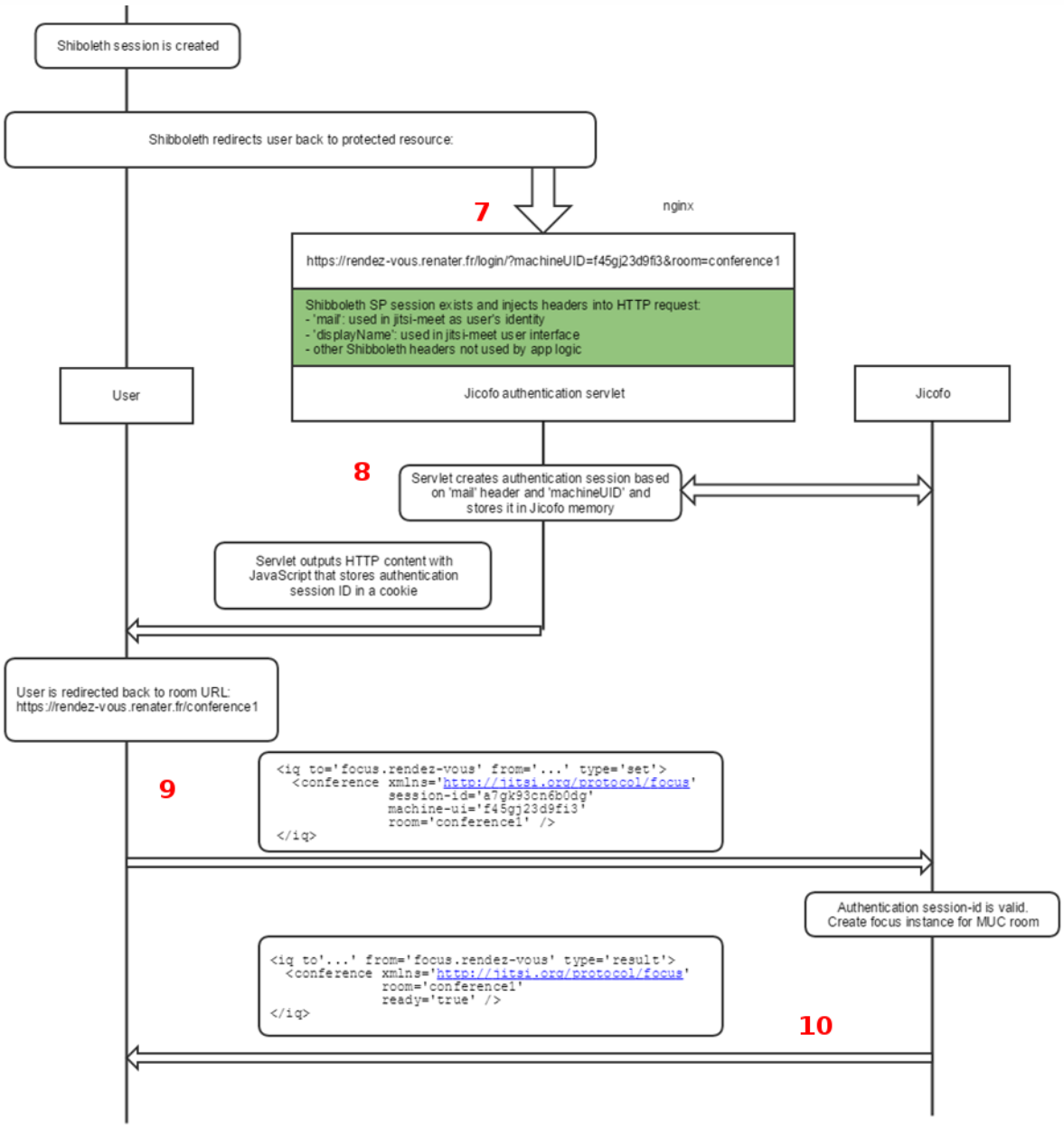
First of all Jicofo is the process in charge of authentication. So it requires special settings into its configuration file to trigger the shibboleth process. Nginx is the web server and it is also configured to trigger a proxy cgi to Shibboleth process.

- Step 1: User request a URL which doesn't exist yet.
- Step 2: Jicofo handles the request but detects the room has to be created, so it sends to the users there is a lack of authorization.
- Step 3: User sends to Jicofo additional attributes such as uuid machine
- Step 4: Jicofo composes an URL to the user for handling the authentication process.
- Step 5: User goes to the URL hosted by Nginx
- Step 6: the request doesn't contains a valid Shibboleth SP Session, it triggers the redirection of the user to its Identity Provider.
- Step 7: User authenticates himself and a shibboleth session is created. The user is redirected to origin web page which triggered the authentication process.



Next steps are illustrated on the next figure.

- Step 8: Shibboleth session is handled by the Jicofo servlet in charge of authentication. It creates a session Id.
- Step 9: The user is allowed to return back to its first URL requested.and he sends the session Id into his request.
- Step 10: Jicofo can verify that the session is valid for the URL so it can handle the room process creation and sends a stanza to user which confirms the room is created.



4 Conclusion

“Rendez-vous” is a project based on innovative technologies such as WebRTC or SFU. Challenging Jitsi videobridge for scalability seems to be a realistic scenario even if there is still some way to go before to provide an elastic service. Technology also allows to cover new use cases address both to end-users and operators.

For end-users, it could be easy to develop their own application embedding media capabilities which allows to be close to user’s needs. For the operator the challenge will be to provide media resources through an API to allow rapid development.

One of the most satisfying point with this project is that the enthusiasm from GEANT and RENATER seems to be the same which make the future very exciting.

5 References

[1] http://www.w3.org/TR/2015/WD-mediacapture-streams-20150414/#constructors

[2] https://jitsi.org/publications/nossdav2015lastn.pdf

[3] https://tools.ietf.org/html/draft-uberti-rtcweb-plan-00

[4] https://tools.ietf.org/html/draft-alvestrand-rmcat-congestion-02.

[5] https://sites.google.com/a/chromium.org/chromedriver/

[6] https://github.com/jitsi/jitsi-videobridge/blob/master/doc/rest-videobridge.md

[7] https://rendez-vous.renater.fr

[8] Franck Rupin 2015 : Rendez-vous-gn4-SA8 documentation.

6 Acknowledgements

to Emil Ivov the Jitsi founder and his team.

Frederic Loui (RENATER)

7 Glossary

JITSI : JITSI stands for " Wire" in Bulgarian. It is an audio/video Internet phone and instant messenger software. It is part software suite developed Emil Ivov initially a student at Strasbourg University in France. The suite is composed by various software components playing specific role : Jitsi Meet, LibJitsi, Ice4J.org, TurnServer, Jitsi Video Bridge.

JITSI-MEET : is an OpenSource (APL) WebRTC JavaScript application that uses Jitsi Videobridge to provide high quality, scalable video conferences.

JICOFO : JItsi COnference FOcus is a server side focus component used in Jitsi Meet conferences. Conference focus is mandatory component of Jitsi Meet conferencing system next to the videobridge. It is responsible for managing media sessions between each of the participants and the videobridge. Whenever new conference is about to start an IQ is sent to the component to allocate new focus instance. After that special focus participant joins Multi User Chat room. It will be creating Jingle session between Jitsi videobridge and the participant. Although the session in terms of XMPP is between focus user and participant the media will flow between participant and the videobridge. That's because focus user will allocate Colibri channels on the bridge and use them as it's own Jingle transport.

JITSI VIDEOBRIDGE : Jitsi Videobridge is a WebRTC compatible Selective Forwarding Unit (SFU) that allows for multiuser video communication. Unlike expensive dedicated hardware videobridges, Jitsi Videobridge does not mix the video channels into a composite video stream. It only relays the received video flows to all call participants. This makes Jitsi extremely scalable and, while it does need to run on a server with good network bandwidth, CPU horsepower is not critical for performance.

PROSODY : Formerly lxmppd is a cross-platform XMPP server written in Lua. Its development goals include low resource usage, ease of use, and extensibility. This is the XMPP server used by JITSI MEET.

COLIBRI : COLIBRI protocol (COnferences with LIghtweight BRIdging). COLIBRI is an open XMPP extension protocol designed by the Jitsi development team. It allows the conference organizer to allocate channels for everyone add or remove participants, and generally remain informed of the call state. All this happens transparently for the user. You can find information on how the COLIBRI protocol works in the COLIBRI XEP. In JITSI environment, a COLIBRI session is set up between the JICOFO and the JITSI Video Bridge.

JINGLE : Jingle is an XMPP extension XEP-0166 that enables two XMPP entities to set up, manage, and tear down a multimedia session. The negotiation takes place over XMPP, and the media transfer typically takes place outside of

XMPP. In JITSI environment, a Jingle session is set up between the participant and the the server side focus components: JICOFO

ICE : Protocol for NAT traversal for UDP based multimedia sessions established with the offer/answer model. Conceptually, it is a form of Interactive Connectivity Establishment.

Trickle ICE : Begin connectivity checks while still gathering candidates that would be used by the session between the peers. Improvement, extensions of ICE. Making candidate harvesting a non-blocking process

WebRTC : Web Real Time Communication