30-04-2016

SA8T2 Internal Deliverable
Distributed RENdez-Vous service with Jitsi software suite

SA8T2 Internal Deliverable

Contractual Date:	30-04-2016
Actual Date:	30-04-2016
Grant Agreement No.:	691567
Activity:	12/SA8
Task Item:	Task 2 — WebRTC
Nature of Deliverable:	R (Report)
Dissemination Level:	PU (Public)
Lead Partner:	NORDUnet (UNINETT)
Authors:	Frédéric Loui (RENATER)

The research leading to these results has received funding from the European Union's Horizon 2020 research and innovation programme under Grant Agreement No. 691567 (GN4-1).

Abstract
This document reports on results and findings from a technical investigation into potential deployment configurations of a federated RENdez-Vous service distributed across several NREN domains and scalable at the European level. The technology scout was conducted by the Service Activity 8 (SA8, Real Time Communication and Media), Task 2 (WebRTC) team as part of the GN4-1 project. This report should, as such, be read in context of the related work produced by GN4-1 SA8-T2.

Table of Contents

Table of Figures

Executive Summary

RENdez-Vous is a multi-party web conference service deployed by the French NREN, RENATER. The service is based on Jitsi; an open source software suite powered by WebRTC. This technology scouting report investigates potential deployment configurations of a federated RENdez-Vous service distributed across several NREN domains and scalable at the European level. The Jitsi suite is made up of three main components:

Jitsi Meet — the front end user interface, e.g. where users meet via a common URL.

Jitsi Videobridge — acts as a central component that relays participants' audio, video and data streams to the other peers.

Jicofo — a component between Jitsi Meet and Jitsi Videobridge responsible for room creation requests and to tie users to a room.

Further investigations identify that multiple video bridges can be deployed for a Jitsi instance. A key role of Jicofo is to balance the load of the whole system by distributing rooms among available video bridges. This is how unlimited scalability can be achieved in theory.

Jitsi Meet can delegate authentication to SAMLv2 IDP and thus integrate with eduGAIN so that only people belonging to this federation can spawn a virtual conference room. Jitsi Meet also provides an API that, although not yet fully mature, can be used to integrate (embed) and control meeting rooms inside other applications (in-context).

Although scalability is ensured by deploying more than one video bridge, Jitsi Meet and Jicofo can be identified as single point of failure components. Jitsi Meet runs behind a web server and may thus achieve high availability. This does not apply to Jicofo, where additional development efforts are required. Blue Jimp, the company behind Jitsi, has not yet committed to development in this area.

A RENdez-Vous instance constructed with one Jitsi Meet, one Jicofo and a set of Jitsi Videobridges is self contained. It cannot interact with Jitsi Meet or Jicofo components from a different instance.

A number of architectures were proposed for a European R&E global context. However, after further investigation and discussions with Blue Jimp, the current 3-tier structure inherently restricts the number of possible architectures. One alternative is to deploy a global instance across Europe; i.e. one Jitsi Meet, one Jicofo and distribute video bridges across NRENs' IaaS infrastructures. The second alternative involves building a self-contained instance within each national domains infrastructure, either managed by GEANT or the NREN.

A potential scenario combines 3 deployment considerations:

- One single GEANT RENdez-Vous be deployed for e-infrastructure that needs multi-party video conference turn key solution from the GEANT catalogue;

- GEANT creates and manages a RENdez-Vous instance for each NREN wanting to benefit from the service, but can not handle the operations themselves and do not have the adequate infrastructure to host the service;

- For each NREN willing to host and operate their own service instance; re-use the service delivery model elaborated in the frame of GN4-1, GN4-2. A variant of this solution would be to have GEANT create an instance of RENdez-Vous for each NREN within each NREN's infrastructure.

1 Introduction

1.1 About this document

This report documents the investigation into the technical and practical feasibility of various deployment configurations to realise a scalable multi-party web conference service based on Jitsi software.

The technology scout was undertaken as part of the Geant4 Phase 1 project by the WebRTC Task 2(T2); one of three tasks of the Real Time Communication and Media activity (SA8). This report should, as such, be read in context of the related work produced by GN4-1 SA8-T2. The WebRTC task ran from 1 May 2015 to 30 April 2016.

1.1.1 Target audience

This document targets technical management and specialists, in particular those working in the fields of real time communications, eLearning and eResearch.

1.1.2 Responsible task members

Fréderic Loui (RENATER) had the lead on this tech scout. Jan Meijer (UNINETT) and Simon Skrødal (UNINETT) were the document editors.

1.2 Background

The French NREN, *RENATER,* was the first NREN to provide a native WebRTC-based online desktop audio/videoconferencing service to its community. The RENdez-Vous service (https://rendez-vous.renater.fr) was set up as a pilot in the fall of 2014 and announced as a production service in March 2015. eduGAIN authentication (http://services.geant.net/edugain/Pages/Home.aspx) was added as part of RENATER's national WebRTC effort during 2015 to allow harvesting experiences from a wider user community. The service builds on the open source software product Jitsi, and is currently the only known example of a Selective Forwarding Unit system.

Based on the technologies underpinning RENdez-Vous, this technology scout sought to investigate their potential in realising a larger-scale distributed service to facilitate more users in the wider GÉANT community.

1.3 Rationale

The RENdez-Vous service utilises the Jitsi open source software, using a Selective Forwarding Unit (SFU) as a central server component rather than routing media traffic peer-to-peer. Experiences with R&E web conferencing services show online meetings of eLearners and researchers often reach a group size of 10–15 people, which is too challenging for the peer-to-peer technology used by most commercial native WebRTC solutions.

It made sense to investigate the technical potential for a scalable GÉANT desktop videoconference service based on Jitsi: a distributed RENdez-Vous.

1.4 Technology scout objectives and methodology

The key objectives of this technology scout were to assess whether the Jitsi software could be used to scale out to a distributed deployment supporting all of EU R&E. This would create a single distributed WebRTC desktop video conference service for all NRENs with minimal effort.

Based on an assessment of various Jitsi technology options, an architecture for a distributed Jitsi deployment was defined and implemented as a PoC. The concept was (manually) validated with 2 nodes in 2 countries.

Unfortunately, the project staff member undertaking the work left for a new job and there was not enough time available to replace his unique expertise in the project. As such, the automation of the process could not be validated, or tested on an even larger scale.

2 Technology introduction

Innovation and, more widely, open innovation are enabled by research projects involving people from different countries. These projects become de facto “global” in the geographical sense. An array of telescopes can be located in Chile or Australia, while the datacenters in charge of computing an unmatched 3D resolution representation of the galaxy is located in the NCSA (Illinois University/US) or CCIN2P3 (Lyon/France) and the astrophysicists are based in Paris or Stanford University. On the same token, open source projects such as OpenStreetMap is an iconic example of an open innovation success enabled by developers and crowds working all around the globe.

2.1 History

In this context, audio/video conferencing is key for researchers participating in global projects. In the 90’s, these services relied upon traditional H323 protocol standards implemented by hardware vendors operating according to their own interpretations of the standard. Therefore, interworking between vendor solutions were not guaranteed.

In addition, the technology context was significantly different; the WAN was powered by network links based on X25, frame-relay and ATM technology and most of the access link bandwidth configured for the circuit were multiple of 100 Kbps. Consequently, MCUs had the difficult hurdle to multiplex, compress and relay in audio and video streams from the participants in real-time. Usually, MCU implementations relied on specific hardware ASIC tailored for specific functions tied to the MCU tasks. Considerable hardware engineering was therefore demanded to elaborate these solutions. While this may justify the price tag imposed by the major vendors, many organisations could not justify (afford) such investments.

2.2 Enter WebRTC

Since 2011, however, WebRTC-based conference technology has started to emerge. The technology proposes to implement a set of API standards enabling real-time audio/video and data communication through the web browser. While the W3C is standardising this API at the browser level, the IETF is working on all standard protocols enabling signaling and effective media communication real time transfer between “browser peers”.

In parallel, the WAN is now powered by circuits whose bandwidth are multiple of 100 GE (Gigabit Ethernet) and most client devices such as desktop and even tablets or mobile phones tend to have hardware able to handle audio/video operation in various resolution in real time.

The convergence of these observations provides a favorable context for WebRTC adoption. Note also that this technology is backed up by major Internet players like Google, Mozilla, Opera, Microsoft, Sony Ericsson etc.

Being a web technology, WebRTC inherently presents a number of interesting facilities such as being able to operate on commodity hardware and the capability to interoperate with other web standards that pave the way to develop "web media in-context applications". The Technology Readiness Level (TRL) of WebRTC can be debated, but we can establish that it has easily reached TRL7. While WebRTC can be identified as mature, the standard still evolves at a high pace. Open source software relying on WebRTC therefore need continuous development in order to cope with the steady evolution and maturity of the standard.

2.3 About Jitsi

In the tech scout we evaluated the Jitsi software portfolio in order to determine its suitability as the base of a shared WebRTC-based conferencing service for European R&E. Jitsi presents convincing capabilities pertaining to this domain; it is the only identified open source software suite stable enough to provide audio/video and data communication between R&E users.

Jitsi ("wire" in Bulgarian) is an audio/video Internet phone and instant messenger software. It makes part of a software suite developed by Emil Ivov, currently Chief Video Architect at Atlassian, and initially a student at Strasbourg University in France. The suite is composed by a number of components providing specific functionalities, e.g. Jitsi Meet, LibJitsi, Ice4J.org, TurnServer and Jitsi Video Bridge.

Jitsi originated as "SIP Communicator", a software developed in JAVA by the University of Strasbourg Research Labs. It was meant to enable audio/video communication via the SIP protocol. A French startup, Blue Jimp, then emerged from the University of Strasbourg and joined the WebRTC bandwagon at an early stage alongside Google and continued work on SIP Communicator (Jitsi). The Blue Jimp CEO is an active member of the WebRTC group within the IETF and the software is essentially focused on providing communication leveraging WebRTC technology.

The GPL licensed software benefited from significant contributions from both Blue Jimp and the open source community in its stride to become the "Open Source Skype". As of 2015, the Jitsi software portfolio was mature enough to realise a large scale web media communication platform (Jitsi Meet). The success was such that Atlassian acquired Blue Jimp in order to integrate Jitsi at the heart of Atlassian's own media communication solution (HipChat). Atlassian, who switched Jitsi's license from GPL to Apache in order to lower adoption barriers, are committed to support the Jitsi software. While Jitsi is now a wider project, encompassing multiple software modules, its objective has not changed.

3 Why Jitsi?

Among the various WebRTC solutions, the Jitsi software suite is the only open source solution proposing an SFU model that aims to provide a stable WebRTC web conference service. Since 2012, RENATER has worked in closed collaboration with Blue Jimp in order to improve Jitsi. The collaboration produced the following key features:

- Federated access to Jitsi Meet
- Multi-bridged Jitsi Meet configuration, boosting scalability

The collaboration helped produce a French National Web Conference Service called "RENdez-Vous" (https://rendez-vous.renater.fr), which was successfully launched in March 2015.

As detailed in the document "RENdez-Vous: One year of operation experience" (see https://wiki.geant.org/display/WRTC/GN4-1+WebRTC+Roadmap), the service has since enjoyed substantial traffic, and the feedback from the French community has been extremely positive. However, even in a production state, the service is relying on bleeding edge technology that need to get stabilised.

The RENdez-Vous service is still under development, and promising features will be implemented in order to improve the user communication experience.

4 Proof of Concept

4.1 PoC plan

The plan was to

1. Set up a simple Jitsi PoC installation
2. Assess the best architecture for a distributed Jitsi solution
3. Select the preferred/possible architecture(s) for a distributed solution
4. Validate solution with a simple manual test
5. Automate the process and test at larger scale

We were unable to complete step 5 since a key member of the project left for a new job before the PoC was completed. For this reason, this report only concerns steps 1–4.

4.2 Jitsi technology overview

The Jitsi Web Conference application is called Jitsi Meet and is accessible through a web browser (see Figure 1: RENdez-Vous User Interface).

Jitsi Meet key features include:

- Multi-party audio/video web conference via WebRTC compliant browser
- Up to 10 participants per virtual room is supported
- Jitsi rooms are accessible through a unique URL
- Room access can be secured
- An integrated chat is available
- As a web based technology, integration of/in third party components is easy (screen sharing, Etherpad etc.)

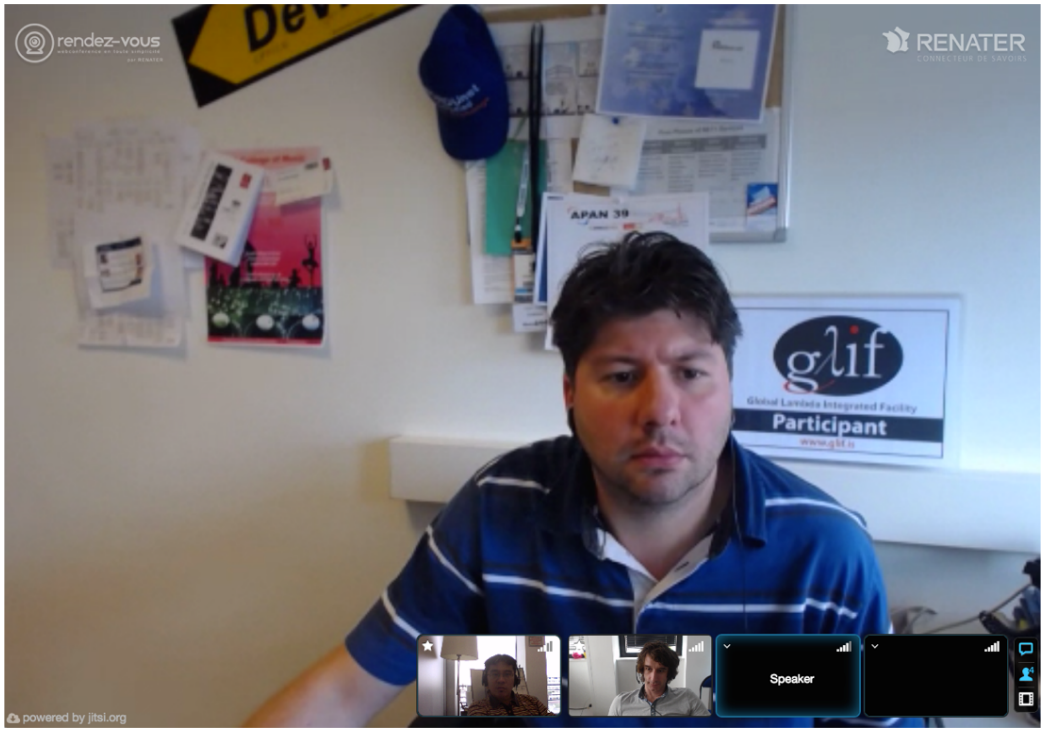

Figure 1: RENdez-Vous User Interface

4.3 POC set up

The first step was to implement a simple multi-party video conference proof of concept. In order to provide such a service, the Jitsi software suite requires the following as a minimum:

- Jitsi Meet
- Jicofo
- Jitsi Video Bridge.

The Jitsi suite also provide additional libraries, but these are feature add-ons and not part of the core multi-party video conference solution.

4.3.1 Jitsi Meet

Jitsi Meet is the user application front-end that runs in the web browser. It is an open source (MIT) WebRTC JavaScript application that, combined with the Jicofo and Jitsi Videobridge, provide high quality and scalable multi-party video conferences.

A conference is initiated by pointing the browser to the Jitsi Meet address, which is concatenated with a keyword corresponding to a room name. The RENdez-Vous service, for example, is accessed by using the pattern `https://rendez-vous.renater.fr/<keyword>`.

4.3.2 Jicofo

Jicofo (JItsi COnference Focus) is a server side focus component used in Jitsi Meet conferences. Conference focus is a mandatory component of the Jitsi Meet conferencing system next to the Jitsi Video Bridge. It is responsible for managing media sessions between participants and the video bridge. Communication between each participant is done through the Jingle protocol, while the Colibri protocol is used between Jicofo and the video bridge. Whenever a new conference is about to start, an (XMPP) info/query IQ message is sent to the component to allocate a new focus instance. A special focus participant then joins the multi user chat room. It creates a Jingle session between the Jitsi Video Bridge and the participant. Although the session in terms of XMPP is between the focus user and the participant, the media will flow between the participant and the video bridge. That's because the focus user will allocate Colibri channels on the bridge and use them as it's own Jingle transport.

4.3.3 Jitsi Videobridge

The Jitsi Video Bridge is a WebRTC compatible Selective Forwarding Unit (SFU) that allows for multiuser video communication. Unlike expensive video bridges built on dedicated hardware, the Jitsi Video Bridge 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. In an optimal Jitsi architecture, a multi-party video conference can be powered by multiple Jitsi Video Bridges by using the Jicofo component.

4.3.4 Jitsi POC configuration

A typical Jitsi environment deployment is depicted in Figure 2: Jitsi environment deployment.

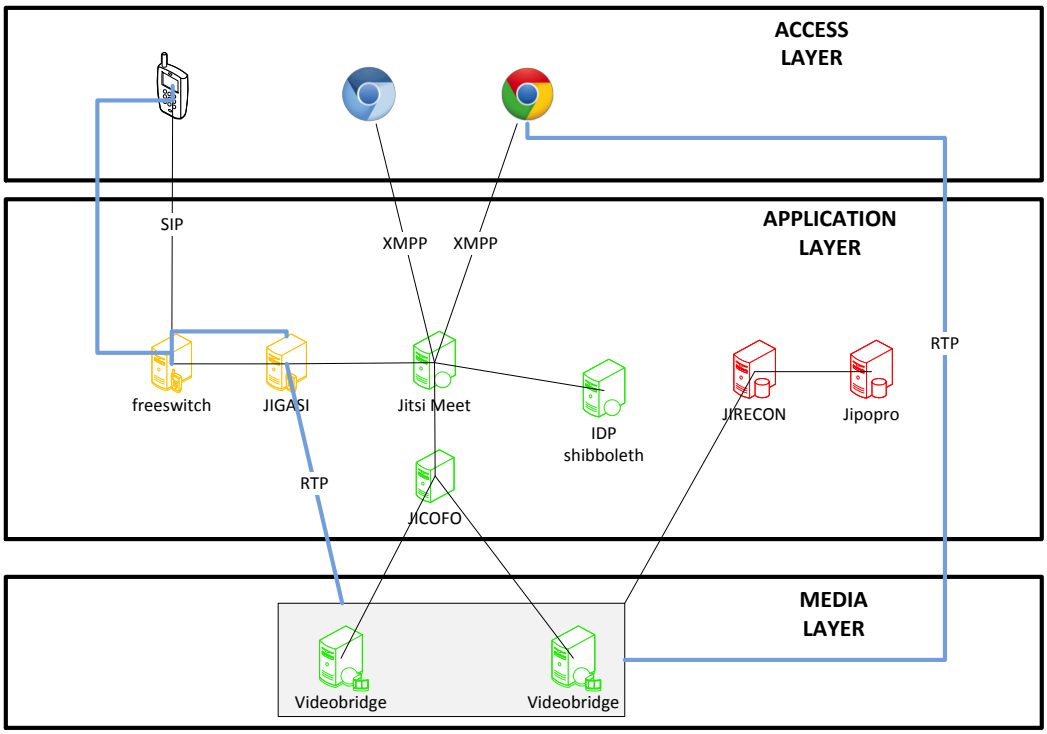

Figure 2: Jitsi environment deployment

This is the architecture evaluated in the RENdez-Vous pilot. Further to the video conference use case, the following additional features were proposed:

- Authentication through SAML federation (eduGAIN)
- Support for multiple video bridges
- Load balancing
- Statistics
- Logging (SDP browser collection)
- Rest API for JICOFO communication with our controlling resources system
- VoIP integration
- Recording capabilities
- High level API for rapid media applications development
- Reservation system integration
- A glimpse of SIP → WebRTC audio bridge

These features are provided by the following modules:

- JIGASI: Sip Audio Gateway
- Freeswitch: IPBX
- JRECON: recording system
- JIPOPRO: Recording post processing server

The last 4 features are in a working state and usable in a SOHO environment for experimental purpose.

4.4 WebRTC Technology & architectures consideration

Two main architectures may be considered when building a multi-party conference system; full-mesh and star model.

In the full-mesh configuration, as illustrated in Figure 3: WebRTC full-mesh model below, browser participants are directly connected to each other and requires no central infrastructure. This architecture is straightforward and induces cost savings. The communication flow uses the most direct network route, which should guarantee the lowest latency. However, the model is applicable only when there are few participants involved in the video session (e.g. one-to-one communication); peer-to-peer communication only works as long as it scales on the browser side. The current limit is typically around 3-4 participants, with factors such as CPU capabilities, GPU hardware acceleration, GPU drivers and code stability on the browser side affecting this number.

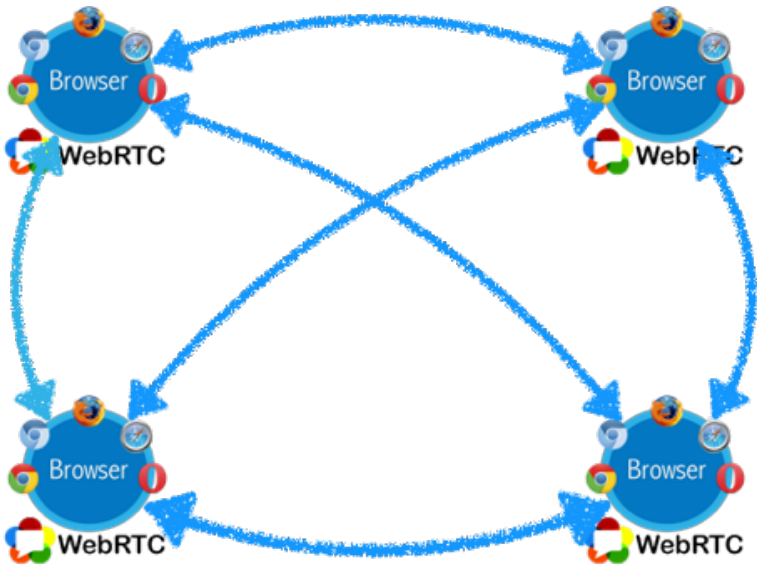

Figure 3: WebRTC full-mesh model

In the star (or bridge) model, each peer is maintaining one session with one central component that bridges the streams from each peer and acts as a relay between each browser participant. As expected, the complexity increases as more components are involved in the conference. The paths followed by the audio/video streams are indirect (relayed by the bridge), which increases the latency which again may affect the user experience. The bridge model should be considered for applications involving parallel room sessions. The workload on the central component depends on two key parameters; the number of streams per rooms and the number of rooms handled by the central component. Variants of the basic central star model may be implemented, where multiple bridges are deployed to balance the workload and thus providing a highly scalable service.

The WebRTC standard specifies how communication between two end points is done. Many WebRTC services use peer-to-peer connections to create a full mesh between all participating end points. This has an inherent scaling problem leading to limitations in the maximum number of participants able to participate in a multipart conference. Experience with existing web

conferencing solutions clearly indicates a solution for higher education needs to be able to support around 15 participants to facilitate online learning (group discussions). This number is not supported by peer-to-peer. Hence you need a box connecting all participants. This central component may be implemented in two different ways, through the use of MCU or SFU:

1. Implement/use a WebRTC MCU

In this case, all the flows circulate through a central bridge able to multiplex audio/video from each peer and output a single flow to each conference participants (H323 MCU model).

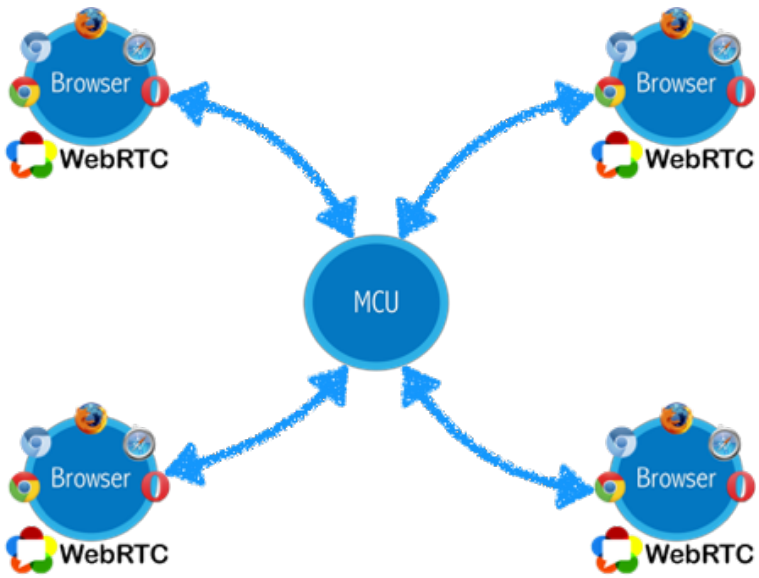

Figure 4: WebRTC MCU model

2. Use a WebRTC SFU

In this case, the bridge is only relaying the flows from the participants leaving the complexity to handle the audio/video streams to the browser application running the WebRTC library. Each SFU is restricted to its simplest duty: relaying a stream from one peer to the others. The net benefit of this is that the SFU functionality can be implemented into commodity hardware. In order to provide scalability, more than one SFU may be instantiated — a paradigm that makes sense in cloud environments to provide elasticity to the service.

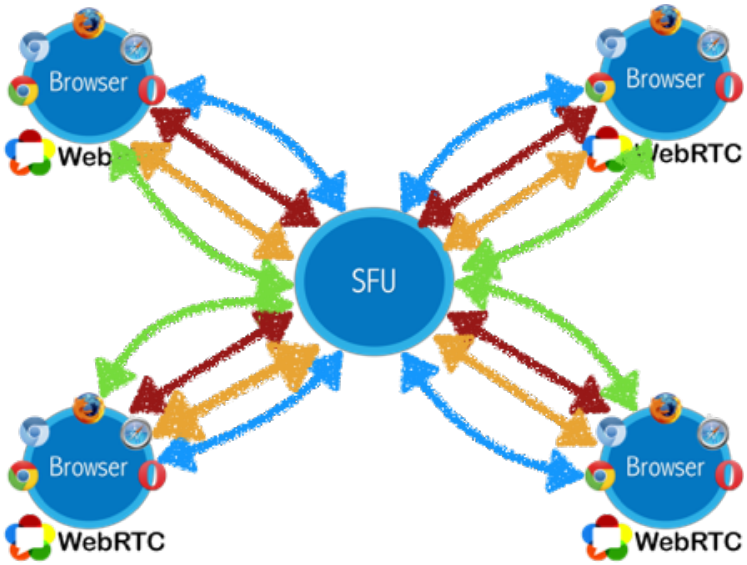

Figure 5: WebRTC SFU model

4.5 Technical options for distributed Jitsi

Jitsi provides an easy to use multi-conference service. It is the only project proposing an open source SFU and is therefore a good candidate for an organisation to build a DIY multiparty video conference service. The proof of concept demonstrated that the software is a good starting point to build such a service.

Scalability is not an issue. The Jitsi team published a stress test in order to evaluate the bridge performance; in order to handle 1000 video streams, a 550 Mbps bandwidth connection is required for a video bridge powered by 1 CPU used at 20% (https://jitsi.org/Projects/JitsiVideobridgePerformance). Jitsi Meet can work in a multi-bridge configuration in order to provide load balancing behavior via the JICOFO component, and there is theoretically no limitation regarding the number of users.

The Jitsi Meet application proposes a convenient way to integrate web collaborative applications, such as Etherpad, screen sharing functionality etc. In this case, we are using a conference application linked to other web applications.

A different approach is to integrate Jitsi Meet inside an existing web business application by utilising the Jitsi Meet API. However, the API is at stage too basic to support such functionality.

Building a service relying on the SFU model is the optimal solution, as it provides unmatched scalability at an affordable price when compared to traditional MCU solutions. While the current state of the Jitsi SFU is usable, it is not perfect. Additional features are being implemented in order to improve the Jitsi Video Bridge performance and thus user experience. Functionality such Last-N and Simulcast are proceeding.

In a web conference with many participants, people do not typically all speak at the same time. In the interest of preserving bandwidth and other resources (e.g. screen real estate and GPU), it is

therefore desirable to stream only the audio/video feeds of active participants. Last-N, a feature residing on the video bridge, does exactly this. It is designed to identify the Last-N individuals who spoke and only the audio/video feed from these N people will be relayed to the other participants. If a new individual speaks, she will become part of the Last-N and push someone else off the stack.

Simulcast is an additional feature that resides on both the browser and the video bridge. An important characteristic of many MCUs is the ability to modify the bandwidth of the streams they are sending to participating endpoints. This allows them to support diverse network conditions and devices. Since encoding and decoding are never performed on an SFU, it is important that adaptive bandwidth can be done in a different way. SFUs achieve this by subcontracting it to the endpoints; rather than sending a single high resolution stream, all endpoints encode their video streams into multiple resolutions. 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.

Leveraging the great momentum from the community, the Jitsi team continues their work as part of the Atlassian development and research engineering team. Apart from improving the video bridge performance in order to bring user experience at the next level, highly requested features like recording and elaboration of a full media API are under development.

As a web technology, hooking Jitsi Meet to SAML2-based identity federation has been discussed and studied within the group. Jitsi can now be coupled with eduGAIN so that room creation is restricted to the organisations being part of the federation.

Last but not least, RENATER conducted a “production pilot phase” based on the current version of Jitsi, the service is called RENdez-Vous. This service received a warm welcome and currently delivers 100 days of video conferences, for 10000 users distributed among 2500 virtual conference rooms per month.

4.6 Technology shortcomings in Jitsi

This section lists the limitations of the Jitsi Meet multi-party video conferencing solution compared to other solutions. Some limitations were expressed by the sample of users questioned and are, as such, not tied to the software itself but worth mentioning nonetheless.

The key differentiator between Microsoft Lync and Jitsi Meet is that the latter lacks a presence system.

A degradation of the audio/video quality start to be perceptible with around 5 participants. This is subject to investigation, but experiences from mesh-model based solutions (e.g. appear.in) provides a better experience than the star/SFU model.

While the multi-bridge architecture makes Jitsi Meet extremely scalable, the JICOFO and the Jitsi Meet components are identified as a single point of failure. At this stage, there is no clustering or high availability mechanism available at the JICOFO level and more architecture design validation is required. Atlassian’s Jitsi team is working hard on improving the current solution.

Aside from basic media control, the media API lacks further functionality. However, the current code is being heavily reworked in order to decouple the Jitsi Meet user interface from the Jitsi Video Bridge.

While it is possible to deploy a central European Jitsi environment, the architecture of the software does not allow multiple Jitsi Meet instances to be interconnected. The behavior is not implemented and might never be by Atlassian. Even if most NRENs should want a shared European desktop video conference service, many NRENs would likely want to have one national instance as well. Even though the national instances cannot be interconnected, parts of the infrastructure (e.g. videobridges) may be shared with the European instance and thereby power both.

A Pan-European multi-party video conference service based on Jitsi may be implemented using different approaches. Three possibilities of distributed architectures are available:

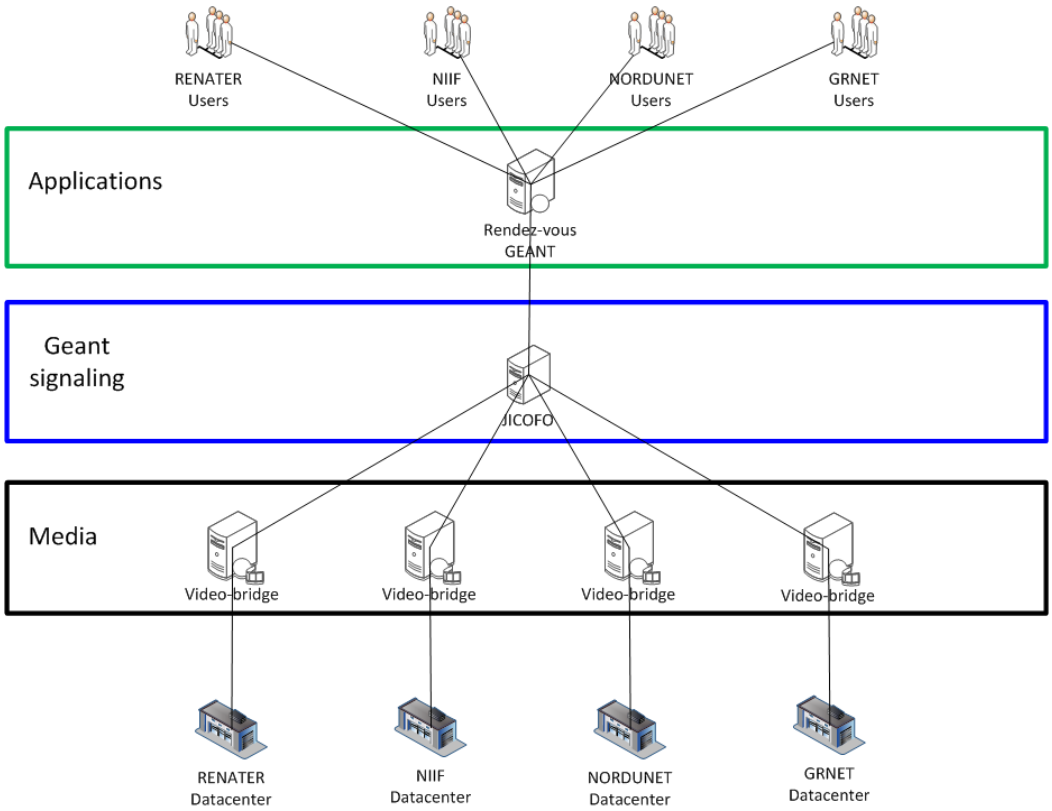

Figure 6: Approach #1

The first approach involves a central JICOFO, managed by GEANT, and a set of Jitsi Video Bridges distributed across GÉANT and NREN premises.

Note that Jitsi Meet exposes an API allowing video conference logic to be embedded seamlessly inside existing web application in order to improve user collaboration.

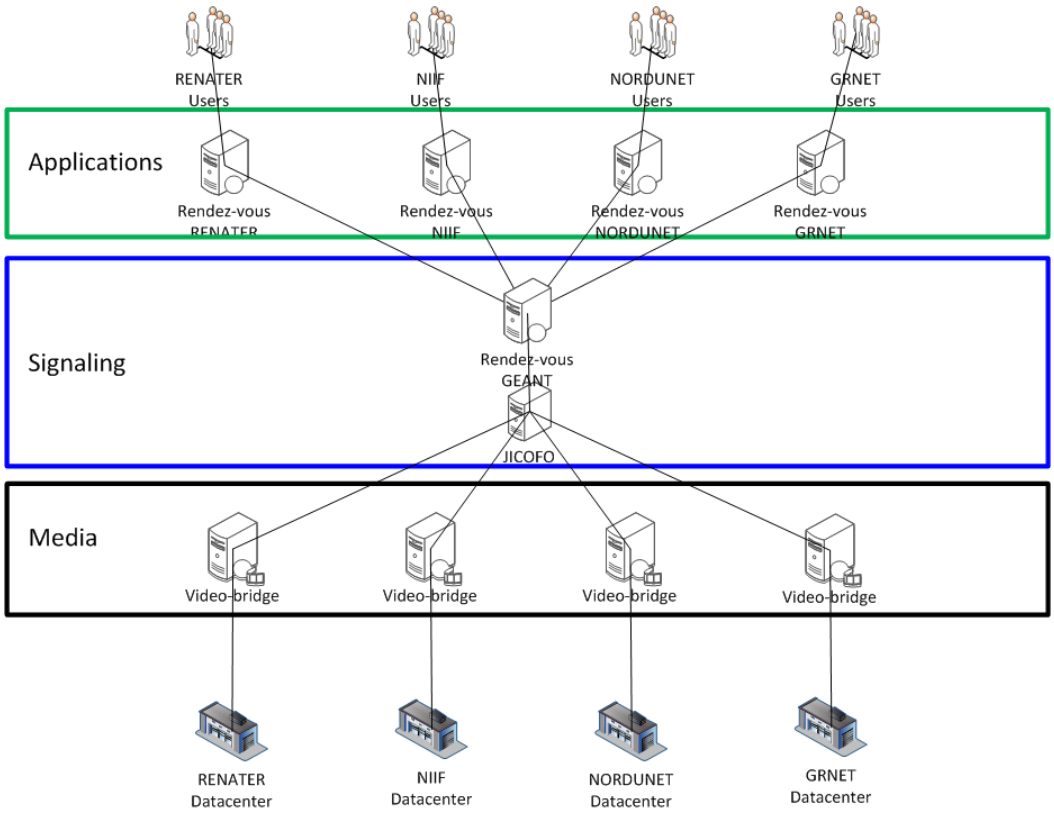

Figure 7: Approach #2

The second approach implements instances of Jitsi Meet dedicated to each domain (NREN), including the GÉANT domain, forming one single logical cluster leveraging the media horse power of the distributed video bridge.

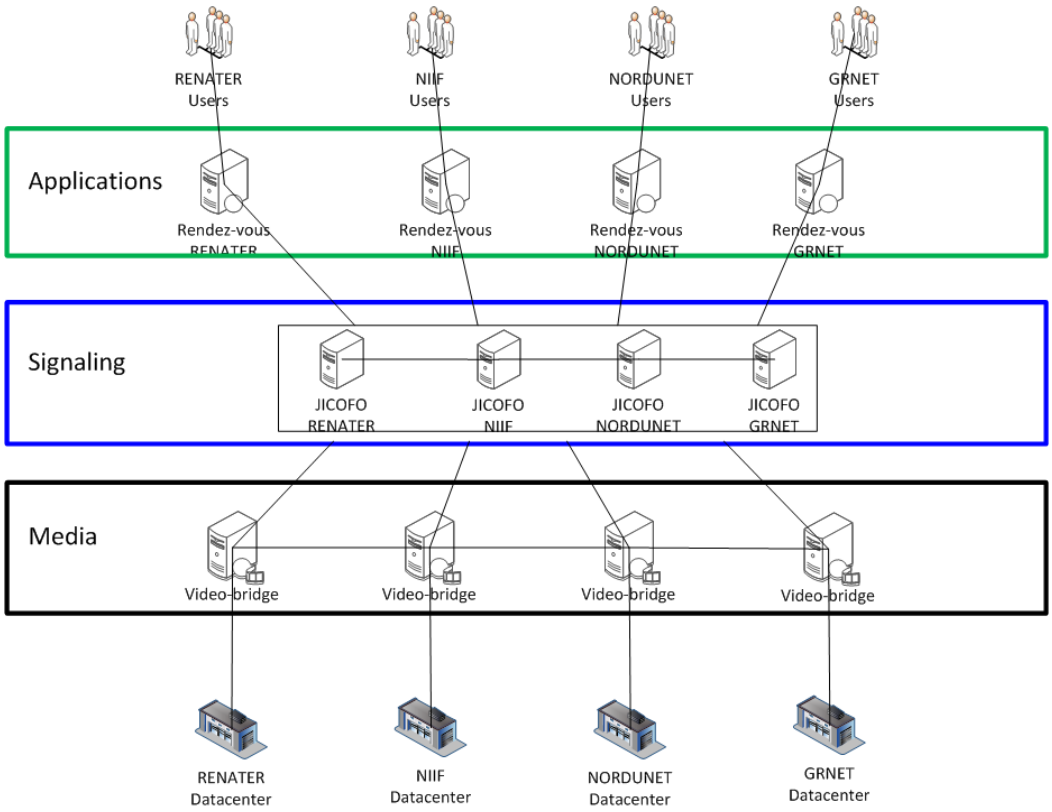

Figure 8: Approach #3

The third approach implements an instance of Jitsi Meet in each domain (NREN) connected to a cluster of JICOFOs. As one role of the JICOFO is to find the best bridge based on various criteria (load, utilisation, bandwidth available, RTT etc.), the cluster of JICOFOs will manage the session between the best video bridge and the participant.

Given the current state of the software, *only the first approach is feasible today*. While other alternatives are possible in theory, they require additional development.

4.7 Validation of basic distributed Jitsi

Approach #1 was tested by deploying SFU nodes at RENATER (France) and NIIF (Hungary), controlled from a central control node in France. The concept worked. Unfortunately, the project staff member undertaking the work left for a new job and there was not enough time available to replace his unique expertise in the project. As such, the automation of the process could not be validated, or tested on an even larger scale.

5 Conclusions

This technology scout reported on our investigations on the Jitsi software suite, in particular with regard to the end-user component of the multi-party video conference solution; Jitsi Meet. The middleware component, JICOFO, plays an important role in the Jitsi Meet service, as it manages user sessions on one or more Jitsi Videobridges in order to distribute CPU & network load. The Jitsi Videobridge lies at the heart of the Jitsi solution, as it plays the role of the SFU, which relays streams between all parties and handles conference room optimisations to improve the user experience.

In its current state, the solution only allows for one instance of the Jitsi Meet end-user component, for which a JICOFO and one or more Jitsi Videobridges may be added to the infrastructure. Hence, in a European solution, there can only be a single shared Jitsi Meet instance, one JICOFO (e.g. in the GEANT domain) and a number of Jitsi Videobridges within the various NRENs' IaaS.

A possible architecture worth to investigate further involves one Jitsi Meet instance per NREN domain (for those who want to offer their own national service) and one shared European instance at the GEANT level. Each NREN Jitsi Meet instance domain could share a subset of their JITSI Videobridge pool with the shared GEANT instance to allow service scaling.

Currently, there are no alternative open source solutions that can be use to build this type of DIY service. Although the Jitsi suite has its limitations, it holds great promise. Feedback received by RENATER in France (for the RENdez-Vous service) strongly put forward the technology's potential to fill a gap in the market.

Implementation, roll-out and maintenance for the Jitsi Meet service demand software development competencies within the participating NRENs, as well as software life cycle management expertise. Flexible and rapid application deployment in a mixed cloud environment allow the WebRTC service to benefit from cloud properties.

Usable solutions, such as Jitsi scouted in this document, have emerged. However, WebRTC technology is still in development and the standards are not yet completely stable. This means that any solution utilising WebRTC-based standards by necessity need to be kept aggressively up to date. When the standard changes, the clients change, the product changes and thus the product deployment needs to change. Implications are observed throughout the chain, with a particular impact on organisational expertise management.