Case Study — Jitsi

What is Jitsi?

Jitsi is a collection of free and open-source multiplatform voice, videoconferencing and instant messaging applications for the web platform, Windows, linux, Mac OS X and android.

Technology

a. Coding language used

Jitsi is written mostly in Java, and but it also contains part written in nativecode. If the project was started today, I think Java will still be the priority choice. For the obvious reason, Java is one of the most popular programming language, for one thing amazing about the project is that Jitsi aims at providing its service in a wide range of protocols and platforms. It allows people to make video and voice calls, share desktops, and exchange file and messages, over a number of different protocols ranging from the standardized XMPP (Extensible Messaging and Presence Protocol) and SIP (Session Initiation Protocol) to proprietary ones like Yahoo! and Windows Live Messenger (MSN). And as has been said before, the project should run on Microsoft Windows, Apple Mac OS X, Linux, and FreeBSD. Since Java is well known for its cross-platform, it will be the ideal choice of this project.

b. Build system/tools

According to the developer document on the projects official website, in order to build and run Jitsi, all you need is a recent JDK and ANT.

JDK

- Download the latest version on java.sun.com website
- Unzip the package
- Add <jdk_package_path>/bin to your PATH

ANT

- Download the latest version from http://ant.apache.org
- Unzip the package
- Add <ant_package_path>/bin to your PATH

Here's some useful ant command for the project,

To see a list of the most useful ant commands for the project, you can simply execute:

```
ant
```

To see a list of all the external ant targets in the project, you should execute:

```
ant -projecthelp
```

To safely build the project from the latest source and create all Oscar bundles, you need to execute:

```
ant rebuild
```

To run the application against your latest successful build, simply enter:

```
ant run
```

.. but be careful, because this might not reflect the latest source changes. If you want to pick up your latest changes first, then enter:

```
ant make run
```

.. or if you are worried about subtle dependency issues, use this "paranoid" command:

```
ant rebuild run
```

c. Features

- Unlike other videoconferencing technologies, Jitsi Videobridge, the heart of

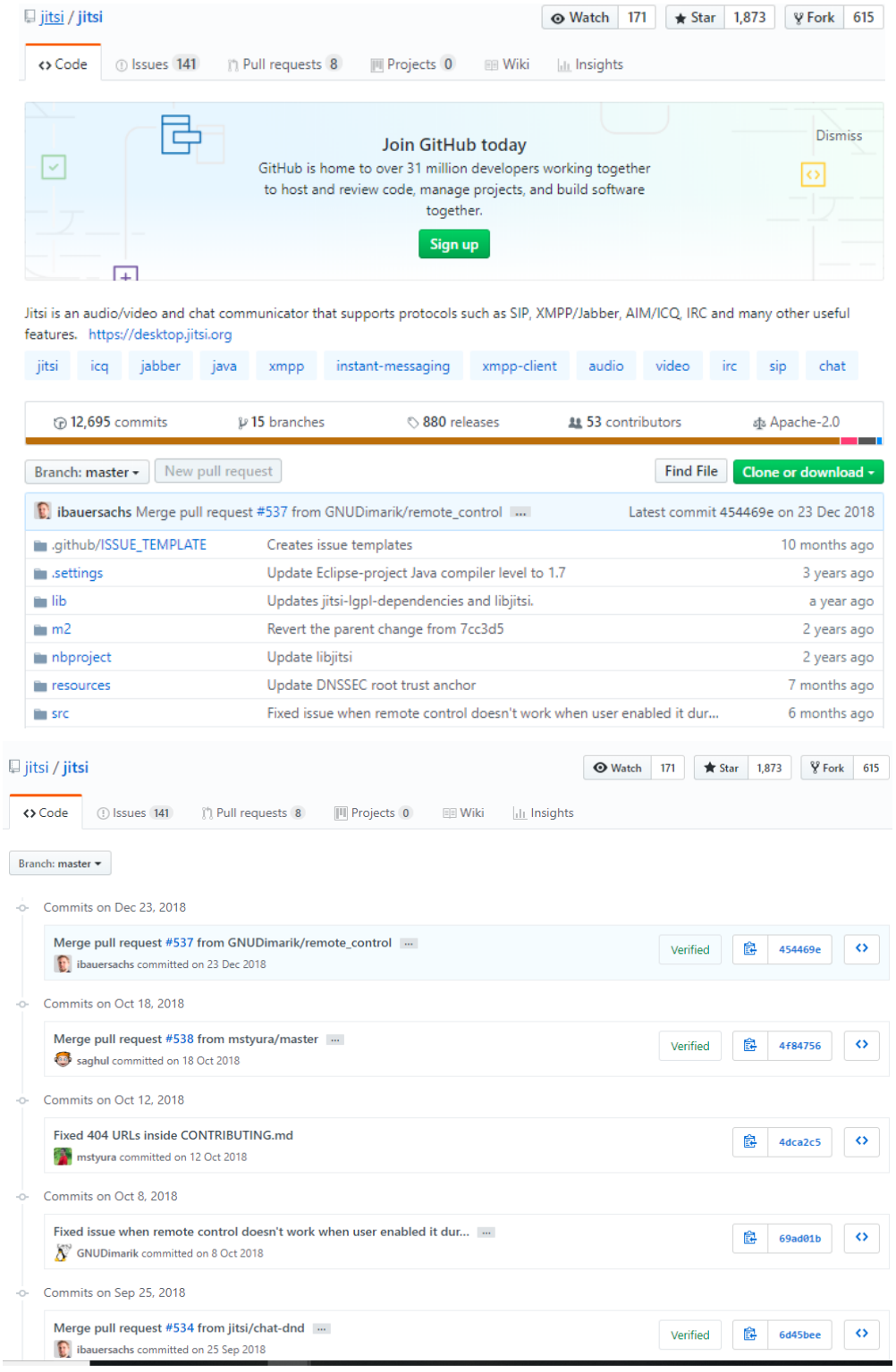
Jitsi, passes everyone's video and audio to all participants, rather than mixing them first.

- The result is lower latency, better quality and, if you are running your own service, a much more scalable and inexpensive solution.
- Jitsi is compatible with WebRTC, the open standard for Web communication.
- Jitsi supports advanced video routing concepts such as simulcast, bandwidth estimations, scalable video coding and many others.
- Ubuntu and Debian packages for easy installation

The core tech to ensure the fast speed is Jitsi VideoBridge.It is a video conferencing solution supporting the WebRTC that allows multiuser video communication. It is SFU and only forwards the selected streams to other participating users in the video conference call, therefore, CPU horsepower is not that critical for the performance.

- **Testing**

From my observation to Jitsi based on its github link, the project is not using any CI tools.



However, the developer document do introduce how to prepare and run the tests.

The project maintains a large number of unit tests. In order to make sure that everything is working as expected , the steps is provided as below:

For tests that deal with ICQ, you need to do the following:

- Create two ICQ accounts on icq.com/register
- In the lib directory, copy the accounts.properties.template file to a new file called accounts.properties.
- Configure the new file with the following parameters :

```
accounts.icq.TESTED_IMPL_ACCOUNT_ID = <first_account_uin>
accounts.icq.TESTED_IMPL_PWD = <first_account_password>

accounts.icq.TESTING_IMPL_ACCOUNT_ID = <second_account_uin>
accounts.icq.TESTING_IMPL_PWD = <second_account_password>

accounts.icq.CONTACT_LIST = group1.321947947 group2.269274750
 group3.294057493 group3.219630674
```

(The contact list above is an example, you could use other groups and users)

- Execute the following command :

```
ant rebuild
```

Test selection

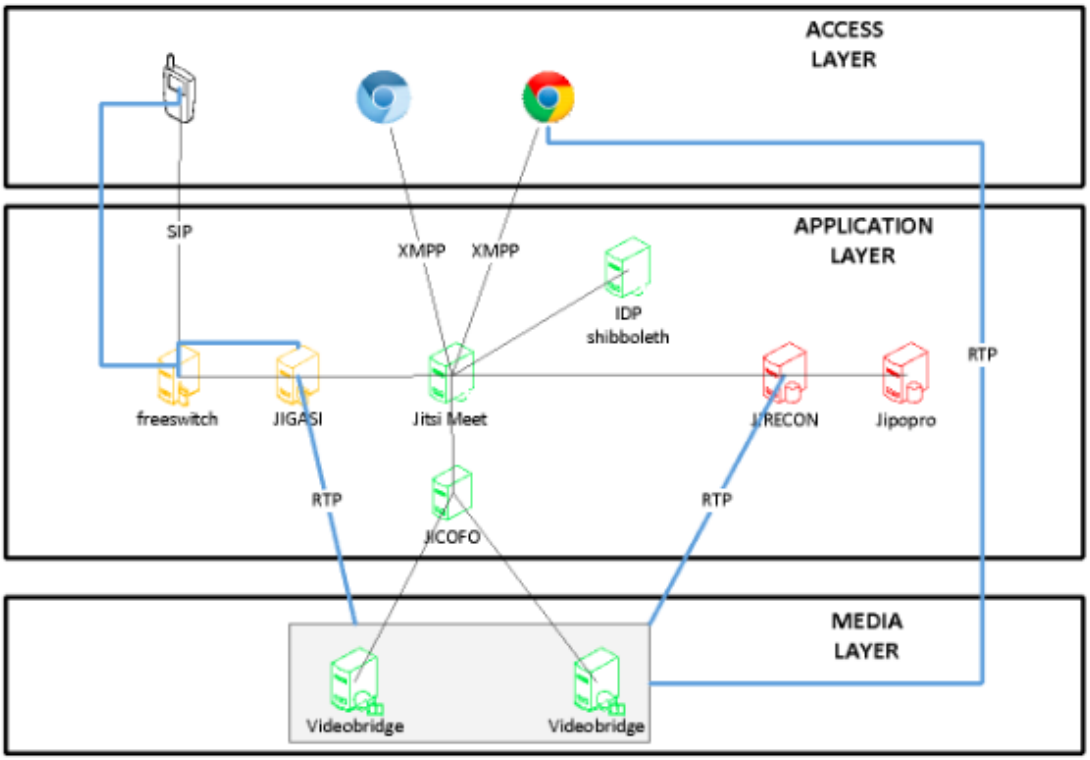
By default, the SVN repository holds a file called ./lib/testing.properties that causes all available unit test suites to be executed by the ant test target.

These test suites are often specific to an individual protocol, such as Jabber or MSN. Before these test suites can be run successfully, you will need to define suitable test user credentials as described in TestingProperties.

System Architecture

- Uses the Apache Felix OSGI implementation for modularity.
- Uses the JAIN-SIP protocol stack for SIP support and the Jive Software Smack library for XMPP
- Complete an ICE implementation called ice4j.org
- Audio systems supported are PortAudio, PulseAudio and WASPI

Here's a diagram I have found to show how Jitsi works



Overview:

- Access Layer : Web browser, which has a Jitsi Meet application executing in it.
- Application Layer : Jicofo, which provides basic signalling service, load balancing etc.
- Media Layer : Jitsi Video-bridge, which relays the video streams and is a selective forwarding unit (SFU).

Defects

Here are two issues I have found from GitHub:

#388 Jitsi not showing hints - extension status of other extensions (SIP)

The issue is close for its similarity to issue#394 and a possible reference of solution is listed below: https://github.com/jitsi/jitsi/wiki/Busy-Lamp-Field

Here's another recent issue

#579 Jitsi 1.0 snapshot compile errors

According to the reply, the developer have moved MediaType to the jitsi-util package(as `org.jitsi.utils.MediaType`) You need to update your app if you want to follow snapshots.

References

https://github.com/jitsi/jitsi

https://desktop.jitsi.org/Documentation/DeveloperDocumentation

http://www.tothenew.com/blog/demystifying-jitsi-2/

https://www.aosabook.org/en/jitsi.html