Video-conference system based on open-source software

Emanuel Silva
Instituto Superior Tcnico

***Abstract*—This thesis proposes a free video conference solution for multi-room presentation sessions. It considers a system with a speaker, a local audience and a remote audience.**

Even though there are many, well developed, video conference solutions that can perform this task they are typically expensive. The objective of this thesis is to develop a free solution, supported on low cost peripherals and computers, that can be extended to meet the requirements of the user.

The requirements set for this thesis are: dual view capabilities (to support a conference or presentation); bidirectional audio and video between the two rooms; a support on laptops hardware. This is achieved by using open source software for the interface, streaming and call management features. Moreover, it is also a goal of the proposed solution to communicate with current video conference solutions by using call signaling standards.

The implemented solution, can be run in a Linux operating system and has a web interface, where the multimedia streams are played and the application is controlled.

I. Introduction

A. Context

Video conference gives individuals the opportunity to communicate with each other in real time, using video and audio. Using video conference over the Internet Protocol (IP), todays panorama of video conference is submersed by capable applications which offer great value for long distance communications. Video adds another layer of touch between individuals when communicating with each other, which cannot be achieved with audio. Video conference systems are able to provide to all participants an experience that leads them feeling they are attending a real meeting [1]. Therefore, communication is facilitated, ideas become clearer and less time is consumed [2].

1) Presentations: The main problem with these solutions is that they are developed for communication between individuals, where everyone speaks and collaborates the same way for the call. This is different from a presentation. When attending a presentation there are two main attention subjects:

- Speaker
- Supporting Media

The speaker is the person which is leading the presentation. Often, the speaker uses supporting media to back up the information he is stating. The supporting media can be any type of media. Formerly score cards, slides or even blackboards were used. Nowadays the most used media types are electronic slides and video.

2) Video Conference Rooms: Video conference rooms aim to solve these issues. In these rooms it is installed hardware that enables a speaker to make a presentation to local and remote attendees. Remote attendants also need a video conference room, even though being slightly different from the speaker's video conference room. Thus, for this solution to work, it is needed two physically different video conference rooms with video conference equipment. These two video conference rooms will be from now on be called the speaker's room and the remote attendees room.

B. Application Domains

Most current laptops already have built-in microphones and web-cameras, making them the perfect tool for video conference presentations.

- **Education and Research**: A portable video-conference solution could help in a few tasks in academic world. Tasks such as: judging a thesis, present a subject, teach a class or participate in a conference meeting could be done just using a laptop and the video conference software.
- **Health Care**: There may be situations when a doctor cannot be present to lead a presentation which is very valuable to the subject of a congress. The doctors attending the presentation could be in a big auditorium, where the congress takes place, watching the remote presentation and communicate with the speakers by making questions. The speaker would also see his audience from his laptop enabling a visual communication by both parts.
- **Business**: Information technology organizations which develop software products need to schedule training sessions for the users. Instead of having to physically move to all the organizations that bought the software, they could schedule video conference sessions where the software specialist would give a presentation/class about the software to the users.

C. Objectives

The objectives of this article is to briefly present a video conference application specific for giving presentations. As it was seen before, there are plenty application domains for a portable and low cost video conference presentation. The work to be implemented, as discussed in the previous section, aims to fill this gap in video conference solutions available today. The main focus is to build a video-conference system with the following characteristics:

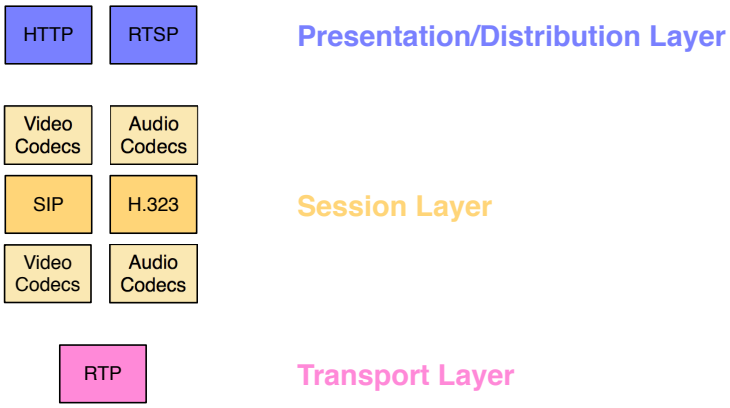

Fig. 1. Protocols/Standards Overview.

- Develop a video conference application for giving presentations;
- Use open source software (Framework and codecs) for a lower costs down to a free solution;
- Communicate with other video conference devices and systems using standard call signaling protocols;
- Use standard video and audio codecs.
- Run on any kind of personal computer (laptop or desktop);
- Use any kind of peripherals (microphones, webcams, etc).

II. Supporting Technologies

There are several supporting technologies in the context of a video conference solution that are worth mentioning. They can be described using the Open Systems Interconnection (OSI) model.

A video conference application needs to transmit multimedia streams to the user and transfer multimedia streams through the network. In figure 1, it illustrated the standards and the protocols that will be described later in this chapter. The figure is organized in a similar way of the Open Systems Interconnection (OSI) model. This is done because of the nature of the these protocols and standards. Most of them are networked based protocols and the others were defined (video and audio codecs) because of the bandwidth limits imposed by the IP network. The standards and the protocols were divided the following layers:

- **Presentation/Distribution Layer**: This layer introduces the protocols and the standards that are used to distribute media streams to the user. Hypertext Transfer Protocol (HTTP) and Real Time Streaming Protocol (RTSP) can be used to distribute the media streams received from a remote client to the user.
- **Session Layer**: Since this layer is a middleware between the presentation and the transport layers, all the call signaling and application logic is implemented here, in this layer. The Session Layer can be further subdivided in two categories:
  - **Call Signaling Protocols**: Call signaling protocols, like H.323 and Session Initiation Protocol (SIP), need to implement all the video conference logic in order to:
    1) Transfer audio and video over the network with other remote client.
    2) Transmit audio and video streams received from the remote client to the local user.

    H.323 and SIP standards manage multimedia call between two or more terminals. Therefore, they need to manage the handshake and the flow of media streams.
  - **Audio/Video Codecs**: Audio and video codec standards define multiple ways of encoding raw media streams. These media streams are required to have different properties depending what they are going to be used for.
- **Transport Layer**: The transport layer is used to transfer media streams and protocol specific information over the network. In the addition to the User Datagram Protocol (UDP) and Transmission Control Protocol (TCP) protocols that are usually defined in this layer, it will also be introduced a third protocol: Real Time Protocol (RTP). RTP is used to transfer live multimedia streams over the network to maximize the viewing Quality of Service (QoS).

III. Software Development Frameworks and Libraries

In the following sections it will be described the current frameworks available for developers to build video conference applications. These frameworks aim to deliver the developer, tools for implementing platform independent applications, as well as implementing the most common call signaling standards described earlier. These frameworks can be divided in:

- Video conference frameworks: These frameworks aim to deliver the developer, tools for implementing platform independent applications, as well as implementing the most common call signaling standards described earlier:
  - **PTLib**: PTLib is a framework library that enable developers to implement multi platform applications that can run on MS Windows, Unix, and Mac OSX operating systems.
  - **H.323 Plus**: H.323 Plus [3] is a protocol framework which aims to fully implement the H.323 call signaling Standard.
  - **Open Phone Abstraction Layer (OPAL)**: The OPAL framework implements both the H.323 and the SIP call signaling protocols. The OPAL main goal is to provide a framework that enables, by extending its API, the addition of other call signaling protocols. It uses the PTLib framework and it is also capable of acquiring video and audio data from input sources.

- Multimedia streamer frameworks: These frameworks or software packages offer multimedia streaming capabilities so that they can be used as part of a higher level software:
  - **FFServer**: FFServer is a module of FFMPEG[1] which acts as an encoding and streaming server for both audio and video streams. It supports several live feeds (with different qualities) with time shifting capabilities (that enables the user to seek to positions in the past on each live feed).
  - **GStreamer**: GStreamer is a framework for creating streaming media applications. It is possible to use GStreamer for just transcoding tasks as well as for streaming tasks. GStreamer uses a pipeline design (every step of the encoding/streaming process is separated from the other and takes the data from the previous one) which makes no extra overhead above what the applied filters induce.
  - **Flumotion**: Flumotion is a streaming software based on the GStreamer framework, which uses its encoding capabilities and adds more powerful streaming features, such as streaming using HTTP protocol. Flumotion aims to build an end-to-end software solution that integrates acquisition, encoding, multi-format transcoding and streaming of contents. It also allows the processing to be spread across multiple machines, so that the platform can scale to handle more viewers of more streams in more formats. All this is done using a modular approach.

IV. Current Solutions

In this chapter, it will be presented some currently technology solutions for video conference, either by using specific hardware implementations or software implementations.

- **Hardware Solutions**: Hardware solutions, also referred to as embedded systems for video conference, do not need any kind of personal computer since all the acquisition, codification, communication and reproduction is done by dedicated hardware boards.
  - **Aethra AVC 8400**: The Aethra AV 8400 [4] is a rack-mount hardware video conference system, in which the communication protocol relays on implementations of H.323 and H.320 Standards.The multiple features offered by Aethra AVC 8400 include multiple network connectivity, simultaneous dual-stream video, an Multipoint Control Unit (MCU) and a custom graphic user interface. The supported codecs to encode/decode transmitted/received video stream are: H.261 [5], H.263++ [6], H.264 [7] and H.239 [8]. The used audio codecs in this hardware device consist of: G.711 (56 kbps), G.722 (48/56 kbps), G.722.1 (24/32 kbps) and G.728 (16 kbps).
  - **Tandberg 990/880/770**: The Tandberg [9] hardware solution for video conference is a portable set-top unit already packed with a camera. Since it has built-in peripherals, all the external devices needer are a monitor, a microphone and speakers. This device can work both with H.323 and the SIP video conference signaling IP network protocols. Tandberg device features include an MCU, dual video streaming (DuoVideo), embedded security encryption, and IPv6 network support. The video codecs available on Tandberg line series are: H.261 [5], H.263 [6] and H.264 [7]. The audio codecs this device supports are the following: G.711 [10], G.722 [11], G.722.1 [12], G.728 [13] and 64/128bit MPEG4AAC-LD [14].
  - **Polycom QDX 6000**: The Polycom QDX 6000 device [15] aims to bring to the market a video conferencing product with emphasis on cost/quality relationship. This means that it does not necessary offers high quality features in order to offer the best compromise in terms of price in to its hardware competitors. The Polycom QDX 6000 implements both SIP and H.323 video conference protocols, which enables it to communicate through an IP Network with virtually all other available video conference solutions. It supports dual streams and comes with a camera, two microphones and a remote control. Polycom QDX 6000's video codecs include H.261, H.263 and H.264. In what concerns audio, the codecs available in this equipment are (among others): G.719, G.722, G.711. It also includes some Polycom proprietary audio codecs.
- **Software Solutions**: Software solutions assume that a personal computer with Internet connection will be used to run video conference applications. Software solutions (as hardware solutions) are commonly responsible for the communication and the encoding processes. Software solutions can also be only responsible for the communication protocols, being the codification process the responsibility of some kind of hardware codec.
  - **VCON vPoint HD**: vPoint HD [16] is a software-only client, running over MS Windows, which offers personal conference using either SIP, H.323 or H.320 signaling protocols. It can transmit and receive video, audio and data streams. vPoint HD incorporates H.329 protocol (HD DualStream™) for simultaneously sending and receiving video and data streams. Making use of VCON's SimulCast™technology, it can also participate in multi-conferences by being the chair leader or a participant. VCON vPoint's supported video codec standards consists of H.261, H.263+/++ and H.264. The supported set of audio codec standards include G.772.1, G.772.1, G.711, G.723, G.728, G.729 and AAC-LD, in a frequency range of 3.4KHz to 20KHz, thus offering a wide option of codecs to choose from, depending on

[1]FFMPEG is media format converter that can also grab from live multimedia sources.

the circumstances and network connection signal strength.

- **Skype**: Skype is a popular software application that allows users to make voice calls over the Internet [17]. Calls to other users, within the Skype service network are free, while calls to both Public switched telephone network (PTSN) and Integrated Services Digital Network (ISDN) telephones and mobile phones can be made for a fee using a debit-based user account system. Skype has also become popular for its additional features, which include instant messaging, file transfer, and video conferencing. Skype uses a proprietary telephony network protocol, called Skype Protocol. The protocol has not been made publicly available by Skype and official applications using the protocol are closed-source. Skype has a variety of clients for different platforms: Linux, Linux-based Maemo, Symbian S60, Mac OS X (Intel and PPC), iOS (iPhone and iPod Touch), Android, Microsoft Windows (2000, XP, Vista, 7, Mobile).
- **Ekiga**: Ekiga is an open source SoftPhone, video conferencing application over the internet. It was implemented using OPAL (and therefore Portable Tools Library (PTLib)) framework, so it is multi-platform compliant and can communicate with most VoIP service providers as it implements both SIP and H.323 Standards. Ekiga also offers standard telephony features support like call hold, call transfer, call forwarding and Dual-Tone Multi-Frequency (DTMF) giving the user the freedom to manage calls the way he wants to.
- **Asterisk**: Asterisk [18] is an open source communications platform, which can run on Unix, OpenBSD, FreeBSD and Mac OS X operating systems, that converts a personal computer into a voice communication server. Asterisk enables the deployment of several telephony applications and services, including: IP Private Branch Exchange (PBX)[2], VoIP gateways, call center, Automatic Call Distributor (ACD)[3] and Interactive Voice Response (IVR)[4] systems.
- **Adobe Connect**: Adobe Connect [19] is a proprietary web conferencing solution for Web Meetings, eLearning and Webinars. It is available to run in many different devices such as a PC, Tablet or Smartphone, due to it's web based interface. In fact, since Adobe Connect's web interface is based on Adobe Flash technology, this makes it easier for users to join sessions, since there is little to none setup required. For mobile devices, Adobe developed Blackberry, Android and iOS applications, so that Adobe Connect can be run virtually on every device.
- **Microsoft Lync**: Microsoft Lync [20] is an instant messaging client with support for video/audio conferencing and collaboration tools. Its main advantage over its competitors is the unified approach Microsoft took in developing Lync. It can be linked with other Microsoft products such as Word, Sharepoint and Outlook. This makes the users feel immersed in a collaboration/communication bubble. Lync acts as the communication and collaboration brain in a Microsoft deployment environment. Microsoft Lync can be accessed by various medium. These include: Microsoft Windows, web access, mobile devices and even Mac OSX. This does not mean that every feature is available in every device. Features go as limited as only being able to use instant messaging.

[2]PBX stands for Private Branch Exchange, which is a private telephone network used within a company. The users of the PBX phone system share a number of outside lines for making external phone calls.

[3]Automated Call Distribution is a device or system that distributes incoming calls to a specific group of terminals that agents use.

[4]Interactive Voice Response is a technology that allows a computer to interact with humans through the use of voice and DTMF tones input via keypad.

Solutions \ Features	SIP	H.323	Multiple Video Sources	Multipoint Control Unit (video)	Free
Aethra AVC 8400	✗	✓	✓	✓	✗
Tandberg 990/880/770	✓	✓	✓	✓	✗
Polycom QDX 6000	✓	✓	✓	✓	✗
VCON vPointHD	✓	✓	✓	✓	✗
Skype	✗	✗	✗	✓	✗*
Ekiga	✓	✓	✗	✗	✓
Asterisk	✓	✓	✗	✗	✓
Adobe Connect	✗	✗**	✓	✓	✗
Microsoft Lync	✗	✗**	✓	✓	✗

* Skype offers support for a three way (or more) video conference only in paid version.
** H.323 compatibility can only be achieved using a proprietary gateway and additional tweaking.

Fig. 2. Current solutions comparison table

To conclude the presentation of the several solutions available in the market that were studied in this chapter, a table (illustrated in figure 2) will be presented for a simple comparison between all of them. The features that were compared are those that are considered as the most important for this research. Among them, a particularly importance was given to its compatibility with other solutions (if it uses H.323 and/or SIP protocols), support for dual streaming, support for more than two participants and its market availability (free/paid).

The conclusions that can be taken from this table can be broken down into the following:

- Most solutions make their top priority the usage of standard communication protocols. The choice of H.323 and SIP is also explained because they are communication standards.
- None of those solutions provide simultaneous multiple video sources for free.

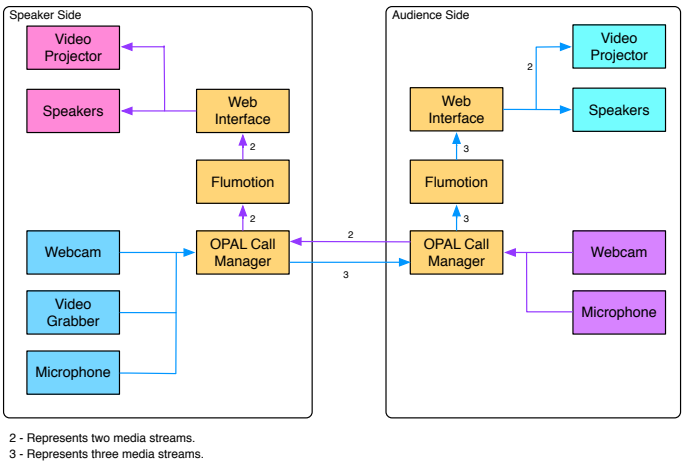

Fig. 3. Global system architecture.

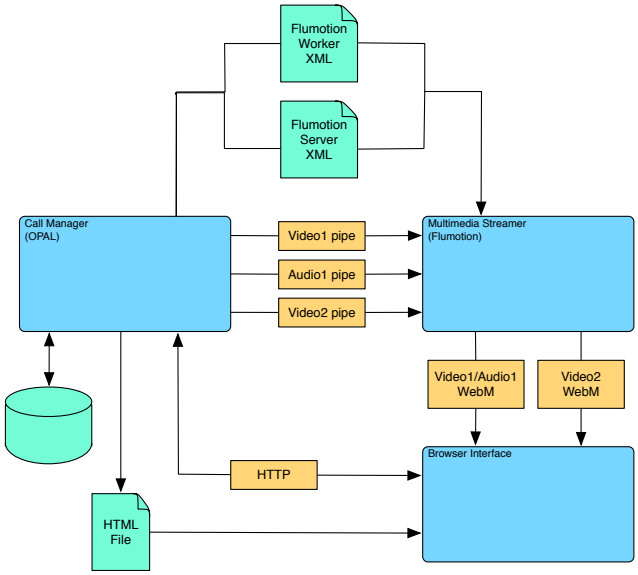

Fig. 4. Detailed solution's modules interaction

- There are not many free solutions that focus on video conference. As it can be seen, only three solutions are free to use in this set.
- Perhaps the most important conclusion that it can be taken from this table is that currently there is not any free solution that can be used for a presentation session and properly satisfy its requirements (dual video stream being the most important requirement).

V. Proposed Architecture

Figure 3 illustrates an overview of the global system architecture of the proposed system. Since there will be a speaker in one side, and an audience in the other side, there will be two possible configurations of an instance of the system.

The speaker side will feature a webcam to capture the speaker, a video grabber to capture the supporting media (i.e. presentation slides) and a microphone to capture the speaker's voice. On the same side, the speaker's side, it will also feature a video projector to display the audience and the speakers to also play the audience questions.

In the audience side, there will only be one webcam and one microphone to correspondingly capture the audience's image and their questions. The audience also features the speakers for reproducing the speaker's voice and the video projector, this time, for playing both the speaker's video and the supporting media video.

The architecture comprises three distinct modules: a **Call Manager** (implemented by the OPAL framework), a **Multimedia Streamer** (implemented by Flumotion) and an **Interface** (which is implemented by an HTML web interface). They interact in different ways with each other and most of the communication between them has the specific purpose of transferring multimedia streams.

The **Call Manager** module is responsible for all the communications with the remote clients, such as the acquisition and dispatch of video and audio, the capabilities management and all other aspects of call management. As an example, it is the **Call Manager**'s responsibility to initiate the process of answering and establishing a call. This process implies the acquisition of video signal from a webcam or a screen capture device and the audio signal from a microphone. It is also in the **Call Manager**'s tasks to negotiate what will be the codecs used in the call as well as what video/audio streams are to be sent and received.

The **Multimedia Streamer** module serves as a bridge between the **Call Manager** and the **Interface**. Its only job is to stream the multimedia streams it receives from the **Call Manager** to the **Interface** module. It implements a streaming standard in order to successfully stream without hassles and so that the **Interface** can present the streams as smoothly as possible.

Lastly, the **Interface** module's work is as the name suggests the visible part of the whole solution's architecture. The **Interface** presents the multimedia streams to the user, as well as it provides the user with the power to interact with the application. This interactions include the change of video definitions, accept calls and start calls.

A. System description and framework integration

In this section it will be described in better detail how the three described modules communicate with each other in order to implement the final solution. In past sections, the three modules were introduced and their responsibilities were also defined. In this section, however, the main objective is to describe the connections between them, as it is showed in figure 3.

Figure 4 illustrates, in greater detail, the connections and interactions between each module.

OPAL and Flumotion Integration: The OPAL call manager and the Flumotion stream server modules are the first in the chain between the acquisition of remote video and the presentation of the composite multimedia streams in the interface. As it can be seen in figure 4, OPAL sends not only the singular media streams, but it also starts the Flumotion Server through the generation of the configuration XML files.

OPAL and Interface Integration: The integration between OPAL and the HTML interface is done using an HTTP server embedded in the Call Manager. The developed HTTP Server is responsible for updating the interface, as well as dealing with the requests from the browser interface module. The HTTP Server is an important submodule of the whole architecture and it can be seen as middle software part between the Call Manager and the Interface. All information that is passed between these modules is passed through this server.

Flumotion and Interface Integration: The integration between Flumotion and the interface is the last and most simple integration procedure that takes place in the proposed architecture. The integration between Flumotion and the Interface is virtually almost transparent to the developer. Since Flumotion streaming server sends the WebM streams through HTTP, all that has to be done is to point the HTML5 video tag to the url that Flumotion generates.

VI. System Implementation

In this chapter, it will be presented how the proposed architecture presented in the previous chapter was implemented and what choices were made over the course of the development phase.

A. Call Manager Architecture

The Call Manager module was implemented by an OPAL application. When OPAL was introduced in chapter 3, the main classes that compose OPAL were also described. In this section, it will be described which classes were redefined and extended and how they were implemented, in order to develop the final solution.

In figure 5 it is illustrated the set of involved classes, when a call is taking place. What is important to refer about this figure is the existence of an hierarchy of class instances at each moment of the established call. The manager manages the endpoints and informs each one that a new call can be answered.

The endpoints's job are to be aware of outside information. In the case of the H.323/SIP endpoint, its job is to be listening for remote calls, as well as to send the media streams from inside of OPAL to the remote client. The local endpoint is used to communicate with the user and with the media inputs, such as a webcam or a microphone. The Call object represents the call and it is responsible for managing both an H.323/SIP Connection and Local Connection. The Local Connection is responsible for transporting the local media streams (that acquire the audio and video input sources) to the H.323/SIP Connection, as well as, to received the media streams from

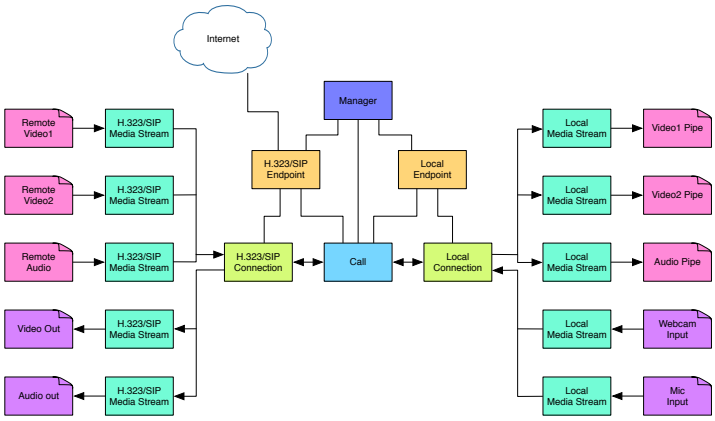

Fig. 5. Detailed OPAL architecture.

the H.323/SIP Connection. Each Connection has associated MediaStreams, that represent the type of information that is being passed from one connection to the other.

The MediaStreams transport information from one side to the other. These sides vary depending on which MediaStream is being used. In the example of figure 5, it is shown that the LocalConnection's job is to acquire media streams from hardware (mic, webcam) and to transport it to the remote client. It is also its job to acquire the remote media from the H.323/SIP Endpoint and export it to a named piped where Flumotion will be connected to.

It is important to note that the instances for H.323 and SIP in figure 5 are being seen as one single instance. This is not the case in reality and it was presented this way to simplify the diagram.

1) Manager's Implementation: The Manager is a singleton object (only has one instance) in the architecture and is the brain of the whole application. It orchestrates what happens when an instance is created, as well as the initialization of every object and the callbacks from every class.

a) Local Endpoint's Implementation: The local endpoint plays an important role in the conceived architecture. Since its role is to interface with the user, it is important to describe how it was implemented. The local endpoint has also the task of selecting the audio and video devices to grab the media streams from, according to the settings. Finally, it also has to redirect the streams received from the H.323 and/or SIP endpoints to named pipes. In the following paragraphs, it will be presented a brief description of those tasks:

b) Starting and accepting a call: The HTML interface has the task of presenting and grabbing information from the user. It has two important tasks: enable the initiation of a call, and signaling a new call to the user (from a remote client). For the completion of both tasks, the HTML Interface needs to communicate with the Local endpoint. This is done by means of the HTTP Server, which is embedded in the Call Manager module. In case there is a call waiting to be answered when the HTML page refreshes, the HTTP Server returns a ringing page so that the user can accept or reject the call.

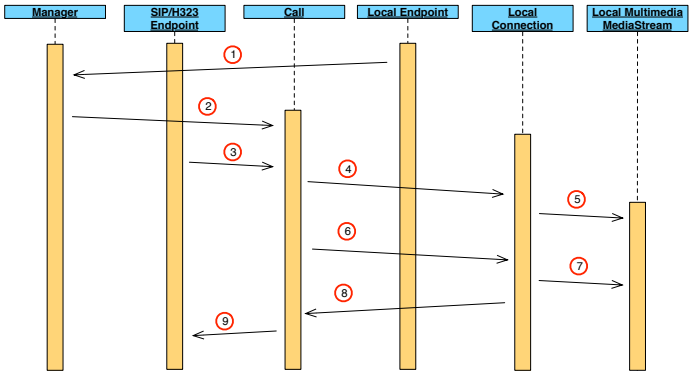

Fig. 6. Call flow state diagram.

c) Grabbing the video/audio media streams: Another task of the local endpoint is to define the devices that should be used to grab the video/audio streams. This is done when the Local Connection creates a new source stream. At this time, it should identify the device that will be used to grab the media stream from the application settings and create the appropriate media stream: OpalAudioMediaStream for audio streams and OpalVideoMediaStream for video streams.

d) Writing to the named pipes: The communication between the Call Manager and the Streaming Module (Flumotion) is based on named pipes that transport the raw media streams. The creation and management of those named pipes are the responsibility of the Local Connection object. Just like the grabbing process, at the time the Local Connection creates a sink stream, it also needs to create a named pipe to write the respective media data.

B. Call Flow

To provide a clearer vision on how the OPAL's objects work together to implement the call manager, in this section it will be described the call flow procedure as it is illustrated in figure 6. Note that in this figure, it is only illustrated one media stream. In a real case scenario, it would be two media streams received (video and audio) and three media streams transmitted (two video streams and one audio streams), in case of the audience side. The corded number in this figure represent interactions between the objects:

1) The Local Endpoint informs the Manager that it wants to establish a new call.
2) The Manager creates a Call instance that signals a remote client through the H.323/SIP Endpoint.
3) When the call is established the H.323/SIP Endpoint sends to the Call (through an H.323/SIP Connection) each media stream (H.323/SIP MediaStream) the will be used in the call.
4) For each media stream that is being sent by the remote client (in this figure it is only being illustrated one stream) the call instance asks the Local Connection to generate one Local MediaStream.
5) The Local Connection creates the Local MediaStream which will sink the media to a named pipe.
6) The Call instance will also create output MediaStreams to send to the remote client (H.323/SIP Endpoint). To do this is, it asks the Local Connection to create those MediaStreams.
7) The Local Connection creates a Local MediaStream. This Local MediaStream is responsible for acquiring media from a webcam, audio input, or any other input device.
8) The Local Connection sends the created MediaStream to the Call instance.
9) The Call instance redirects the MediaStream to the H.323/SIP Endpoint, that again sends the media streams to the remote client.

In figure 6 it is illustrated the local side of the Call instance in a much greater detail than the H.323/SIP side. Despite that, the process is similar to the local side. It was omitted from this figure in order to not overcomplicate it.

C. HTTP Server

The HTTP Server is a module inside the Call Manager as it was seen before, its job is to serve as a backbone to the interface module. The HTTP Server runs in a different port than the standard HTTP (80) and receives POST and GET requests from the Interface module.

The HTTP Server was implemented using PTLib HTTP classes. In particular, the PHTTPServer, class implements the HTTP Server which is initialized to listen for connections in a specified port. In this implementation, the server is initialized from port 8080 to above. This means that if port 8080 is busy, the server will try to listen on 8081, 8082, 8083, and so on.

The PHTTPServer uses virtual methods to process the received POST and GET requests. They are called by OnPost and OnGet. This is where all the server implementation takes place by implementing a specific handler for each of interface's website events: standby, ringing, settings, oncall, etc.

The OnGet method is used to pass dynamic information to the user. For example, when a remote user is calling, the interface displays its name. What really happens is that OnGet substitutes the "remote client" token that is present on the interface HTML file.

The OnPost method is used when there is a text form available for the user to submit. For example, when the user wants to place a call, he needs to insert the address of the remote client to be called. This address needs to be passed to the OPAL Manager. This is done using a form tag in the HTML code, that is then handled by the OnPost method.

D. Settings Storage

The Settings Storage module is nothing more than a way to store, in a persistent medium some video conference information such as what device to capture video/audio from. Since an objective of this work is to provide a base for future development, the settings were implemented so that it would be easy to extend the way of how to persist them.

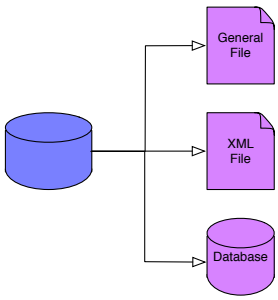

Fig. 7. Settings storage module

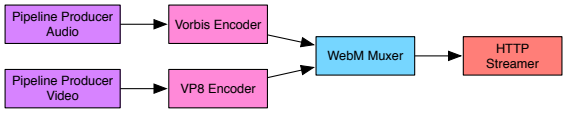

Fig. 8. Flumotion processing pipeline.

This is illustrated in figure 7. To achieve this goal, an abstract class VCSettings was implemented with the methods: Load, Store, SetProperty and GetProperty. All the other classes that implement the storage of options will derive from this class and implement the Load and Store methods.

In this solution, a simple text file was used to extend the VCSettings class. It's name is VCSettingsTXT. It simply stores each property in two text lines. The first line is the property name and the second line is its value. To handle the file writing and reading functions, it was also used a PTLib class: PTextFile.

E. Flumotion Stream Server Configuration

Figure 8 illustrates the processing pipeline that was implemented in the Flumotion Server. As it was discussed before, The Streaming Module is implemented by two Flumotion instances. This figure represents the particular Flumotion instance that is used with both audio and video streams. The other Flumotion instance pipeline is similar to this one, with the exception of the missing audio component, which will be used to transmit the supporting media video.

F. HTML Interface

In this section, it will be described the developed HTML Interface. As it was discussed in the previous chapters, the interface module is implemented by a web page. This way, users can use whichever browser it suits them more. Since the media streams are being delivered within a WebM container, which is natively supported by the HTML5 standard, every browser that adopts HTML5 will be able to play WebM multimedia streams that are being streamed from Flumotion. It is also an advantage to use this implementation approach because it is not dependent of the operating system (or Linux distribution).

Fig. 9. Solution's screenshot with two video sources.

The Web Interface is composed of two IFrames[5]. One for presenting context information and another one for presenting two multimedia streams that are being streamed through Flumotion. This separation was adopted so that when the user is on a call, the system can refresh the context information frame automatically without disturbing the multimedia streams on the other frame. The context information frame is composed of several web pages, they are:

- **Standby**: The standby page is the entry point in the system. It only has one single option to place a call through a text box.
- **Settings**: The settings page enables the user to choose what devices should the system use to capture video/audio.
- **Ringing**: The ringing page is a transitory page that has the purpose of showing context information to the user. When this page is presented for the user, the user will know that the call that was placed by him is ringing on the remote side. This pages refreshes itself every second, so that it can acquire from the HTTP Server the current status of the call.
- **Accept Call**: In the accept call page the user can accept or reject an incoming call.
- **In Call**: The in call page, illustrated in figure 9, is presented when the user has either accepted a call or a call has been accepted by a remote client.

VII. Evaluation Methodology

To evaluate the solution, described in the previous chapter, that was implemented it is necessary to define the parameters that serve as an evaluation meter to how the final solution performs. Such parameters are essential to characterize the solution in terms of performance and correctness.

After the parameters are defined, the solution must be tested against those parameters. The tests should be close to

[5]Iframe stands for inline frame and is used to embed another document within the current HTML document.

real case scenarios where the solutions would be used. The tests should also be made on a controlled environment to assure the results are completely dependent on the solution's implementation, leaving off problems that can happen with the environment (network, hardware, etc).

In the next sections, both the parameters and the tests will be defined. After the tests are conducted, the final work is to document the results. Therefore the results are a cross between the parameters and the tests and serve as a proof to how the solutions works.

A. Parameters Definition

To test how the solutions works, there is a need to define the parameters that, having in account what type of use the solutions will have, can describe the solution in terms of performance and correctness measures. These parameters, again, should be chosen having in account the usage of the solution in its real usage cases. It is important to have the context of solution in mind when defining such parameters. The parameters are:

- Audio and Video Synchronization.
- Delay.
- H.323/SIP Compatibility.
- Codec Compatibility.

B. Tests Definition

In this section it will be defined the tests that will be conducted in order to check the parameters that were defined above and to cross them both and derive the results. The tests will be divided in two major categories: self-contained tests and integration with other video conference solutions. The self-contained tests are tests that are performed with two instances of the solution implemented. The integration with other video conference solutions are tests that check the implemented solution compatibility with other video conference solutions.

The environment setup used to perform these tests are limited to the hardware at disposal. It is needed two instances of the solution running in separate computers. The use of two virtual machines in the same computer is not an option due to having the need to use multiple web cameras and audio reproduction systems. Again, the more close to reality in a controlled environment the better.

In one side it was used a MacBook Pro laptop running Ubuntu in a VMWare virtual machine. Connected to the VMWare was a Logitec Webcam that also features an audio input. The other video input used was the built in macbook camera. In the other side the setup used consisted in a laptop running Ubuntu with a Logitec webcam (and identical one to the described above) and a video grabber (for screen sharing).

To control the network it was also used a router without internet connection. The router connected the two setups together using ethernet.

Parameters / Solutions	Audio and Video Synchronization	Delay	H.323/SIP Compatibility	Codec Compatibility
Implemented Solution	✓	✓	✓	✓
Ekiga	✓	✓	✓	✓
Lynphone			✓	✗

Fig. 10. Evaluation results table

The tests that were performed using this environment setup were:

- Establish a connection between one instance to the other using just one video source.
- Establish a connection using two video sources.
- Make the same tests switching the instance sides.
- Call from the implemented solution to software video conference solution.
- Call from the video conference solution to the implemented solution.

C. Results

This evaluation table, illustrated on figure 10, represents in lines the solutions that the implemented solutions was tested with and in columns the evaluated parameters. It should be noted that delay is marked in the table as a binary parameter but in fact has a temporal measure. In this table it is only illustrated if the delay was acceptable or not.

The self-contained tests showed that the application can communicate successfully communicate with itself. Both H.323 and SIP protocols worked seamless without any problems.

The delay was the same from each side: 6 seconds. This is explained by a slow streaming speed of Flumotion and the time that the VP8 encoder takes to encode RAW video. Flumotion depends on large buffers to stream without breaks throughout the call. Since there is no way of setting the Flumotion buffers, nothing can be done about it. Another cause of delay is the VP8 encoder. Since VP8 (and WeBM) is a recent open source standard, there isn't a fast VP8 encoder available yet.

To finish the self contained test results, the synchronization between audio and video worked fine. There was no problem deriving the audio from the video or vice-versa. The synchronization was not perfect but it was very close, with few milliseconds of delay that didn't disrupt lip-sync.

The results for Ekiga show that Ekiga is compatible in every way with the implemented solution. There is a compatibility both in H.323 and SIP protocols. The codecs were also compatible both in audio and video. The synchronization (lip-sync) was as much as milliseconds. Which means that the user see almost instantaneously the video and the audio. The delay between the implemented solution's streams to Ekiga was 2 seconds. In the other part the implemented solution only received Ekiga's streams at about 6 seconds after the

streams were captured.

Finally, the results for Linphone were not so satisfactory. Firstly, Linphone only supports SIP protocol and from the tests the protocol worked fine when establishing new calls. The problem appeared when the media streams were created. There weren't fully compatible video codecs so there were errors creating the media streams and therefore the call terminates abnormally. Since the call is not initiated correctly, the delay and synchronization between video and audio could not be tested. Nevertheless, since Ekiga tests worked well, it is expected that these parameters would assume the same values as Ekiga tests.

VIII. Conclusion

The implemented solution imposed several challenges that were hard to overcome. The main work that had to be done was to integrate the different technologies into one final solution package. With the different technologies that compose the final solution: such as OPAL, Flumotion and HTML, the challenge was to find the best way to connect them all the best way possible. Thus, in the final solution, it was suggested named pipes (to connect the streams from OAPL and Flumotion), a custom web server (to take requests from the HTML interface) and WebM container (to present multimedia streams in the HTML interface without the need to install third party browser plugins).

A. Advantages

The implemented solution uses open source software. The use of open source frameworks, instead of a custom developed closed framework for a specific problem, guarantees that the the implemented solution will always be operational even when major operating system updates occur.

The solution implements both H.323 and SIP protocols. This enables the implemented solution to communicate virtualy with any other video conference solution that also implement these protocols.

It is the only free video conference that can perform dual video (which is required for a video conference system for presentation purposes, as described in chapter 1).

B. Limitations

In this section it will be described the limitations that were need to be made in order to overcome deeper problems.

The choice of an exclusive use of just one operating system, instead of a multi platform solution, was imposed by the use of Flumotion. Since Flumotion is only available for linux systems, it was needed to find another multimedia streaming software for MS Windows. Since the number one objective was to develop an application that could run on linux operating systems (because they are open source and free), it was impossible, due to time constrains, to adapt the solution to be multi platform.

A strong limitation that Flumotion and named pipes impose is the inability to start Flumotion only when a call is coming or being placed. Since Flumotion needs to know the properties of the streams when it is being started and the named pipes must have a reader (which is Flumotion) when being created, due to OPAL limitations, Flumotion needs to be started at the start of the system.

C. Future Work

The implemented solution is not in a finish state. To be fully usable in a real life application, it needs improvement in various areas which will be described below:

- Start the streaming server only when a call is being start (and the media streams are being created).
- Prepare for multiple platform.
- Include support for other protocols.
- Enable the communication to more than one audience rooms.
- Stream the presentation for third party users.
- Record a presentation session in a multimedia video format.

References

[1] L. Haibo and W. Liang, "The application of network video conference system in E-Government," *International Forum on Information Technology and Applications*, 2010.

[2] G. Zhang, M. Hillenbrand, and P. Mller, "Facilitating the interoperability among different VoIP protocols with VoIP web services," *Proceedings of the First International Conference on Distributed Frameworks for Multimedia Applications*, 2005.

[3] H. P. Project, *H.323 Plus*, 2012. [Online]. Available: http://www.h323plus.org/

[4] Aethra, *Aethra AVC 8400 Datasheet*, 2005.

[5] I. T. Union, *ITU-T Recommendation H.261, Video codec for audiovisual services at p x 64 kbit/s*, 1993.

[6] ——, *ITU-T Recommendation H.263, Video coding for low bit rate communication*, 2005.

[7] ——, *ITU-T Recommendation H.264, Advanced video coding for generic audiovisual services*, 2011.

[8] ——, *ITU-T Recommendation H.239, Role management and additional media channels for H.300-series terminals*, 2005.

[9] Tandberg, *Technical Description of TANDBERG 770/880/990 with software version E3*, 2011.

[10] I. T. Union, *ITU-T Recommendation G.711, Pulse code modulation (PCM) of voice frequencies*, 1988.

[11] ——, *ITU-T Recommendation G.722, 7 kHz audio-coding within 64 kbit/s*, 1988.

[12] ——, *ITU-T Recommendation G.722.1, Low-complexity coding at 24 and 32 kbit/s for hands-free operation in systems with low frame loss*, 2005.

[13] ——, *ITU-T Recommendation G.728, Coding of speech at 16 kbit/s using low-delay code excited linear prediction*, 1992.

[14] ISO, *ISO/IEC 14496-3,Information technology – Coding of audio-visual objects – Part 3: Audio*, 2009.

[15] Polycom, *Polycom QDX 6000 Datasheet*, 2011.

[16] Emblaze-VCON, *VCON VPointHD Data Sheet*, 2009.

[17] Skype, *http://www.skype.com/*, 2012.

[18] I. Digium, *www.asterisk.com*, 2010.

[19] A. S. Incorporated, *Adobe Connect Solution Overview: Improve collaboration, complete work faster, and drive better results*, 2010.

[20] Microsoft, *Microsoft Lync Server 2010 Datasheet*, 2010.

[21] F. Delorme *et al.*, "Butt-jointed DBR laser with 15 nm tunability grown in three MOVPE steps," *Electron. Lett.*, vol. 31, no. 15, pp. 1244–1245, 1995.