Cross-device Videoconferencing based on Adaptive Multimedia Streams

Pedro Rodríguez, Alvaro Alonso, Joaquín Salvachúa, Enrique Barra, Javier Cerviño
Departamento de Ingeniería de Sistemas Telemáticos
Universidad Politécnica de Madrid
Avda. Complutense 30, Ciudad Universitaria
28040 Madrid, Spain
{prodriguez, aalonsog, jsalvachua, ebarra, jcervino}@dit.upm.es

ABSTRACT: *The increase in CPU power and screen quality of todays smartphones as well as the availability of high bandwidth wireless networks has enabled high quality mobile videoconferencing never seen before. However, adapting to the variety of devices and network conditions that come as a result is still not a trivial issue. In this paper, we present a multiple participant videoconferencing service that adapts to different kind of devices and access networks while providing an stable communication. By combining network quality detection and the use of a multipoint control unit for video mixing and transcoding, desktop, tablet and mobile clients can participate seamlessly. We also describe the cost in terms of bandwidth and CPU usage of this approach in a variety of scenarios.*

Keywords: Bandwidth, Video Conference, Smartphones, Video Mixing, Wireless Networks, Cross Device

Received: 4 February 2013, Revised 2 March 2013, Accepted 8 March 2013

1. Introduction

Videoconferencing applications have been trying to successfully migrate to mobile devices for more than a decade. Traditionally, the low size and quality of the screens, available bandwidth and low processing power compared to desktop computers were really hard challenges that developers overcame with complex technical solutions that allowed users to communicate but often getting a much lower quality of experience.

Over the last years, the increase in popularity of smartphones has helped usher in a new era where high quality wireless communications are prominent. The more powerful CPUs and higher resolution screens of these devices allow for complex distributed applications that are similar in functionality to those found in desktop computers.

This evolution has minimized and, in some cases, even eliminated the inherent challenges of mobile videoconferencing. Today, there is a wide variety of video and audio communications tools available for mobile platforms such as Skype, Google Hangouts and Apple’s Facetime. However, at the time of writing this paper, none of these applications provided cross-platform videoconference with multiple participants displaying their videos at the same time.

We present a solution for the scenario of multiple participants communicating simultaneously via audio and video using different platforms. That is, all users, regardless of their access device and network, have to be able to participate in equal conditions obtaining a similar experience that is adapted to their network and terminal limitations. We can divide the challenges tackled by this solution in two groups: device fragmentation and network conditions adaptation.

The variety of devices will be addressed by confining them into three categories: cell phones, tablets and desktop computers. Different applications with the same functionality are designed to work with each of the categories, this way we can take advantage of the pros and minimize the cons of every platform. An example of this adaptation is video resolution which is more important in bigger screens where defects in quality are more noticeable.

Adapting to network conditions will be addressed by constantly monitoring the quality of service provided by the access medium. If the conditions degrade, lower quality video will be sent and received while maintaining the communication. We designed an algorithm that decides how to react at any given time considering the input obtained from the network monitoring.

The use of a centralized architecture allows for the transcoding of the videos at the server side, providing adequate video size and quality for each type of devices without placing any of this work in the client.

To sum up, in this paper we propose an architecture that enables an advanced videoconferencing scenario by providing cross-device video and audio communication tools for multiple participants. We have developed a demonstrator following this architecture as part of a research project called VaaS, undertaken between Universidad Politécnica de Madrid (UPM) and Telefónica. We have quantified the server and measured the resulting bandwidths in different scenarios.

In section II we detail the challenges that we want to overcome with the proposed solution. Section III presents the general architecture of the resulting system as well as the mechanisms used to solve the problem, detailing the media transcoding and the adaptation algorithm. Section IV presents the results of the demonstrator measurements in terms of CPU and bandwidth usage seen from the server side. Section V gives an overview of the similar solutions and related work. Finally section VI provides the conclusions as well as the future lines of work.

2. Videoconference Challenges in Wireless Networks

Wireless networks are a very good chance for videoconference, and they have been widely used with this purpose. But they also present important challenges due to their variety and the diversity of devices that use them.

2.1 Access Networks

Devices in different wireless networks can interact and have a videoconference together. So the videoconference system has to support all of them with the varieties that they present.

The wireless networks that are commonly more used nowadays are 3G, HSPA, HSPA+ and Wi Fi through ADSL. All can be used with a smartphone, a tablet or a desktop or laptop computer.

The first characteristic to consider of these networks is the typical bandwidth, the downstream to get the videos and audios from the rest of participants in the conference but also the upstream to send your video and audio. Also the delay, jitter and bandwidth fluctiation have to be considered. As we will explain throughout this paper, to overcome some of the challenges we used Flash technology that forced us to use TCP as the transport protocol.

As seen in [1], high bandwidth communication coupled with delay and random packet loss such as the one encountered in wireless communications, significantly reduces the available throughput. Thus, it is necessary for a videoconferencing solution to be able to adapt to those random changes in throughput in order to be able to allow for fluent real-time communication. A system that fails to adapt correctly will be imposing longer communication delays due to the packet retransmission present in TCP, and eventually the abrupt ending of the transmission. Furthermore, this adaptation has to be continuous as the conditions of a wireless network can change easily so the system has to be able to react quickly enough to sustain the communication.

2.2 Terminal capabilities

There is a huge variety of devices that can use a videoconference system, smartphones, tablets, desktop and laptops, even televisions are starting to incorporate this possibility with the apparition of the smart TVs.

But in comparison smartphones are the ones that have the more limited features when it comes to videoconferencing. So they are the ones that we give special importance in the scenarios and in the tests performed. The device features that limit a videoconferencing system are:

• **CPU:** The device will have to compress its video and audio before sending it and will have to decompress one or several received videos and audios, this can be very CPU intensive. Some devices, altough considered smartphones, have a slow CPU and the system has to work smoothly also in those devices.

• **Battery:** Even if the device CPU is enough but its load is very high the mobile device will consume its battery very quickly and it will warm up resulting in a bad user experience.

• **Memory:** The available RAM for the application is not very high in some devices. However in our experience it does not suppose a problem as the main bottlenecks are CPU and bandwidth.

• **GUI:** The screen size and resolution is very low in mobile devices, even more compared with other devices such as tablets or desktop. This influences the layout, the way the videos are presented and the way the user interacts with the application. The user experience should be the same in all kind of devices.

3. VAAS

Our proposal and implemented prototype to overcome the challenges listed above is called VaaS. VaaS is a centralized videoconferencing application with clients for smartphones, tablets and the web. It is based on Adobe Flash technology on the client side and Java on the server.

The Adobe Flash platform allows easily portable applications via Adobe Air. This allowed us to create a single application for all mobile platforms with minor adjustments while, at the same time, being able to produce a web for desktop version without starting from scratch. Furthermore, we have previous successful experiences with the technology as seen in [2], [3].

The need to transcode videos to adapt to different terminals suggested a centralized server approach. This main drawback of this approach combined with Flash is that the only transport protocol available is TCP. At the time of developing this project, RTMP and its variants where the only way for a Flash application to transmit real time media to a central media server. This determines some of the decisions we will be explaining in this section.

3.1 Video transcoding and mixing

In this solution we use a combination of transcoding and mixing in order to give the best experience in a variety of devices and conditions. In this subsection we will explain the details of the design in this regard, while in the next one we will explain the how does the system choose what videos are to be served to each participant.

In a typical videoconferencing scenario, every participant would send his or her own video and receive one for each of the other participants. In our case, each client would send one video and receive five. That means that if all videos are of the same quality, the needed upload bandwidth equals the video bitrate while five times that throughput is needed in the download. This is acceptable when all participants have similar devices and network quality, as they will all get a good experience according to their capacities. The problem can be even worse if participants choose their own quality as some could choose a very high resolution and quality video that others cannot possibly receive.

Furthermore, decoding several video streams at the same time requires significant CPU power. This is not a problem in desktop computers, however, even todays powerful smartphones have trouble coping with this workload and even if they are able to do it, the hit in battery life is very noticeable. That is why most videoconferencing applications in smartphones display only one video at any given time.

In order to tackle these problems, the central server will use two techniques widely known in real time communications scenarios: video mixing and transcoding.

We define video mixing in this context as combining several video streams into one. It can be done by taking frames from each of the videos and arrange them together in a new frame at the desired rate.

Feature	Upload Desktop.	Upload Smartphone.	LB Mix.	HQ Mix.
FPS	10	10	5	10
Resolution	320×240	320×240	80×60	320×240

Table 1. Video Resolution and Frames Per Second

Video transcoding is to adapt a video to a lower bitrate. By losing spatial and temporal resolution, the video can be transmitted using less bandwidth.

We combine these two techniques generating a video mosaic from all the participants and transcoding it to two bitrates that will be sent to clients depending on the conditions of their access network and the output of the algorithm.

Table 1 displays the chosen configuration for the prototype. Desktop and smartphone clients upload the same quality. Two mixed videos are generated, a high quality and a low bandwidth. As we will see in the next subsection, the adaptation algorithm will switch between the available qualities depending on the network situation.

Due to the solution being Flash-based the video and audio codecs and parameters are limited to those implemented in the Flash Platform. At the time of the development, only Sorenson Spark (a variant of H263) was available in Adobe Air for mobile operative systems. Speex is the codec used for audio.

3.2 QoS Monitoring and adaptation algorithm

This architecture also tackles the problem given in the mobile clients that are connected via wireless networks. The quality of these networks could vary depending on different aspects: distance to the antenna, amount of clients connected at the same time, overall coverage of the Service Provider, etc. Furthermore, VaaS clients send and receive multimedia streams through TCP connections, so that there are not and packet losses between ends, but this decreases the TCP throughput (the bandwidth used by the connection) when there is network congestion or the packets are sent through lossy transmission channels. In this section we will explain how we perform an active monitoring and adaptation of the traffic depending on the network condition.

Mobile VaaS clients automatically change the video quality of sending and receiving streams when they detect network problems. They could first decrease the video quality and increase it later when problems do not persist. But the system always measures the quality network conditions prior to increase or decrease this quality. This quality variation allows the system to use less bandwidth and avoid network congestion that is the most important source of problems in a TCPbased communication or, at least, the only cause that could be avoidable.

Algorithm 1 Algorithm to decrease sending quality video in case of network congestion

```
Require: $currentBW, minBW, maxLatency$
1:  if $latency > maxLatency$ then
2:    if $currentBW > minBW$ then
3:      $currentBW = currentBW = 2$
4:      $changeBW\ (currentBW)$
5:    else if $isSendingVideo$ () then
6:      $stopSendingVideo$ ()
7:    else if $isReceivingHighVideo$ () then
8:      $stopReceivingHighVideo$ ()
9:      $startReceivingLowVideo$ ()
10:   else if $isReceivingLowVideo$ () then
11:     $stopReceivingLowVideo$ ()
12:   else
13:     $disconnect$ ()
14:   end if
15: end if
```

3.2.1 Initial QoS Measurement

The system obtains network parameters at the moment of establishing the connection between client and server. During this phase the user will see a progress bar indicating that the system is measuring QoS parameters. These measurements calculate the latency and estimate the available bandwidth between client and server. Once finished the system configures the video to be sent from client to server to use half the bandwidth measured.

In the rest of the communication the system will follow a pessimist adaptive algorithm, assuming that there will be never better conditions than initial ones. If network congestion occurs during communication the system will take the initial measurements as taken in a stationary state, without congestion. And then the system will try to reach again the initial state by dynamically increasing the video quality when no congestion is detected.

3.2.2 Network problem detection. Decreasing video quality

Algorithm 1 shows how VaaS system adapts when it detects network congestion.

This is the problem that the system avoids by monitoring the network conditions: it will measure the time it takes to a packet travel from end to end (*latency*), and if this time increases above a threshold (*maxLatency*) the system will assume that the network is becoming congested. The system then reacts by decreasing the video quality in both sides of the communication, so that it consumes less bandwidth in both directions. We explain below the steps followed by the algorithm:

1) Every time the system detects network problems it reduces the bandwidth consumed by the client (*currentBW*) to half, decreasing the video quality (lines 2-4). If the consumed bandwidth is lower than a minimum limit (*minBW*) the client stops sending its video (lines 5-6).

2) If the connection does not improve the next step is to reduce the bandwidth consumed by the server in the same connection. In this case, the mobile clients that are subscribed to a mixed video could subscribe to another video with lower quality. This second video consumes less bandwidth (lines 7-9).

3) When problems persist, the clients will stop receiving video from the server. This case will be the same as they would be connected to a audio conference, because they do not send nor receive video streams (lines 10-11). 4) In the worst case, the problems will persist. In this case the system will disconnect the client from the session (lines 12-13).

This algorithm will be executed by the system every decRate seconds. This parameter should not be very low, because the system will decrease the quality very fast, and it is preferable to stabilize the communication on every change rather than detect false positives in the latency.

During this process the user will see different messages in the mobile application, indicating the different states of the connection.

3.2.3 Improved network conditions. Increasing video quality

The system assumes the network condition is favorable when it does not detect high latencies for a period of time. Actually, this new situation could be caused also by additional factors, such as variation in antenna coverage. The current problem is that the system is not able to distinguish if the network traffic is close to the level of congestion or not, so VaaS reacts trying to progressively increase the traffic rate up to its original state. In this case it follows a generate and test approach. In other words, the system increases the quality of the communication after a configurable time interval.

The system will increase the video quality step by step, in the opposite way it follows during the congestion state. In case network congestion is detected it will reduce the quality again. If the system does not detect congestion the upper quality limit is the one measured during the initial phase.

In this case this algorithm will be executed by the system every *incRate* seconds.

We can represent the relationship between the time to adapt to bad conditions and the time to adapt to good conditions as $k = incRate = decRate$. This parameter becomes a trade-off in our adaptive mechanism. Based on the practice k should be equal or higher than 1, because for the system it is more important not to experiment problems than to rapidly adapt to good conditions. But if this parameter is too high the system will react slowly to good network conditions.

4. Demonstrator and Measurements

The deployment of the system is shown in Figure 3 and will be briefly explained in this section. In the next subsections we will go through the tests and measures performed.

It is composed of the VaaS server that contains the basic services of the system, the SMTP server for the email communications and the MIB (Message Integration Broker) server for SMS messages communication.

The three servers are located behind a firewall and only the VaaS server will communicate through TCP port 25 with the SMTP server and through TCP port 9800 with the MIB server. The VaaS server is the only one accessible from outside through ports 80 (for HTTP communications) and port 1935 (for RTMP communications) for all kind of devices.

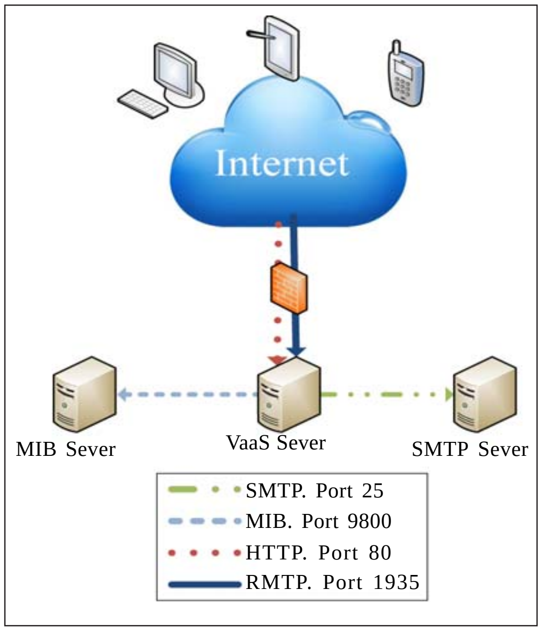


Figure 3. VaaS Architecture

4.1 Experimental set-up

In the experiment participate a total of six client devices, the maximum number that the system allows: three desktop computers, two smartphones and one tablet. In order to make an optimal measurement of the traffic we used devices with different CPU configuration and network interfaces:

• **Desktops:** They present an Intel Core 2 CPU with a capacity of 2.40GHz and 4GB of RAM. The web application runs on Microsoft Windows and Google Chrome. We connected all desktops to the server through a wired LAN.

• **Smartphones:** Nexus S with 1GHz ARM Cortex A9 and 512MB of RAM. We used a native application that runs on Android 3.1, connected through wireless 3G network.

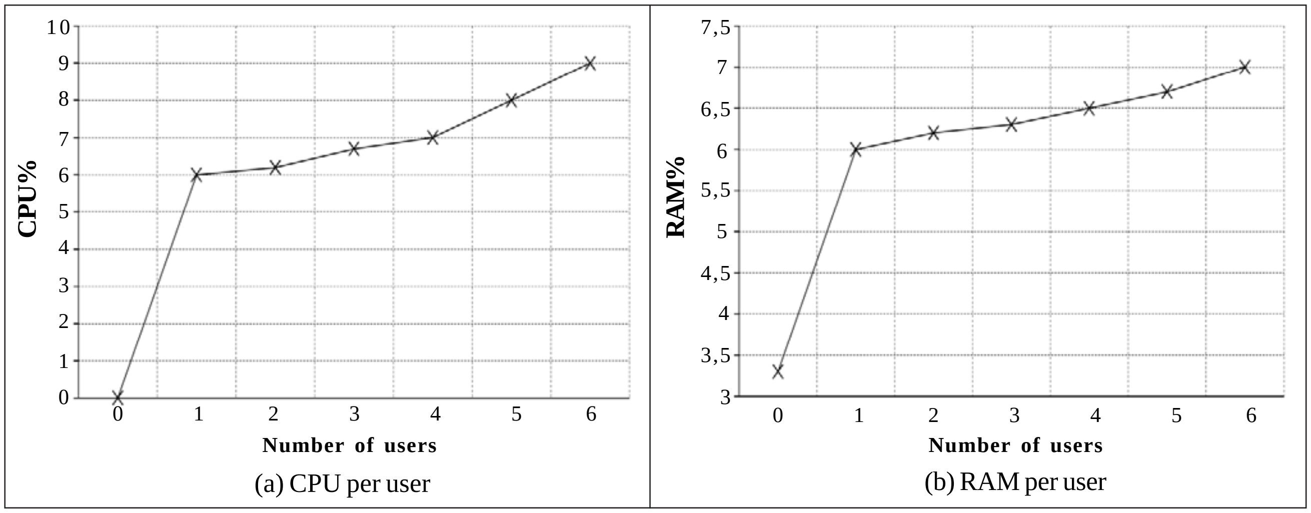
• **Tablet:** Samsung Galaxy Tab 10.1 with 1GHz Tegra 2 and 1GB of RAM. We used a native application that runs on Android 3.1, and we connected the tablet to the system through wireless LAN.

The server runs on a HP Compaq 600 Pro, which has an Intel Core 2 Quad CPU with 2.66GHz and 2GB of RAM.

4.2 Server profiling

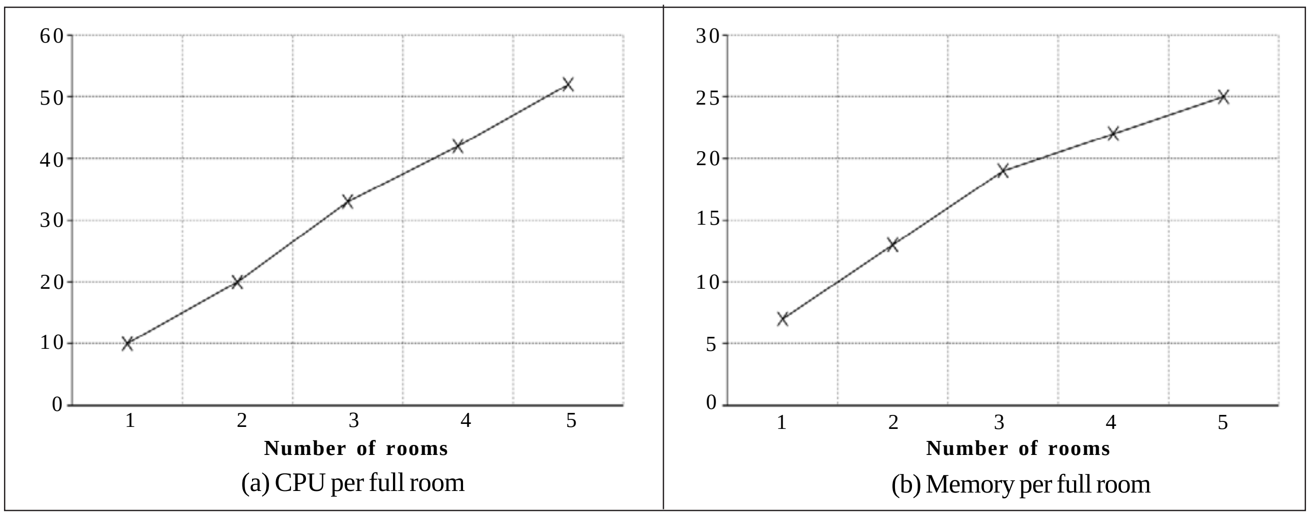
To characterize the server part of the system, we measured CPU and memory consumption in the computer destined to be the server in the demonstrator. The percentages are taken from the total 2 gigabytes of RAM and of only one of the CPU cores.

We can estimate the total capacity for serving videoconference rooms in the server from the data obtained in these experiments. Figure 1(a), 1(b) map the CPU and memory usage of a single room as the number of participants increases. The first user



(a) CPU per user (b) RAM per user

Figure 1. Resources per user in a single room



(a) CPU per full room (b) Memory per full room

Figure 2. Resources per full room

produces a steep jump in resource usage. This is expected as the transcoding is started and the necessary memory structures are allocated. However, as more clients enter the room, the stress over the CPU and memory increases almost linearly.

Figure 2(a), 2(b) show the evolution of CPU and memory consumption with the number of rooms filled with six participants. As we can see, the increase is also linear. The CPU is at 60% when we reach five rooms but, as mentioned above, this measure is over only one core so there is plenty of room to grow. However, memory consumption is almost at a quarter of the total, meaning that, in this case, memory is the bottleneck in the server side.

4.3 Bandwidth measurements

In this experiment we want to analyse the output and input traffic in the server measuring the consumed bandwidth in a scenario with different types of devices connected to the system. These devices will be connecting gradually to the server. In the figures we can observe the captured input (Figure 4) and output (Figure 5) bandwidth in the server. We have measured the total aggregated bandwidth during the entire process and the aggregated bandwidth of each type of device since they are connected to the experiment.

We can obtain some conclusions from the graphics. First we can observe bandwidth consumption peaks when we connect each device. The peaks in seconds 20, 60 and 110 correspond to the connection of the desktop devices, the peaks in seconds 320 and

480 to the connection of the smartphones and the peak in second 610 to the bandwidth measure of the application in the tablet device.

Secondly, observing the total bandwidth capture between seconds 110 and 320 we can check the consumption when the three desktops are connected. Between seconds 320 and 480 the three desktops and a smartphone, between 480 and 610 the desktops and two smartphones and second 610 and the final the complete scenario with all the devices including the last smartphone and the tablet.

Thirdly, in the second line of Figure 4 (desktop bandwidth) we can observe that there is not a perceptible variation of the consumed bandwidth when we connect the smartphones and the tablet to the session. However in the third line (smartphones) there is a little increase of the bandwidth when the tablet connects.

Finally we can observe that, logically, the increase of the number of clients in the session does not affect to the upload bandwidth consumed by the rest of participants because of this only depends on the video stream than each client sends. Also the consumption of the desktops is greater than the smartphones.

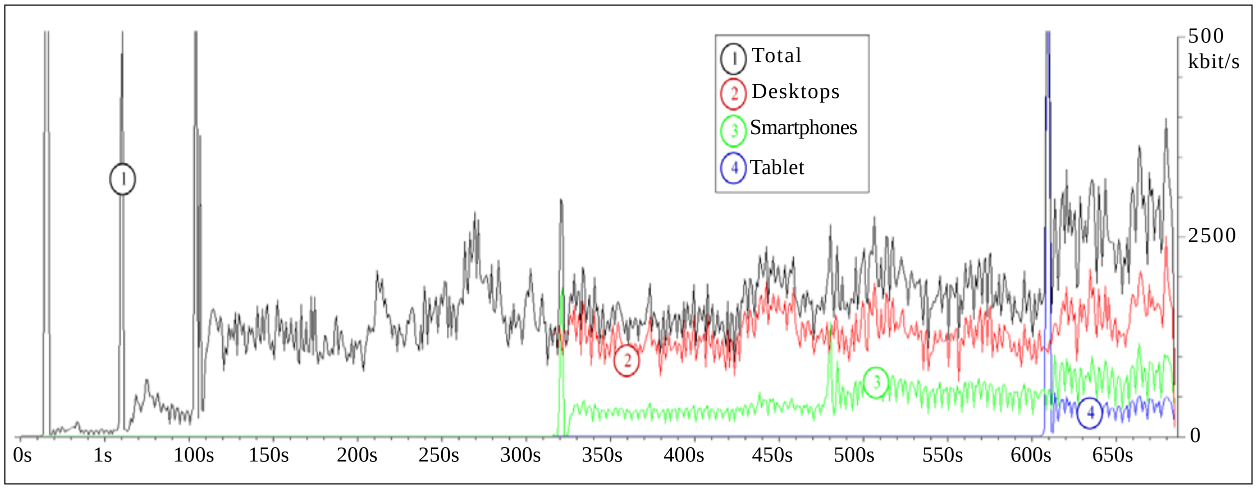


Figure 4. Incoming Bandwidth

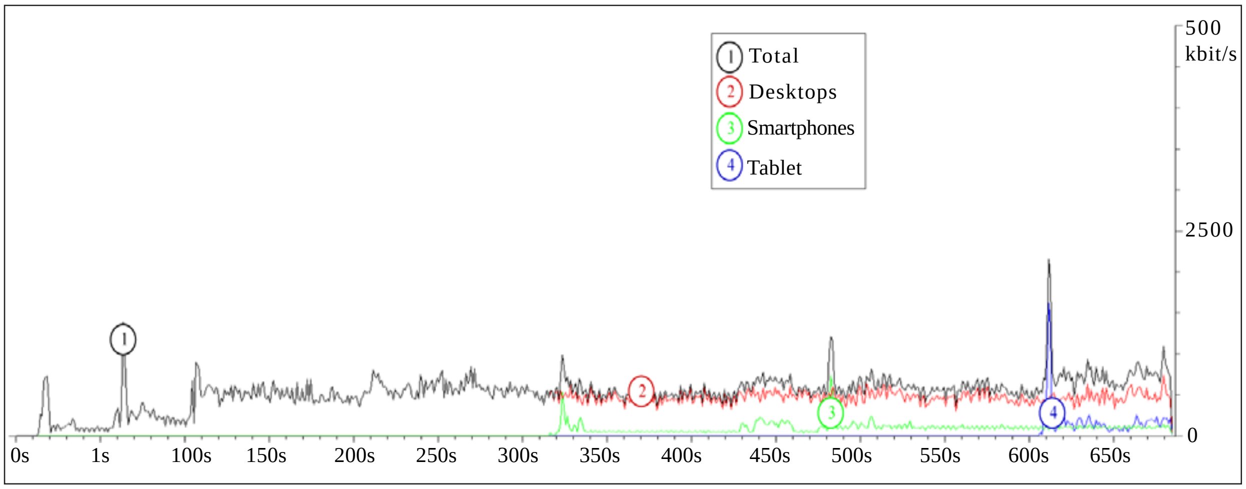


Figure 5. Outgoing Bandwidth

5. Related Work

As mentioned in the introduction the new technical capabilities of tablets and smartphones today are encouraging its integration in multimedia environments and videoconference systems. Many studies analyse different mechanisms to measure the state of the network and manage the multimedia stream exchange between the components in the system.

Another important topic to consider when transmitting multimedia streams through networks with variable bandwidth is the codification used to compress the video streams.

5.1 Network state management
We believe that the measurement of the network conditions must be done the simplest way possible in the sense of not adding complexity to the system. In [4] and [5] the proposal is to add a device near the mobile network that receives the multimedia traffic from the server and adapts it before redirecting it to the clients. Furthermore, this device checks periodically the network state in order to configure the adaptation.

To avoid including extra components in the system, [6] proposes sending signalling packets between clients and server indicating the state of the network and using this information to manage the quality configuration. In this solution no device is added to the environment but the definition of a new protocol is needed. Our solution implies the analysis of the network in the server based only in the traffic received.

A very good approximation is proposed in [7]. In their solution each mobile device monitors some parameters like link quality info, signal strength at receiver, noise level at receiver and number of discarded packets due to different causes. Applying this parameters to different algorithms the device could adapt by:

- Changing the quality of image.
- Increasing or decreasing the frame rate.
- Activating or deactivating the color informat

As said in the study in most cases the client would make the adaptation keeping an acceptable frame rate.

In server-side, the system also manages the traffic quality adapting the media streams that it sends to the clients. The size of the frames received is directly related to the adaptation process, and this adaptation depends on the quality state. Therefore depending on the received frame size the server adapts the traffic sent to the client.

This way of estimating the mobile client state from the server can be imprecise depending of the video codecs used in the client or the different sizes of the emitted videos. However the detection is made using only the multimedia traffic without adding extra components or streams.

Solving all these problems, in our system the detection is made with the TCP latency monitoring. The server measures the latency between packets at the beginning of each connection and during the session. Depending on this latency it adapts the video quality in both directions to the requirements of the network.

5.2 Video codecs
[8] explains how important the video transcoding is in a core network infrastructure to improve interactivity on mobile applications. An intelligent design of the transcoding system can maximize the use of reusable information from the video stream content. At the same time, video transcoding can improve visual experience in lossy communication environments, which are very common in mobile network environments.

Related with video transcoding a very interesting idea, as said in [9], is the SVC (Scalable Video Coding) standard. SVC is an extension of H.264/AVC that allows efficient, standard-based temporal, spatial, and quality scalability of video bit streams. The SVC codifies a high quality video stream that contains a subset of video streams. This is achieved by removing packets from the high quality video stream. Using these subsets the codification cab be adapted to the network conditions reducing the bandwidth.

However this idea can not be applied to our work because of two main reasons. On the one hand the Flash platform does not support scalable video codecs like the proposed by SVC standard. One of the video codecs used in our system is Sorenson

Spark, based on H.263. And even thought the SVC standard has been included in different video coding standards like H.262 MPEG-2 Video, H.263, and MPEG-4, all the efforts with these standardizations produced results with inferior coding efficiency than with H.264/AVC.

And, as explained in [10], even the H.264 codec implies many challenges when using in mobile environments with dynamical changes in the network conditions. These challenges are related with unnecessary retransmissions, the heterogeneity in wireless users and the bandwidth fluctuations.

On the other hand the SVC standard needs to be integrated in existing mobile networks for example by the use of the 3G file format.

6. Conclusion and Future Work

In this paper we have presented a cross-platform videoconferencing solution that aims to provide a consistent communication experience across different platforms, including smartphones. The use of mobile devices implies the use of wireless communications for a highly demanding real-time communication application as multiple participant videoconferencing.

The main challenges to overcome were the differences in terms of processing power and screen size of the terminals and the often-varying conditions of wireless networks.

Firstly, the fragmentation in the device space is overcome by designing the user interaction specifically for each platform while retaining the functionality. Adobe Flash, Adobe Flex and Adobe Air technologies were chosen to this end because of its multi platform nature and the flexibility of the tools.

Furthermore, transcoding and mixing videos in the server side helps smaller and less powerful devices to display several videos simultaneously without delay and improving battery life.

The use of the Flash platform imposes the use of TCP as a transport protocol when communicating with a central server. The use of a reliable protocol with retransmissions coupled with the varying conditions of wireless network can be problematic in real-time communication applications. We designed and implemented an algorithm that constantly monitors the network conditions and can affect the quality of the videos displayed in each terminal separately. Finally, we present the results of our measures in terms of bandwidth and CPU usage.

In the future, the work here can be continued in several ways. First of all, the use of Flash has proven to be effective when bringing communication to different platforms, however, its decrease in popularity and the arrival of HTML5 questions the viability of this solution in the short term.

WebRTC is being defined and developed to offer real-time peer-to-peer communications to the web taking advantage of HTML5 and existent real-time protocols and codecs instead of defining new ones. WebRTC is a joint effort by the WebRTC working group of the World Wide Web Consortium (W3C) and the rtcweb group from the Internet Engineering Task Force (IETF) where the first provide the HTML and JavaScript API and the latter defines the protocols and codecs to be used in the communication. WebRTC will allow for the development of videoconference applications such as the one described in this paper while using standard protocols instead of proprietary ones.

The first step towards implementing our solution in the WebRTC world would be to develop a MCU such as the one described in [11] capable of interconnecting users in a centralized way and being able to transcode and mix the received streams to adjust quality and bandwidth. Later, the QoS adaptation algorithm should be adjusted to the new conditions.

7. Acknowledgments

Authors would like to thank Telef´onica for funding this project and the rest of the project members for their collaboration and support.

References

[1] Lakshman, T., Madhow, U. (1997). The performance of tcp/ip for networks with high bandwidth-delay products and random loss, *Networking*, *IEEE/ACM Transactions on*, 5 (3) 336 –350, June.

[2] Rodriguez, P., Gallego, D., Cervino, J., Escribano, F., Quemada, J., Salvachua, J. (2009). Vaas: Videoconference as a service, *In*: Collaborative Computing: *Networking, Applications and Worksharing*. CollaborateCom 2009. 5th International Conference on, nov.

[3] no Arriba, J. C., Pérez, P. R., Rodrýguez, J. S., Toribio, G. H. F., Cantero, F. E. (2008). Marte 3.0: Una videoconferencia 2.0, *In*: Libro de Ponencias de la VII Jornadas de Ingenierýa Telematica, p. 209– 216.

[4] Hokimoto, A., Nakajima, T. (1996). Handling continuous media in mobile computing environment, *In*: 6th International Workshop on Network and Operating Systems Support for Digital Audio and Video, p. 175–182.

[5] Hokimoto, A., Kurihara, K., Nakajima, T. (1996). An approach for constructing mobile applications using service proxies, in Distributed Computing Systems, *In*: Proceedings of the 16th International Conference on, may, p. 726–733.

[6] Ruiz, P. M., Garcia, E. (2003). Adaptive multimedia applications to improve user-perceived qos in multihop wireless ad-hoc networks, in Multihop Wireless Ad-Hoc Networks, *In*: Proc. PIRMC 2002, p. 1467–1471.

[7] Esteban Ines, D., Fujikawa, K., Kawai, E., Sunahara, H. (2009). Streaming mobile multimedia optimization for video-conferencing scenarios, *In*: Advances in Multimedia. MMEDIA '09. First International Conference on, july, p. 80–85.

[8] Shen, B., Tan, W.-T., Huve, F. (2008). Dynamic video transcoding in mobile environments, MultiMedia, *IEEE*, 15 (1) 42–51, jan.-march.

[9] Schierl, T., Stockhammer, T., Wiegand, T. (2007). Mobile video transmission using scalable video coding, Circuits and Systems for Video Technology, *IEEE Transactions on*, 17 (9) 1204 –1217, sept.

[10] Hsiao, Y. -M., Lee, J. -F., Chen, J.-S., Chu, Y. -S. (2011). H.264 video transmissions over wireless networks: Challenges and solutions, *Computer Communications*, 34 (14) 1661 – 1672.

[11] Rodrýguez, P. J., Cervino, Trajkovska, I., Salvachua, J. (2012). Advanced topics in videoconferencing based on webrtc, *In*: 2012 IADIS Collaborative Technologies. Lisbon, Portugal: *IADIS*.