Deploying a Multipoint Control Unit in the Cloud: Opportunities and Challenges

Álvaro Alonso, Pedro Rodríguez, Joaquín Salvachúa, Javier Cerviño
Departamento de Ingeniería de Sistemas Telemáticos
Universidad Politécnica de Madrid
Madrid, Spain
{aalonsog, prodriguez, jsalvachua, jcervino}@dit.upm.es

***Abstract*—A Multipoint Control Unit (MCU) is a software component that manages different aspects of multimedia systems: mixing, forwarding, recording or transcoding media streams. This paper shows how Cloud infrastructures offer new opportunities to MCUs in a range of scenarios, scaling to a variable number of users. However, this deployment also implies some important challenges that need to be solved, considering the MCU functionalities and the common scenarios in which it will be used. These challenges are related to up and down scalability, geographic distribution of the users and the MCU system profiling. We provide an overview of the most effective solutions to face them and the characterization of a previously developed MCU in two videoconference scenarios. A Cloud-based MCU provides important advantages to take into account and the challenges we detected are already solved in similar environments making its deployment a promising research area.**

***Keywords*—Cloud Computing; MCU; multimedia.**

I. INTRODUCTION

Multimedia systems have gained a relevant role within software applications and services in the Internet over the recent years. Thus, we daily use multimedia applications, like video streaming or video recording. Some of these applications have strong real time requirements, such as videoconference or multiplayer online games.

In this type of applications we need to interconnect two or more users that will exchange some resources like video, audio or data. Moreover, this exchange can be made in real time. Frequently, and specially when there are more than two users, is necessary an intermediate device that manages the communication between the users and the exchange of the resources. The name of this component is Multipoint Control Unit (MCU) and its function is to coordinate the distribution of audio, video, and data streams amongst the multiple participants in a multimedia session [1].

Due to MCU's characteristics, it is possible to convert a mesh topology of connection in a star topology. This way the MCU acts as a central device forwarding the multimedia streams among the participants in the session. However, it can make some additional task that frequently reduces the compute requirements of the devices in the final user. Also it adds interesting features due to the fact of all data is going to go through the MCU, allowing several operations that can provide advanced services often requested in multiconferencing and collaborative multimedia applications:

- *Broadcasting*: this is the basic operation of the MCU, by which it sends a stream from a publisher to multiple subscribers. These subscribers receive this stream once and the publisher only sends it once to the MCU, saving bandwidth in its network interface at the expense of the MCU, which has usually better network performance.
- *Transcoding*: the use of a more advanced MCU able to mix and transcode media streams can pave the way to solving the heterogeneity of devices and access networks. By transcoding streams into different bitrates and sizes, the communication can be adapted to diverse network conditions and screen sizes optimizing the use of network and CPU in the clients at the expense of the MCU. This is also useful in a gateway scenario where media streams have to be translated.
- *Composing*: by generating a single video or audio stream from the available inputs, the MCU can reduce the amount of CPU overhead and control needed to participate in a multiuser multimedia system when needed.
- *Recording*: the MCU is receiving all the streams present in the session and, as stated in the previous point, is able to generate a composed stream by combining them. If a recording of the session is required, the MCU can store that stream for future reproductions.

All these features normally require a high computation level in the device where the MCU is running. The computer usually needs high level of memory and CPU power. However, these capabilities may change dynamically with the variations in the number of users or the different scenarios of the applications. The requirements of this type of devices blend very well with the Cloud Computing model because, according to the NIST definition [2], it provides characteristics like *On-demand self-service*, *Broad network access*, *Resource pooling*, *Rapid elasticity* and *Measured service*.

In the next section we analyse the opportunities and advantages that, according to these characteristics, the deployment of an MCU in the cloud offers. However, it implies important challenges that we describe in Section III, presenting also the most effective solutions to them. Finally, Section IV describes the conclusions as well as the future lines of work.

II. OPPORTUNITIES

In this section we will review the main advantages of running an MCU on Cloud Computing systems. An MCU component may require different computing characteristics depending on the number of participants, session conditions

 ISBN: 978-1-61208-271-4

(recording, forwarding, transcoding or composing), and the physical location of the participants.

Furthermore, these conditions of operability may vary dynamically during the session. Thus, we will demonstrate the benefits of deploying the MCU in a cloud scenario, where the session can be adapted easily and dynamically to variations on this type of conditions according to the particular requirements in each moment.

The cloud model defined by NIST and its essential characteristics illustrates these advantages and help us to better understand them:

- *On-demand self-service*: Users can provision computing capabilities (CPU, network, storage, etc.) as needed.
- *Broad network access*: Those capabilities are available over the network in different locations and are served through standard mechanisms.
- *Resource pooling*: The cloud follows a multi-tenant model, assigning resources to different users.
- *Rapid elasticity*: Capabilities can be provisioned and released automatically to scale to user demand.
- *Measured service*: Resources are automatically controlled, monitored and reported by metering systems.

Below we explain how these features provide new opportunites to MCU-based communications in these scenarios.

A. Scale to user demand

Multimedia systems offer their users the possibility of joining a conference before and during the session. They could also leave the session while it is running. Depending on the type of session this variability could be high.

A high number of users usually means more bandwidth, memory and CPU consumption. In other words, an MCU would demand more capabilities from its computing infrastructure.

In a traditional environment the provider should previously provision its own physical machines to tackle with the high peaks of demand. However, this solution implies more idle resources when the user demand is low.

In a cloud environment the multimedia provider could dynamically provision and release virtual machines on demand. This is usually done by turning on and off those virtual machines depending on the resources needed, according to the participants in the session.

This could also be achieved by dynamically increasing the performance of virtual machines. We could, for example, increase the CPU and memory capacity of a running virtual machine. We could also improve the network performance of these machines by changing their size. For example, Amazon EC2 [3] offers different network performance depending on the size of its virtual machines.

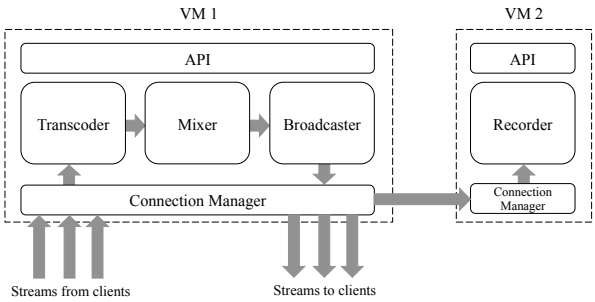


Fig. 1: Example of two MCUs performing different tasks.

B. Scale to scenario requirements

MCU operation also depends on the type of session it runs, and it could perform a variety of tasks: forwarding, recording, mixing (composing) and transcoding. Each of these features requires different computation capacities.

A basic MCU device only forwards streams from one participant to others and requires low levels of computation. However, the required level of memory and CPU increases considerably if the MCU performs the other advanced tasks. These additional features may change during a session depending on different factors: number of users, size of available and generated videos, codecs, etc.

For example, a high number of users usually forces the MCU to compose a single video from the others. Besides, in scenarios where clients connect from different type of devices, the MCU will transcode video and audio to adapt to their different CPU and bandwidth requirements. Finally, the MCU could record the entire session or part of it, including all individual videos, a subset of them, or a composed video.

Virtualized environments of cloud systems help the MCU to adapt to the varying requirements of such features. As in the previous case, we could turn on a new machine when more CPU is needed and later turn it off when this need decreases. Moreover, we could vary the capabilities of a specific virtual machine on the fly, by increasing or decreasing its memory, CPU, number of cores, etc. This would allow our MCU to adapt faster to variations on the scenario requirements.

Another workaround offered by the cloud is to configure different types of virtual machines depending on the features that they will perform. In the example in Figure 1 a machine responsible for broadcasting flows will consume a lot of CPU and memory. On the other hand, a machine responsible for recording a videoconference session consumes low memory and CPU if it receives a single flow with the whole composition of the session.

Summarizing, a cloud-hosted MCU component could easily and dynamically manage the configuration of different types of machines, adapting it to all scenarios. Thus, in a cloud environment we could provide an adaptive multimedia service, which efficiently uses the available resources, reducing costs while improving overall performance in every scenario.

 ISBN: 978-1-61208-271-4

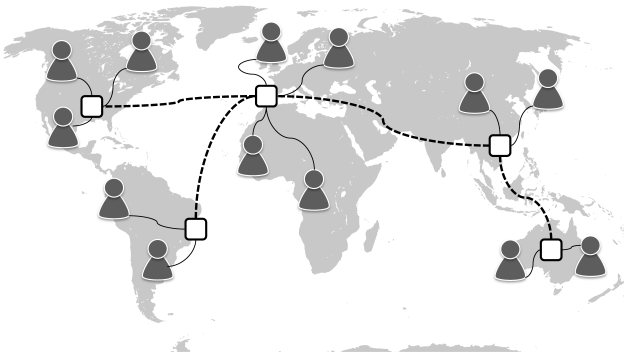


Fig. 2: Example of MCUs interconnection in a global multimedia session.

C. Geographic flexibility

Another critical factor in real time applications that directly affects user experience is the latency of packets that travel between peers. This latency usually depends on the geographical distance between them, and we should reduce it to achieve the lowest possible latency. In multimedia scenarios all streams are sent or received from the MCU and the geographic location becomes crucial for decreasing latency.

Thanks to a cloud-based system we can run MCU devices in different geographic locations, connecting each one with the users that are using the service in each region. For example, at the time of writing this paper, Amazon EC2 provides data centers in North Virginia, Oregon, North California, Ireland, Singapore, Tokyo, Sydney and São Paulo, while Rackspace [4] owns data centers at Texas, Illinois, Vancouver, Hong Kong, London and Slough, UK.

These cloud providers also allow to interconnect MCUs in different regions, so we could even offer sessions around the world, by connecting users to the closest MCUs and interconnecting all the MCUs in the same session. An example of this scenario is shown in Figure 2.

III. Challenges

As seen in the previous section, the deployment of a software-based MCU in the cloud can bring several key advantages to multimedia service providers. The increased efficiency in terms of hardware requirements together with the flexibility in terms of geography and the possibility to adapt the solution to different scenarios encourages the move to the cloud. However, to make the most of the cloud and make the most of its advantages its important to design the system accordingly and take into account the target of the deployment.

Furthermore, we will propose strategies to scale up and down in the cloud that differ from more general approaches such as the one seen in [5].

This section analyses the challenges posed by the optimal adaptation of an MCU to the cloud.

A. Characterizing the system

Characterizing the MCU's performance is the first step towards its efficient deployment in the cloud. Depending on the task (recording, transcoding...) to be performed by a given MCU, the hardware and bandwidth requirements vary significantly. By measuring the performance in a known environment, we can approximate the tier of the instance or the amount of CPU, RAM and bandwidth that is going to be needed when deploying in the cloud. For instance, transcoding needs considerably more CPU power than just forwarding packets. In order to optimise the deployment, we have to quantify this type of characteristics.

Once a complete characterization of the system is made, it is interesting to find correlations between the pure technical resources and the more high-level, application based ones. For example, in a web environment this would mean assessing the increase of CPU usage for each concurrent request of a given type. In the videoconferencing world, the number concurrent users is the typical unit that shows the capabilities when it comes to capacity of a system. Furthermore, we can group these users in different conferences that coexist in the same MCU. We will call this conferences 'rooms'. The number of users in each room is usually limited to a fixed number in videoconferencing systems.

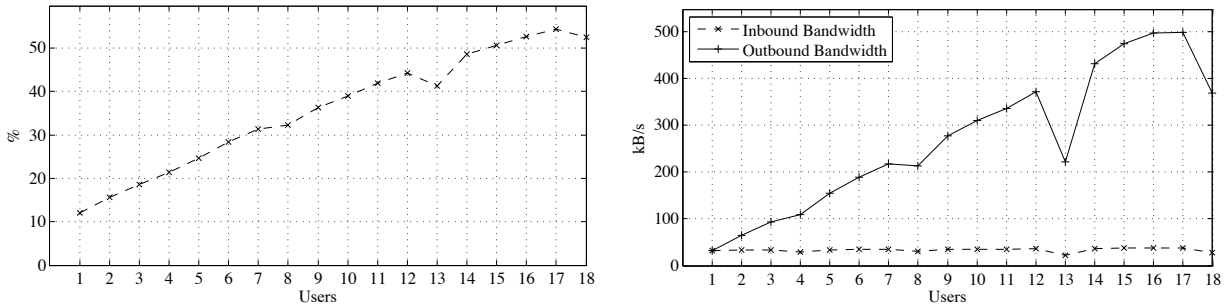
Finding a correlation between the hardware resources needed and the number of users and rooms in a system can simplify the work we are going to do in the next subsections, scaling the system up and down. Knowing the incidence of each new user combined with continuously monitoring the resources consumed by a deployed instance we can react effectively to changes in demand. Of course, this implies measuring the incidence of each user in the cloud instances.

However, there is still a further challenge imposed by the deployment of a known system in the cloud. There is plenty of literature [6][7] on the possible interferences between different virtual machines running on the same host as well as possible ways to characterize the problem [8]. For the purpose of this paper we will assume that the deviation caused by these interferences will not be big enough to invalidate the per-user estimations.

As an example of this type of characterization we will show and comment a real case of MCU deployed in Amazon EC2. We have used an MCU designed by us for WebRTC [9] compatible systems [10][11]. For the characterization we have designed two scenarios that are the most common in videoconference systems: a real time video streaming and a multiuser videoconference. On both systems we have monitored the CPU and memory usage and the bandwidth (incoming and outgoing) consumption in the MCU computer during the experiment.

In the first scenario, live streaming, one of the clients is publishing its media stream (audio and video) in the session and subscriber clients are gradually added to view the published stream. As we can see in Figure 3 the CPU usage in the MCU increases linearly with the increase of the number of users subscribed to the streaming. This occurs because WebRTC standard uses SRTP [12] for the packet transmission implying that the MCU has to unprotect and protect each packet in order to make the retransmission from the publisher to each subscriber. We can observe that the inbound bandwidth consumption is constant during all the session and the outbound increases linearly because of for each new client connected is necessary to make a retransmission

 ISBN: 978-1-61208-271-4



(a) CPU Usage

(b) Bandwidth Consumption

Fig. 3: Use of resources in the MCU when the number of users increases in a live streaming

more. About the memory used by the MCU it increases also linearly but with a minimum variation during the experiment (just a ten of MBytes). Finally note that in sometimes during the test a descent is registered in the results (when user number 8 and 13 connects). We can notice that this anomaly is associated with a decrease of the inbound traffic so we can deduce that it is doubt to a small bug of performance in the publishing client computer, which is also running in a Amazon EC2 virtual machine.

In the second scenario, multi user videoconference, each user that connects to the service publishes its media stream and subscribes to all the streams published previously. We have established a limit of 6 users because this is usually the maximum number of users in a standard videoconference room. In Figure 3 we can observe that in this case the inbound bandwidth consumption increases linearly with the increase of users in the room doubt to the fact of each new user publish its own stream to the room. However, the outbound bandwidth and the CPU usage increase exponentially because of for each new user the MCU has to forward the new stream to all the rest of users. Therefore the number of outgoing flows increases following the equation $N = n(n-1)$, where n is the number of users in the room. The memory usage also varies exponentially but like in the first case the variation is irrelevant.

B. Scaling up

When the currently provisioned resources reach their limit, we should able to take advantage of the cloud to keep providing service as seamlessly as possible for the users. In order to do that there are two main types of scalability: horizontal and vertical.

To scale horizontally means to add new servers to the existing pool of resources while scaling vertically is to upgrade the already running servers on the fly.

When it comes to an MCU, both methods have its uses. If the new resources are required to make some additional task in the session like, for example, recording a videoconference talk, the horizontal scalability may be a better solution. However, if the resources are needed because of an increase of the number of users in a determined session the easiest solution may be to add more capabilities to the same computer already managing that session. However, it has to be kept in mind that vertical scalability is not present in all cloud platforms and not all operative systems allow for it.

With the exception of some very specific cases that we will explain later, the fact of have all the participants in a session in the same MCU implies facilities in the forwarding and composing of the media streams. As discussed in the next subsection this is especially critical if a scale down is necessary during a session.

So an important challenge in the case of scaling up is the decision of which type of scalability is better to choose when an increase of the resources in the MCU module is needed.

Both types of scaling involve a latency caused by, either starting a new machine or modifying the existing one. In order to have a satisfactory user experience, this has to be taken into account so no interruptions take place in the communications. This problem can be avoided by anticipating the rise in demand whenever it is possible and react accordingly.

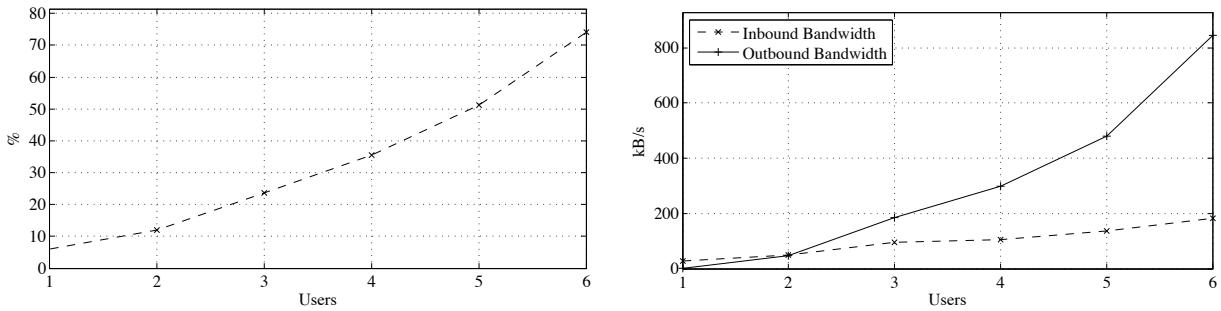
A first approach is to use algorithms to, based on a monitoring of the system, calculate when it is in a limit situation and this way anticipate the necessity for resources starting new machines or adding more capabilities to the existing. The MCU must monitor at all times the state of the system by the analysis of the different factors studied above. If we have been able to establish the limits of the system and the correlation with the number of users and rooms it should be quite straightforward to react whenever the deployed system is reaching its peak.

The result would be a set of thresholds that would define when to add more processing power to the system. This is similar to setting elasticity rules that define the system scalability, an example of this type of approach can be seen in [13].

We can go one step further by use predictive models to anticipate the changes in the requirements of the MCU based on the analysis of previous data. With these type of models we can analyse behaviour patterns of the system to predict the activity that will be in a determined moment. These patterns may be obtained in two main ways. The most effective way is to obtain it from the previous behaviour of the own system. However, if it is not possible we can use the patterns from the behaviour of similar systems.

A good starting point is [14] where this problem is ad-

 ISBN: 978-1-61208-271-4



(a) CPU Usage (b) Bandwidth Consumption

Fig. 4: Use of resources in the MCU when the number of users increases in a videoconference

dressed by an algorithm that predicts resource usage by using pattern matching.

C. Scaling down

In the same way that during a session may be necessary to increase the available resources, it may also occur that at any given time, the demand peak has ended and we have provisioned more resources than needed. As discussed in previous subsection, there are different ways to increase the computation level of an MCU module. In scale down case we can make the reverse operations to reduce the resources dedicated to the MCU.

Therefore the scaling down presents a similar challenge than scaling up. When we detect that the demand has gone down and we have allocated more resources than necessary we must select the closer to optimal way to reduce them. We can reduce the capabilities of the running computers or shut down one of more of them.

But moreover in this case the second option presents additional difficulties. It must be taken into account that the computer that we are going to turn off most probably is performing tasks that we must distribute between the rest of computers. Such redistribution is not a trivial issue.

A client participating in a session is sending and receiving several multimedia streams to and from an MCU. If this computer is going to be shut down, it is necessary to forward the traffic and it should be done in a transparent way to the user. Moreover, to optimise the use of the resources, a full redistribution of the clients and the rooms on the system may be in order.

A possible solution to this problem, shown in Figure 5, is to include a proxy between the clients and the MCU module. When a machine is going to shut down the proxy begins to duplicate the streams between the old and new MCU. When all the streams are prepared the proxy changes the sending and receiving to the client from the old MCU to the new and in this moment the old MCU can be turned off.

D. Geographic distribution

With the flexibility provided by geographically distributed cloud providers comes the challenge of optimally placing the MCU instances in order to get the best service as possible. While the decision might be trivial when all users are located in the same continent or cloud provider's zone, deciding how to react when users are located in distant places can greatly determine the quality of the session.

When making the decision must be taking into account the number of users in each geographical region but also the quality of their connections. To characterize the links between different regions is interesting to do measures of bandwidth, jitter, packet loss or Round Trip Time (RTT). By testing the connection of each user to the different regions of the cloud we can decide where it will perform better.

As seen in [15] the network connections between Amazon instances in different regions perform better than the average internet connection. We should also take this into account when designing and deploying the system.

IV. Conclusions and future work

In this paper we have analysed the main opportunities that the deployment of an MCU component offers in a cloud-based infrastructure. As we have been seen, this alternative provides interesting advantages to multimedia and real time systems with high or variable number of users because these systems usually work in scenarios in which flexibility and scalability are required. However, the deployment is not an easy task and its performance presents also important challenges that need to be solved.

We have also discussed some possible alternatives to face these challenges. The first step is to characterize the system in order to establish relationships between the number of users and the task that the MCU will realize with the technical requirements of the computers. We have presented an example of these measures in two videoconference scenarios and an overview of the existing solutions to the challenges that the scalability (up and down) presents and the geographic distribution of the MCUs.

The conclusion of our work is that the Cloud provides important advantages and that the challenges we detected are already solved in similar environments, so the deployment of MCUs in the Cloud is a promising research area. Our future work is to further analyse these solutions in multimedia scenarios and apply them to real services.

 ISBN: 978-1-61208-271-4

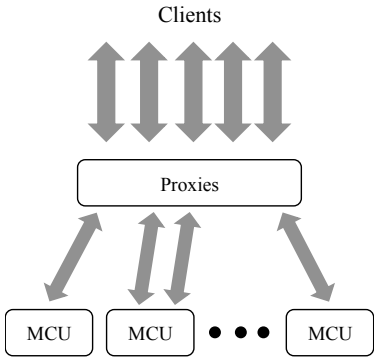


Fig. 5: Proxies forwarding traffic to MCUs.

Finally, we will apply prediction models and algorithms to our open source webRTC project, named Lynckia [16], in order to research the best way to achieve a scalable and flexible real time communication provider. We will also study the performance of the media proxy that will manage the forwarding of media streams when the systems need to scale down.

REFERENCES

[1] M. Willebeek-LeMair, D. Kandlur, and Z.-Y. Shae, "On multipoint control units for videoconferencing," in Local Computer Networks, 1994. Proceedings., 19th Conference, 1994, pp. 356 – 364.

[2] P. Mell and T. Grance, "The NIST Definition of Cloud Computing," http://csrc.nist.gov/publications/nistpubs/800-145/SP800-145.pdf [retrieved: March, 2013], 2009.

[3] Amazon AWS. http://aws.amazon.com [retrieved: March, 2013].

[4] Rackspace. http://www.rackspace.com [retrieved: March, 2013].

[5] L. M. Vaquero, L. Rodero-Merino, and R. Buyya, "Dynamically scaling applications in the cloud," SIGCOMM Comput. Commun. Rev. vol 41, num 1, January 2011, pp. 45 –52.

[6] O. Tickoo, R. Iyer, R. Illikkal, and D. Newell, "Modeling virtual machine performance: challenges and approaches," SIGMETRICS Perform. Eval. Rev. January, 2010, vol. 37, no. 3, pp. 55 – 60.

[7] Y. Koh, R. Knauerhase, P. Brett, M. Bowman, Z. Wen, and C. Pu, "An analysis of performance interference effects in virtual environments," in Performance Analysis of Systems Software, 2007. ISPASS 2007. IEEE International Symposium, April 2007.

[8] A. V. Do, J. Chen, C. Wang, Y. C. Lee, A. Zomaya, and B. B. Zhou, "Profiling applications for virtual machine placement in clouds," in Cloud Computing (CLOUD), 2011 IEEE International Conference on, July 2011, pp. 660 –667.

[9] A. Bergkvist, D. C. Burnett, C. Jennings, and A. Narayanan, "Webrtc 1.0: Real-time communication between browsers," W3C," Working Draft WD, August 2012, http://www.w3.org/TR/webrtc/ [retrieved: March, 2013].

[10] S. Loreto and S. Romano, "Real-time communications in the web: Issues, achievements, and ongoing standardization efforts, september-october, 2012," Internet Computing, IEEE, vol. 16, no. 5, pp. 68 –73.

[11] P. Rodríguez, J. Cervino, I. Trajkovska, and J. Salvachúa, "Advanced videoconferencing services based on webrtc," IADIS International Conferences Web Based Communities and Social Media 2012 and Collaborative Technologies 2012, pp. 180–184.

[12] M. Baugher, D. McGrew, M. Naslund, E. Carrara, and K. Norrman, "The secure real-time transport protocol (srtp)," Internet Engineering Task Force, March 2004, updated by RFC 5506.

[13] F. Galán, A. Sampaio, L. Rodero-Merino, I. Loy, V. Gil, and L. M. Vaquero, "Service specification in cloud environments based on extensions to open standards," in Proceedings of the Fourth International ICST Conference on COMmunication System softWAre and middlewaRE, ser. COMSWARE '09, New York, NY, USA, 2009, pp. 19:1–19:12.

[14] E. Caron, F. Desprez, and A. Muresan, "Forecasting for grid and cloud computing on-demand resources based on pattern matching," in Cloud Computing Technology and Science (CloudCom), 2010 IEEE Second International Conference, December 2010, pp. 456 – 463.

[15] J. Cervino, P. Rodriguez, I. Trajkovska, A. Mozo, and J. Salvachua, "Testing a cloud provider network for hybrid p2p and cloud streaming architectures," in Cloud Computing (CLOUD), 2011 IEEE International Conference, July 2011, pp. 356 –363.

[16] Lynckia. Open Source WebRTC Communications Platform. http://www.lynckia.com [retrieved: March, 2013].

 ISBN: 978-1-61208-271-4