UNIVERSITAT POLITÈCNICA DE CATALUNYA
BARCELONATECH
Escola Tècnica Superior d'Enginyeria
de Telecomunicació de Barcelona

Implementation and Analysis of Low Latency Videoconferencing through Edge Cloud computing

A Master's Thesis submitted by Mauro Lucchini Depompa

to the Faculty of the Escola Tècnica d'Enginyeria de Telecomunicació de Barcelona Universitat Politècnica de Catalunya

London, January 2018

Title of the thesis: Implementation and Analysis of Low Latency Videoconferencing through Edge Cloud computing

Author: Mauro Lucchini Depompa

Advisors: Toktam Mahmoodi and Josep Solé Pareta

Abstract:

Edge cloud computing seems to be a key enabler of 5G networks which essentially brings the servers as close to the users as possible. Among all the benefits this tendency can provide, this master thesis focuses on the advantages in terms of reduction of the latency. First of all, an Edge network model that combines this paradigm with Software-defined Networks (SDN) is presented so as to provide an example of a potential production scenario. Then, a videoconference application is chosen as a particular case study of latency-sensitive and bandwidth exhaustive application and the traffic that it generates is inspected. Thanks to this analysis, a methodology to compute the latency can be proposed which is used during the test runs afterwards. Lastly, a testbed analogous to the model previously presented showcases the benefits of this approach. The results prove the improvement in the quality of the videoconference by means of a noticeable reduction of the latency when the servers are on the edge. Moreover, it is demonstrated the feasibility of providing a dynamic environment where the server can be live migrated. For the sake of providing a complete quality overview, the impact of the available bandwidth and packet loss is evaluated as well.

Keywords: *Edge cloud computing, SDN, videoconference services, latency-sensitive applications, bandwidth exhaustive applications*

I need to drop some lines to my family:

I would like to thank them for the unconditional support I have always received at home to achieve my dreams. Without their backing, I would not be who I am. Among them, I offer my special thanks to Bibi. Thanks for being so close when you were so far. Your visits and your love encouraged me to successfully conclude this chapter of my life.

To my family and my love Bibi, for always standing by me. I love you all.

Acknowledgements

I wish to show my greatest appreciation to Toktam Mahmoodi for giving me the opportunity of coming to King's College London and for the supervision of this project. It has been an honour working with her and learn about her professional methodology and experience. I also owe my deepest gratitude to Luis Sequeira from whom I received generous support along the whole development of the project. I am particularly grateful for his assistance and continuous feedback. Thanks also to Marco Rocchetto, from Konica Minolta, for his insightful comments and suggestions. Finally, I would like to thank Josep Solé too, particularly, for being so helpful to make my stay in London possible.

Scientific contributions

The results presented in this thesis have been used to produce a paper submitted to IEEE/ACM Transactions on Networking.

Mauro Lucchini, Luis Sequeira, Member IEEE, Marcho Rocchetto, and Toktam Mahmoodi, Senior Member, IEEE, "Experimental Analysis of Edge Cloud for Low Latency Networking", IEEE/ACM Transactions on Networking*, January 2017*

Revision history and approval record

Revision	Date	Purpose
0	27/11/2017	Document creation
1	13/12/2017	Document revision
2	05/01/2018	Document revision
3	12/01/2018	Document revision
4	19/01/2018	Document revision
5	24/01/2018	Document final revision

Written by:		Reviewed and approved by:	
Date	24/01/2018	Date	24/01/2018
Name	Mauro Lucchini	Name	Toktam Mahmoodi
Position	Project Author	Position	Advisor

Table of contents

List of Figures

List of Tables

1. Introduction

Edge cloud computing addresses many challenges that enterprise IT faces when running data-centric workloads in the cloud. To begin with, it enables IT to retain sensitive data on-premises while still taking advantage of the elasticity offered by the public cloud. It also reduces the amount of data that flows back and forth between the enterprise headquarters and the cloud. Aside, it reduces the latency involved in dealing with public cloud platforms. This last statement motivates the study presented in this project. Bringing the cloud to the edge seems to be an adequate solution for bandwidth exhaustive and latency-sensitive applications. For this reason, this project considers the case study of videoconferencing services, which meets these two characteristics, and it tries to tackle this challenge with the particular use of Edge cloud computing and SDN.

Videoconferencing is a very interesting case study for examining the potential of Edge cloud computing. It is a very well known application, bandwidth exhaustive and sensitive in terms of delay. Moreover, with the presence of organisations globally and increasing international collaborations, videoconferencing is one of the most common applications running in today's business offices. Business cyclical shifts and economic downturns tend to see a rise in the use of videoconferencing facilities. Used judiciously and strategically, videoconferencing can enhance organizational productivity, engage employees at all levels and reduce travel budgets. Nowadays, real-time services aim to provide more immersive experiences by means of higher resolution videos, three-dimensional audio, or even Augmented reality (AR). With traditional videoconferencing, participants only look into a camera and onto their screens to see one another. This often leads to missed eye contact and a continued feeling of separation. With technology like *HoloLens* [1], participants can turn their head to the left to look at the hologram of the person sitting on their left and turn to their right to interact with the person on their right. In addition, they can share projects and manipulate them in real-time. Nevertheless, one of the main barriers often is not the existence of such high fidelity audio and video systems, but the capacity of in-building networks. Immersiveness of the advance videoconferencing sessions also depends on the latency of the sessions and increased data rate has an immediate inverse impact on the latency. Therefore, there is such an intriguing challenge of offering a network architecture that provides a reliable infrastructure to deploy this kind of services and overcome the latency problem.

The scenario under study consists of a number of individuals either in a large office building or a campus, joining the same videoconferencing session. The main and unique requirement is that the network runs under the same management. In other words, the network topology and configuration within the building or campus is completely adjustable to the needs of those individuals. In this plot, the placement of the termination point of the videoconferencing sessions is designed based on the participants' location. Specifically, these edge-cloud points consist of servers that would be collocated in office printers. This idea clearly follows the edge-cloud paradigm, i.e. placing the server at the edge of the network. Normally, printers are nearby workers and consequently, placing the servers inside these traditional appliances gives the opportunity to reduce the distance between the user and the server.

The project has been divided into two phases. The former focuses on the aggregation of videoconferencing sessions in the edge-cloud and the corresponding SDN functionalities required. The latter, users will be equipped with AR goggles to join the teleconferencing sessions and hence low latency will be required. In this second phase, the implementation will expand to allow in-network function placement and low the latency of the sessions. This thesis is only focused on the first part of this project and all the information found in this document makes reference exclusively to this phase. Nevertheless, the fact that this project will be further developed by the time this thesis is concluded has been crucial at the time of taking some decisions. In other words, some of the made choices during the development of this thesis obviously took into account the roadmap of the overall project.

1.1. Objectives

The aim of this project is to showcase the capabilities of Edge cloud computing in dynamic management of data traffic, and in-network function placement in order to improve the videoconference experience. Combining Edge cloud computing and SDN proves to be a good approach to achieve a critical enhancement in the Quality of service (QoS) of real-time communications. Therefore, this proof of concept (POC) checks the impact of bringing the servers to the edge in the quality of videoconference services.

In order to achieve this broad objective, several tasks have to be attained. Firstly, an edge network model has to be presented so as to depict the scenario in detail. Secondly, a concrete case study has to be selected, i.e. a particular videoconference service. The chosen application need to be studied so as to comprehend how the quality can be correctly measured. Only afterwards, it will be possible to start evaluating the performance of the approach presented. Regarding this evaluation, the first test must highlight the impact of Edge cloud vs different locations, i.e. demonstrating how the latency can be reduced by means of bringing the servers to the edge. A second test must show the feasibility of having a dynamic system where the servers can be even live-migrated from one node to another. Finally, for the sake of providing a complete analysis of the videoconference quality, the impact of the available bandwidth and the packet loss in the system must be studied. Nevertheless, it is well known that in the case of videoconferencing, the latency is the principal matter of lack of quality. For this reason, this last must be analysed in much more detail.

1.2. Motivation

This project was born of the agreement between King's College London and Konica Minolta Laboratory Europe (KMLE), who pioneers office solutions and has an R&D effort on its next generation. KMLE is the hub where innovative solutions in the field of ICT come to life to transform the next generation of products and services from Konica Minolta. As a research organization, KMLE shares innovative projects and ideas with its network of academic and industrial partners. The work foreseen in this project provides a POC for a way forward in

the next generation *Workplace Hub* [2] and allow the in-building network of large offices to run more efficiently. Particularly, the project focus on delivering bandwidth exhaustive applications such as videoconferencing by means of Edge cloud computing and SDN.

1.3. Work plan

Figure 1 depicts the work plan followed throughout the development of this thesis.

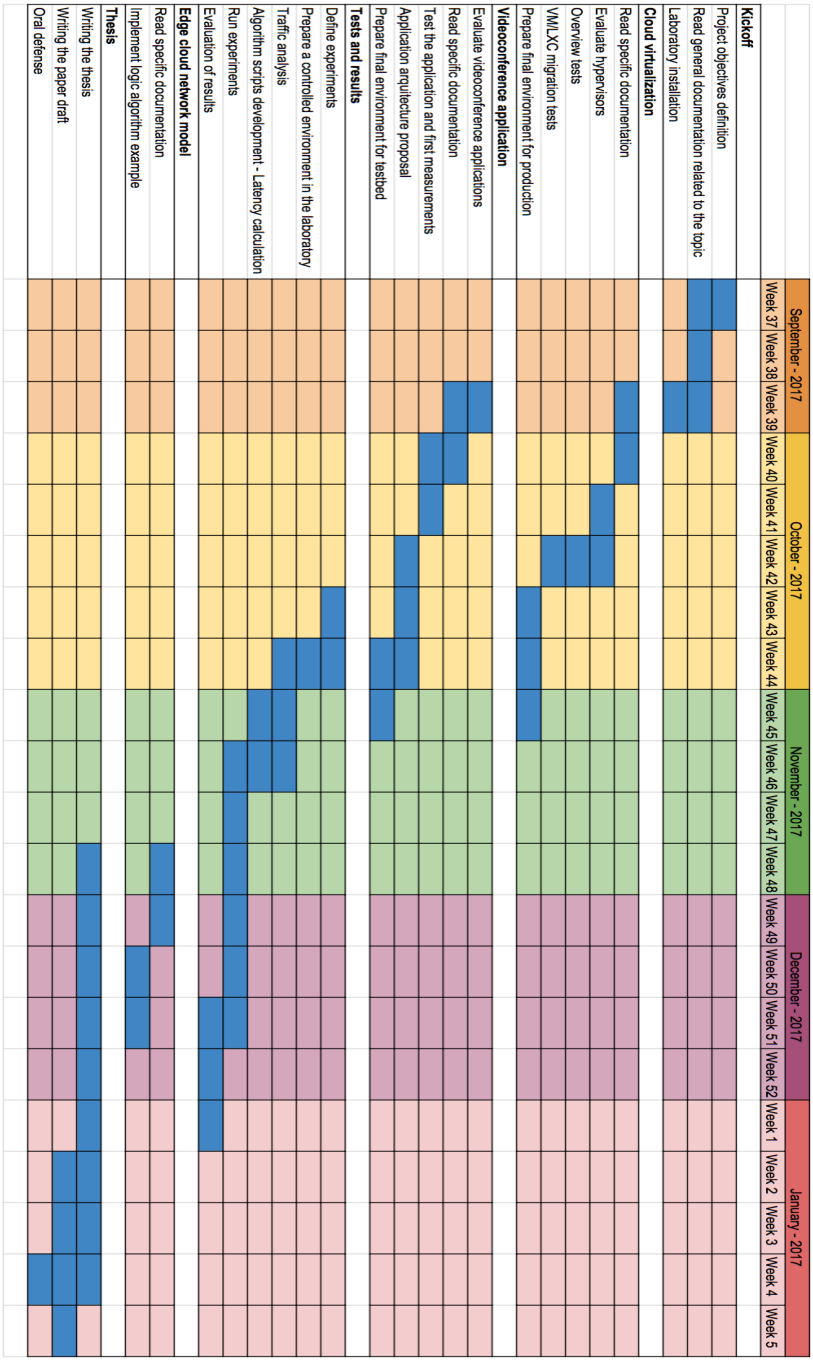


Figure 1 – Gantt chart

1.4. Deviations

The most relevant time deviation in the presented work plan has been the task “Prepare the final environment for testbed”. In theory, this task was completed on time, i.e. week 42, however, two weeks later a misconfiguration in the switch was found. In a nutshell, the port configured in a sniffing mode was duplicating the packets. The port was replicating ingress and egress packets in the switch and therefore the traffic observed was mirrored. This error in the configuration led to a revision of all the experiments in progress and the repetition of them.

Another remarkable deviation has been the fact of using physical machines for the clients connected to the videoconference while the experiments were performed. At the beginning of the process, the clients were emulated with Virtual machines (VMs) but after some testing, it resulted harmful since the traffic due to Virtual network computing (VNC) connections with Proxmox nodes was biasing the results. When this phenomenon was detected, it was needed to find available physical machines in the university so as to gather them and attached them to the testbed. Of course, this logistical issue provoked a certain delay in the progress of the experiments.

1.5. Organization of the document

This document is composed of different sections. Firstly, the State of the Art evidences the current tendency of Edge cloud computing at the same time that the work done so far in this field is presented. Moreover, this section provides the needed background in reference to the application used in this POC to comprehend the subsequent sections. Afterwards, the Methodology presents the suggested architecture for production environments and provides the procedure followed to achieve computing the latency once the application’s traffic is studied. Then, in Test and Results, the tests performed are disclosed and the outcome of those are presented. Finally, a conclusion is provided and future development is suggested as well.

2. State of the art

In this section of the document, the technologies in use in the POC developed in this project are presented. The purpose of the sections below is to put in context the project and provide reliable references of the potential and advantages of all the technologies considered. Firstly, an overview of the cloud environment is provided. It is crucial to understand this technology and its current status among the different industries to better comprehend its evolution towards Edge cloud computing. Afterwards, in a different subsection, a high-level description of a key-enabler of Edge cloud computing is provided, i.e. SDN. Finally, in order to provide some background about the application used as a case study, the last subsection bids a description of the specific videoconference used in this project.

2.1. From Cloud to Edge Cloud computing

Cloud computing is the practice of using a network of remote servers on the Internet to store and process data, rather than a local server or a personal computer. During the last decade, cloud computing has grown from being a promising business idea to one of the fastest growing sectors of the IT industry [3]. A definitive deal breaker of Cloud computing paradigm is the fact that it gives the opportunity to any kind of enterprise to deploy applications and offer services without the costs associated of holding their own premises [4]. Another interesting characteristic of Cloud computing is the ability to scale computing capacity up or down on-demand basis. This elasticity gives the chance to adjust the resources depending on the real demand. This feature is especially important for the companies which their demand is very seasonal and suffer a lot of demand peaks [5]. Moreover, charging policies are elastic as well [6]. In the past, when a company wanted to make use of an IT system or offer any kind of service on the Internet to their clients, they had to face a considerable initial investment to buy the infrastructure required that, of course, only well-established companies could handle. Nowadays, the Cloud offers pricing models that fit better to start-ups [7], which often cannot afford to spend large sums of money at the beginning of their business journey.

Companies are increasingly aware of the business value that cloud computing brings and are have already taken steps towards the transition to the cloud. Anyhow, a smooth transition entails a thorough understanding of the benefits as well as challenges involved. Of course, the main challenge is delivering reliable and available services [8]. Cloud-based services have the potential to meet or exceed the service reliability and availability requirements of traditional deployments only if there is awareness of the proper procedures to perform cloud operations and the new points of vulnerability are recognized. Another fundamental concern relies on how Cloud computing addresses the security and privacy concerns [9].

Cloud computing is going through a fundamental shift in which the traditional model of accessing highly centralized resources is replaced by a distributed, decentralized architecture [10]. This new paradigm, known as Edge cloud computing, brings the core building blocks of cloud – compute, storage and networking – as close to the consumers

as possible. Nevertheless, edge computing is in its early days and vendors from multiple segments are positioning it in different forms. The current market landscape consists of players from the public cloud, networking companies and industrial automation companies [11]. For the sake of simplicity, we could divide Edge cloud computing solutions into two kinds: Fog computing or edge computing [12]. In a nutshell, the former pushes intelligence down to the local area network level of network architecture, processing data in a fog node. The latter pushes the intelligence, processing power and communication capabilities of an edge gateway or appliance directly into devices. However, it must be said that on many occasions the terms of edge-computing and fog-computing are used frequently and interchangeably [13]. At any rate, by moving the computing nodes closer to the origin of the data, the latency involved in the roundtrip to the cloud gets reduced. This benefit has been already exploited in many fields such as gaming [14], healthcare [15], Internet of Things (IoT) [16] or video streaming [17]. Though IoT might be the key driver of Edge cloud computing, many other use cases are accelerating the pace of adoption. Edge cloud computing is all set to become the most preferred architecture for running data-driven applications [18].

The underpinning for the majority of clouds is a virtualized infrastructure which allows increasing hardware utilization density [19]. This is one of the technologies that enables elasticity, and so has provided increased flexibility in terms of speed of deployment, dynamic auto-provisioning, and cloud management [20]. Nowadays, there are multiple vendors offering virtualization environments. A determining factor during the cloud environment selection process in this project was the live migration feature [10]. For the sake of synthesising, only the one used in the testbed of this project is commented, i.e. Proxmox Virtual Environment (VE) [21]. This platform for virtualization tightly integrates KVM hypervisor [22] and Linux containers (LXC) [23], offering live migrations for the former. Therefore, a comparison between the use of these two technologies could be performed similarly to [24] where a review of Edge cloud requirements and the suitability of container technology is offered. Proxmox also handles high availability clusters and disaster recovery tools with the built-in web management interface. Moreover, it is an open-source project. For all these reasons, it seemed to be such a convenient tool for running the experiments involved in the project.

2.2. Virtualized network

Telecommunications networks contain a variety of proprietary hardware. The launch of new services usually demands network reconfiguration and on-site installation of new equipment. By the same token, the innovation cycles require greater flexibility and dynamism than hardware-based appliances allow. In the same way that applications are supported by dynamically configurable cloud environments, virtualized network functions allow networks to be agile and capable to respond automatically to the needs of the traffic and services running over it [25]. Key enabling technologies for this vision are Network Functions Virtualisation (NFV) and SDN.

On one hand, NFV is an initiative to virtualize the network services that are now being carried out by proprietary, dedicated hardware. The goal of NFV is to decouple network

functions from dedicated hardware devices and allow network services that are now being carried out by routers, firewalls, load balancers and other dedicated hardware devices to be hosted on VMs. This capability is important because it means that network administrators no longer need to purchase dedicated hardware devices in order to build a service chain [26]. Any server capacity can be added through software so IT departments can respond in a more agile manner to changing business goals and network service demands.

On the other hand, SDN introduces the ability to program the network via a logically software-defined controller, and separate the control plane from the data plane [27]. Compared to traditional networking paradigms, SDN makes it easier to introduce new abstractions in networking, simplifying network management and facilitating network evolution. SDN allows logical centralization of control, and decisions are made by the centralized controller, which has a global view of the network [28]. This controller installs control demands to the devices in the data plane by means of a well-defined API between them, for instance, OpenFlow [29].

The potential of NFV and SDN for facing the challenge of adapting Edge cloud computing has been discussed for a while and seems to be a key enabler of the future 5G infrastructures. In [30], a preliminary architecture for a 5G infrastructure is presented based on the latest advances in SDN, NFV and Edge computing. On this wise, the architecture aims to reduce dramatically the end-to-end latency for mission-critical type of traffic. In [31], the authors highlight the opportunity of integrating NFV and SDN for building up an edge-cloud based architecture for IoT. In [32], the authors provide an analysis of SDN and NFV different architectural options to address Edge cloud computing. In [33], the authors propose the introduction of SDN-enabled containers to support SDN/NFV applications located at the network edge which are able to trigger on-demand connectivity services. In this case, a video analytics use case is demonstrated. As a good summary of this trend, [34] deliberates how can Edge cloud computing benefit from SDN making use of different use cases.

Following this tendency, the services are increasingly provided by cloud computing systems and they are often accessed while on the move. Therefore, the mobility of users becomes such a relevant topic to confer. In [35], the authors propose an OpenFlow based implementation for migrating services in unison with the user's movement, i.e. follow-me cloud. The basic idea behind this concept is that services provided by a cloud follow users throughout their journey. Of course, the follow-me cloud concept could be implemented by different technologies. However, as shown in [36], SDN and NFV can be used for distributed mobility management instead of the existing protocols and the test results demonstrated that the proposed approach is an efficient mechanism.

2.3. Videoconference application

Since the purpose of the project was not developing a videoconference service but using it as an application example, a research of open source videoconference applications had to be done. The criteria of selection have been strictly based on previously defined

requirements: the videoconference application had to handle high-quality (HQ) sessions and an unlimited number of users. Aside from that, it could be interesting selecting an application that could be easily adapted for developing an AR module in the future phase of the project. The different open source applications that had been under study during the application selection process are listed in Table 1. In this table, it is clearly shown that just two of the applications fully met the requirements defined.

App	HQ	max. # users	Potential AR development
Jitsi	Yes	Unlimited	No
Kurento	Yes	Unlimited	Yes
Licode	Yes	Unlimited	Yes
Hubl.in	No	9	No
Spreed	No	Unknown	No
Janus	No	Unknown	No

Table 1 – Videoconference applications

Kurento [37] and Licode [38] are both open source projects that offer the application and server side to run a videoconference plus other capabilities. Although both projects met the requirements, Kurento offered many more application examples and documentation. Moreover, Kurento seemed to be much more modular what would help to develop AR features as outlined in the project plan [39]. For these reasons, Kurento became the videoconference solution chosen for the POC and a brief study of its architecture has to be presented in section 2.3.1.

In [40], a cloud-oriented solution that uses Kurento is proposed. This platform, known as Nubomedia, follows a Platform as a Service (PaaS) scheme based on ETSI NFV. Nevertheless, Nubomedia did not fit the testbed of this project since it required hardware that could support OpenStack [41] as in a real production environment. Consequently, just Kurento was used in this POC.

2.3.1. Kurento

Kurento is a WebRTC media server and a set of client APIs making simple the development of advanced video applications for WWW and smartphone platforms. Kurento Media Server (KMS) features include group communications, transcoding, session recording and mixing, broadcasting and routing of audiovisual flows. As a differential value, KMS also provides advanced media processing capabilities involving: computer vision, video indexing, AR and speech analysis [42]. Kurento modular architecture makes simple the integration of third-party media processing algorithms (i.e. speech recognition, sentiment analysis, face recognition, etc.), which can be transparently used by application developers as the rest of

Kurento built-in features. KMS capabilities are exposed by the Kurento API implemented by means of libraries called Kurento Clients. Kurento offers two clients out of the box for Java and JavaScript. However, if it is necessary to develop in any other language, Kurento can be used too. For this particular project, Java out of the box client was selected.

The architecture is based on a client-server model [43]. In Figure 2, the client-side applications make reference to the users that run the applications on their browsers, mobile or any other medium. Otherwise, the application server is the responsible for the signalling whereas the KMS is responsible for media stream communications. In the particular case of a videoconference service, the client connects to the application server through a browser. The client requests joining a call and the application server establishes the session. From that moment, the client will exchange media streams with the KMS whereas the application server will keep performing signalling tasks.

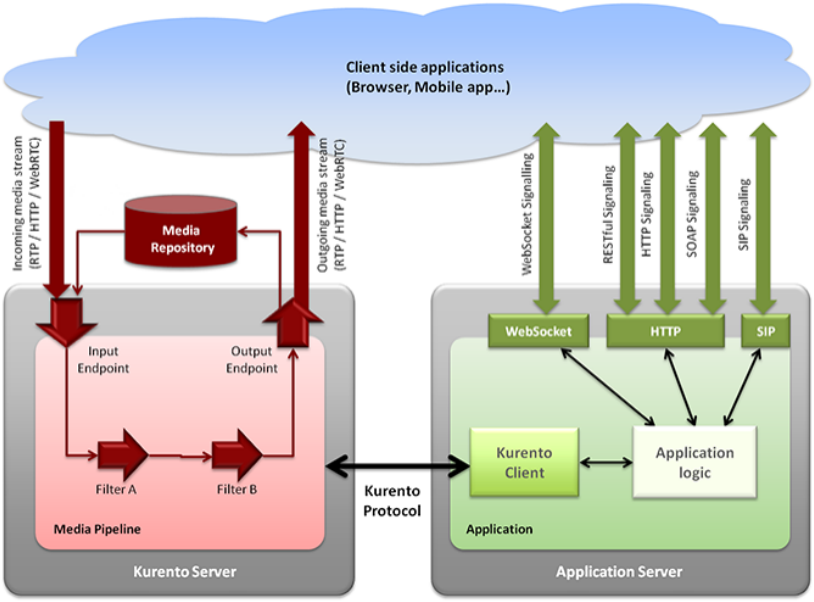


Figure 2 – Kurento architecture [44]

Although the purpose of this section is not understanding the details of each component but to comprehend the overall system, some of the modules pictured in Figure 2 deserve being detailed. This is the case of the filters that the KMS contains which process the incoming data to forward it once processed. Kurento offers a lot of filters, some of them biding AR, but what it is even better, is the fact that new components could be developed and included in the server, thanks to its modularity. Also in the KMS, a database can be used as the media repository, i.e. storing the inputs and outputs of a session. Lastly, in the application server, there are two main blocks that basically depict the fact that there's a Kurento Client component that is responsible for establishing the communications with the KMS whereas there's another component that describes the proper application logic, handling the signalling of the communications system.

The outcomes of this brief study of Kurento were crucial to the development of the project. Discerning that the signalling and the media had two separate planes was a complete deal-breaker. Following the edge-cloud computing paradigm, it seemed reasonable having the KMS as near as possible to the clients whereas the location of the application server became not a critical issue.

3. Methodology

3.1. Edge cloud network model

In this section, the architecture of a production environment is presented. The suggested approach consists of a well-known architecture as a result of its acceptance in the Cloud industry. Since the goal of the project is not to design a new architecture but to offer an Edge cloud computing environment, an eminent implementation was taken as a reference [45]. The combination of OpenStack and SDN has been widely used already and the results show that it is a feasible and valid solution [46] [47] [48].

The starting point is having the computing nodes as near as possible to the users. These are the edge computing nodes whose task is to serve the clients with the lowest latency possible. A good location for these servers could be the office printers where a server can be easily attached to each of these appliances. The computing resources given to each of these printers are independent computing nodes that shape a bigger system. In order to define this pool of computing nodes, a cloud operating system is required. The cloud operating system is responsible for controlling the nodes, the storage, and the networking resources throughout the datacentre, which in this case consists of a set of distributed computing nodes attached to the different printers in the same office.

In this section, for the sake of illustrating the architecture, OpenStack is used as an example. Nevertheless, it is important to highlight that the architecture is agnostic to this technology and any other alternative could be used instead. In fact, Proxmox has been used for the testbed of this POC, see section 4.1. OpenStack and Proxmox both use KVM hypervisor so all the implementations performed in Proxmox are equivalent to OpenStack. Given the fact that the goal is not deploying a production environment but proving the advantages that this kind of architecture can provide, the experiments of this project were run on Proxmox which resulted more economical.

In Figure 3, the different modules that OpenStack offers are depicted. In this case study, just the following modules are of our interest: Nova, Glance and Neutron. Nova is basically the physical computing nodes, whereas Glance is the image repository where the virtualized instances of servers are stored, and finally, Neutron is the networking service of OpenStack.

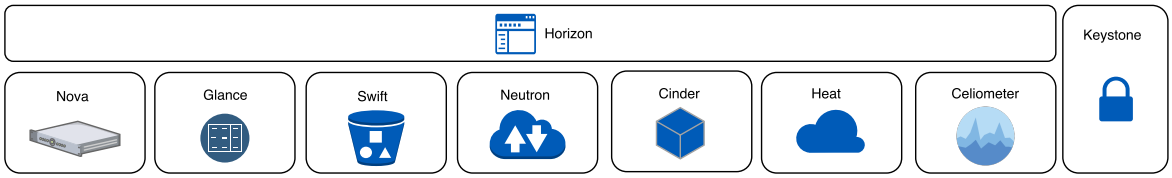


Figure 3 – OpenStack modules

As seen in [45], a good approach to control the resources in a system like the one exposed above is by means of using an SDN controller. This particular approach integrates an SDN controller based on OpenDaylight (ODL) [49] to OpenStack. Anyhow, please keep in mind that in the overall architecture the SDN Controller is again technology agnostic. In the

particular example of OpenStack and OpenDaylight, Neutron interacts with the SDN controller via an API REST and the SDN controller controls the network through OpenFlow.

In Figure 4, the architecture diagram is depicted. The Image repository is the analogy to Glance in OpenStack, which stores the virtualized servers. The physical computing nodes would compose Nova and finally, the network manager would be Neutron. This last is the responsible for interacting with the SDN controller which is the intelligence entity used to change the path of the network packets across the network and send instructions when necessary.

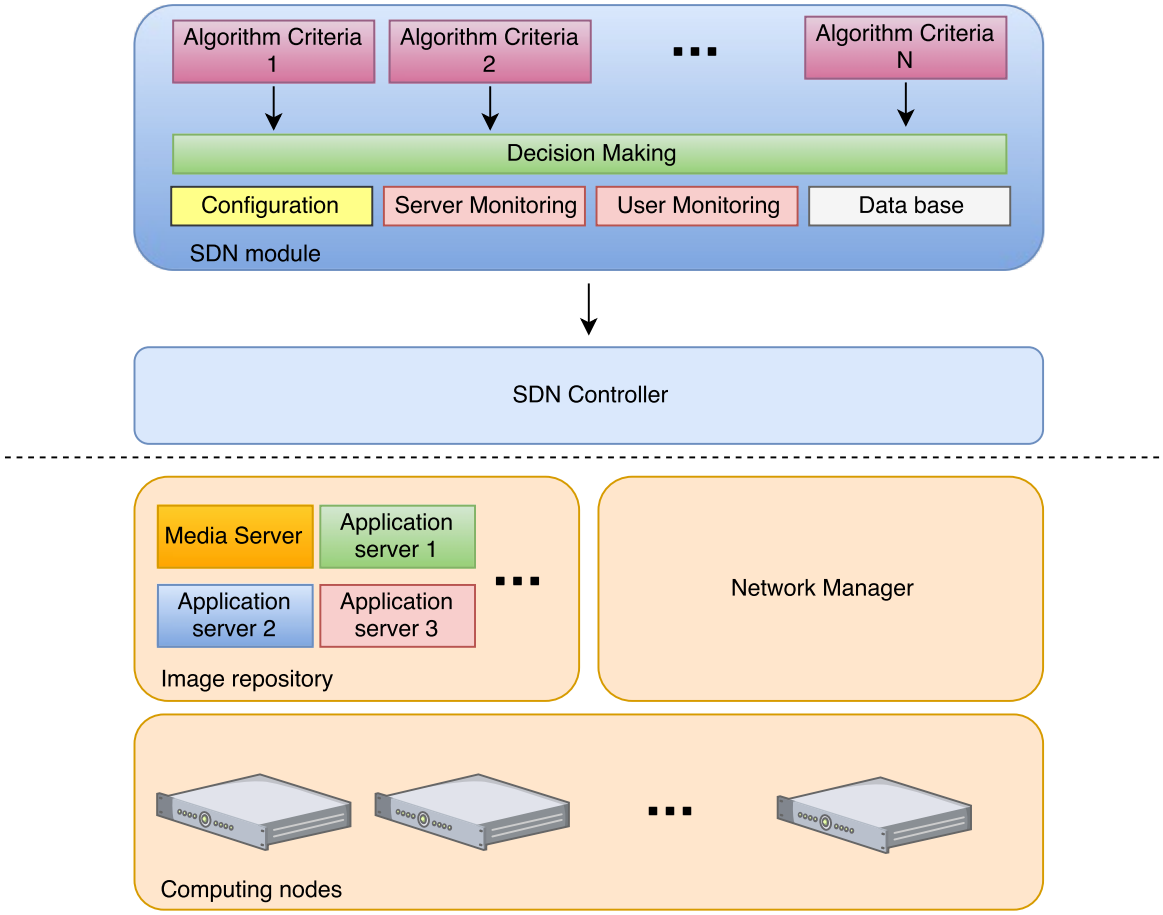


Figure 4 – Architecture diagram

Figure 4 also illustrates how an extra module could be attached to the SDN controller, in [50] a similar approach can be found. In the case of OpenDaylight SDN Controller, an API REST is exposed and this will be used to append the module. The purpose of this module is providing the intelligence to the controller to take action when necessary. Thanks to the nature of SDN, this module can gather networking metrics, i.e. monitoring the server and the user, and afterwards store this information in a database to be used as input in the algorithms running. Notice, that there are different algorithms criteria in the module that can be used depending on the business case. In this case study, the focus is on the latency so this module would be the responsible for computing the latency and providing an outcome to the controller. The result can be, for example, the decision of migrating one server instance from one computing node to another. In brief, these algorithms decide if a change

in the network configuration is required or an instruction has to be sent to one of the nodes hosting certain service. They are the responsible for providing decisions in function of the data gathered and the criterion followed.

The purpose of this project is not the development of these modules, however, in Appendix 1, a pseudo-code implementation example is shown. In this example, the module provides the intelligence to the controller so as to forward the necessary instructions to the computing nodes when the migration of the KMS instance is required.

As a summary, the architecture consists of a cloud operating system such as OpenStack to build up the cloud infrastructure and an SDN controller that is used to run the intelligence components needed to automate decisions. The end purpose of the presented architecture is providing a system that enables adapting dynamically the environment in function of the data gathered. SDN allows obtaining valuable networking metrics such as bandwidth, packet loss or jitter. Furthermore, with this information, other relevant metrics can be computed, such as the latency. All this data can be conveniently stored in a database and this information can be used for decision making. The most exciting feature is the fact of having the opportunity of changing the criterion in the SDN module. Depending on the application, the decision criterion might change. This flexibility is priceless in production environments where the business cases constantly change.

3.2. Videoconferencing application

It is well known that the traffic of bandwidth exhaustive applications impacts the network profoundly [51]. Consequently, in order to have a better understanding of the application selected, some exploratory study had to be done. The first preliminary evaluation tested the hardware required to offer a stable service. The second exercise checked the resources needed in terms of bandwidth so as to ensure that the POC network could be correctly dimensioned and the application met the HQ requirement. Anyhow, before presenting those evaluations, the data flow of the communications involved in the videoconference between two clients deserves being commented.

After the study of Kurento architecture in section 2.3.1, we know that the client interacts with two entities, the application server and the KMS. In a two client communication, each client connects to the application server which offers the possibility of creating or joining a videoconference session. Once both clients have joined the same session, they exchange the media stream with the KMS, see Figure 5. This figure shows how Client A and Client B never communicate in a peer to peer manner but on a client-server basis.

Notice that, for the sake of simplicity, Figure 5 depicts just one flow of data, i.e. the data coming from Client A to Client B. Obviously, it is equivalent in the other way around for the communications from Client B to Client A.

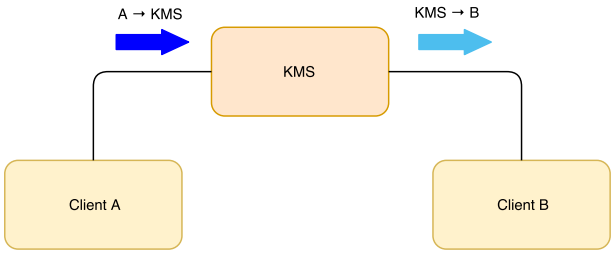


Figure 5 – Data flow diagram

3.2.1. Hardware requirements

A quick hardware requirement test was performed for the sake of setting up the testbed with the adequate hardware to support the traffic and processing load. For this reason, making use of three available spare machines in the workspace, the following ad-hoc environment was set up:

- Machine 1: This machine hosted one VM running the KMS and another VM running the application server.
- Machine 2 and 3: Each machine hosted a VM that simulate a client using a browser to connect to the videoconference session.

All these machines were connected to the same network so there was completely connectivity between them.

➢ **KMS**

The KMS had to be tested in order to be able to justify the specifications requirements for the virtual servers that the testbed would need. During this test, the focus was completely on the VM that runs the KMS since it was modelling the future server in the testbed. Therefore, the main specifications of the VM hosting the KMS in Machine 1 are specified in Table 2.

CPU	**4**
RAM	4GB
Storage (HDD)	30 GB

Table 2 – KMS VM specifications

Once the KMS and the application were running, each client connected to their respective browser. Firstly, each client opened just one browser, simulating a one to one call.

Secondly, each user opened different tabs (sessions) in their browsers to simulate extra users. For this reason, in this section, the outcomes are showcased for two and four clients.

The tools used to record the performance of the server while running were *htop* [52] and *bmon* [53]. In a nutshell, the former provides memory and processors information whereas the latter informs about the traffic in the running interfaces. However, for specifying the server requirements, we focused only on *htop* since *bmon* only provided information about the data flow.

During the testing period, it was observed that the server could easily handle the communication among all the users and the performance was very good. As an example, in Figure 6 and Figure 7, it is shown that for two and four users respectively, the usage of the processors is minimal and that the memory is almost idle.

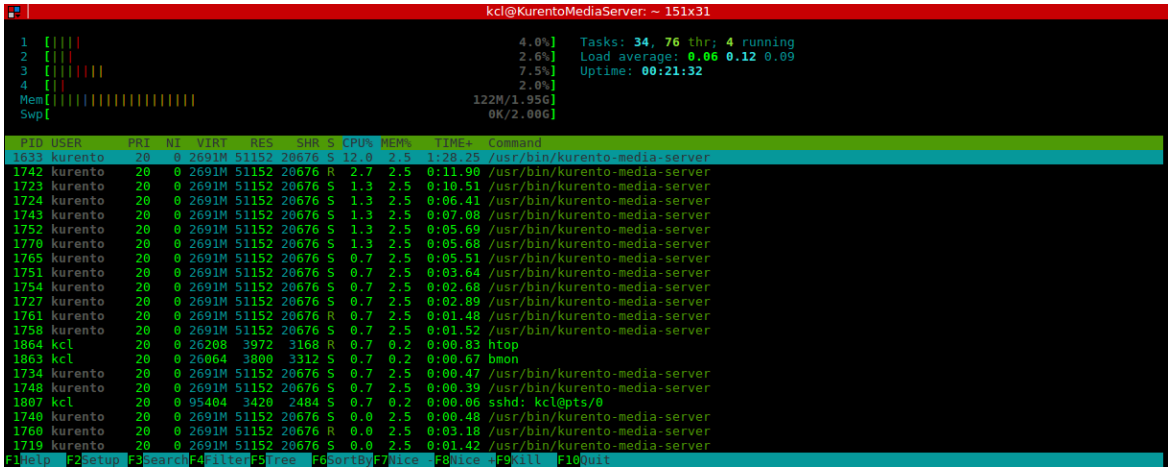


Figure 6 – htop: KMS performance screenshot for two users

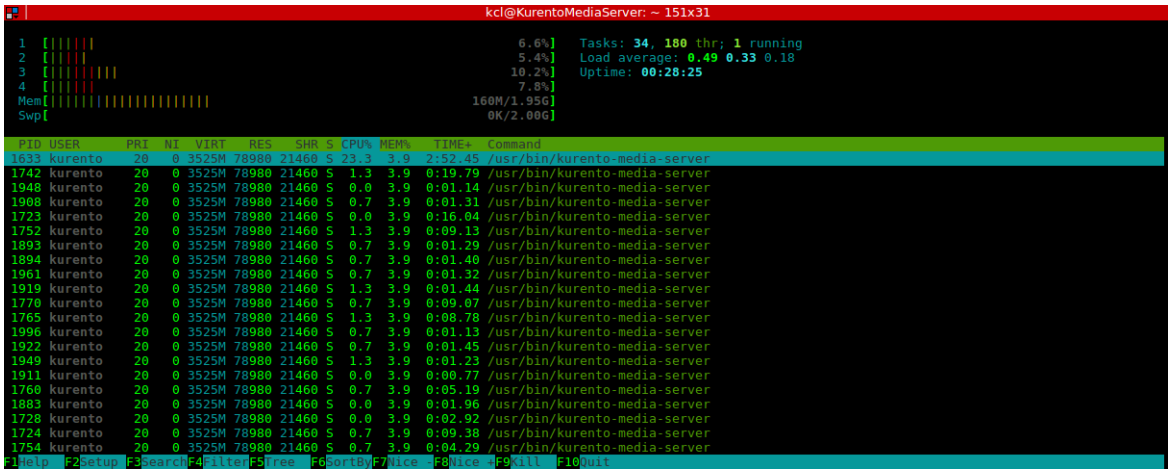


Figure 7 – htop: KMS performance screenshot for four users

As shown above, with a regular machine it is possible to offer such a service. This tests gave credibility to the fact that Kurento claimed to be possible to host this server on any regular machine and not high-performance servers were required. It could be found in the literature [54], that the KMS could handle up to thirty users with machines of 8 CPUs and

16GB RAM. However, since the analysis performed did not require such amount of clients, the specifications shown in Table 2 are good enough for the testbed set up.

- **Application server**

The analysis of the application server was much more simple. In this case, the VM that runs application server is just the responsible for the signalling as detailed in section 2.3.1. This application server could be even co-hosted in the same VM that runs the KMS but for experimental reasons, it has been always hosted in a different machine. In that way, an extra degree of freedom was achieved.

Making use of the same tools commented in the case of the KMS, it was proved that any machine with specifications such as listed in Table 2 could perform the signalling tasks correctly.

3.2.2. Bandwidth consumption

The main requirement the application had to meet was handling HQ communications. As a result, it was necessary to check that the application taken as an example met the previous condition. For this sake, several traffic captures were performed while an ongoing session was running. These traffic captures were used to compute and plot the bandwidth. In Figure 8, it is shown the bandwidth for each of the existent communications, i.e. Client A to KMS, KMS to Client B, Client B to KMS and KMS to Client A. In this figure, it is shown that, approximately, each of the communication consumes 600Kbps.

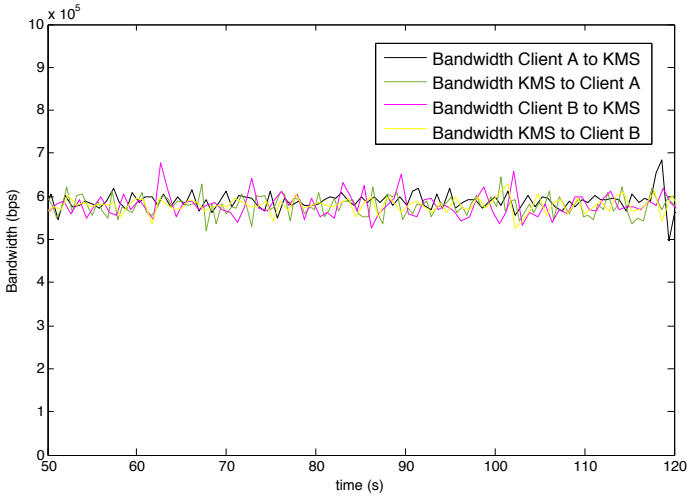


Figure 8 – Bandwidth consumption (bps)

In [55], it is defined that in order to support HQ videoconferences, it is required a bandwidth above 400Kbps. Since the clients were above this threshold, the fact that the KMS can handle HQ communications was proven.

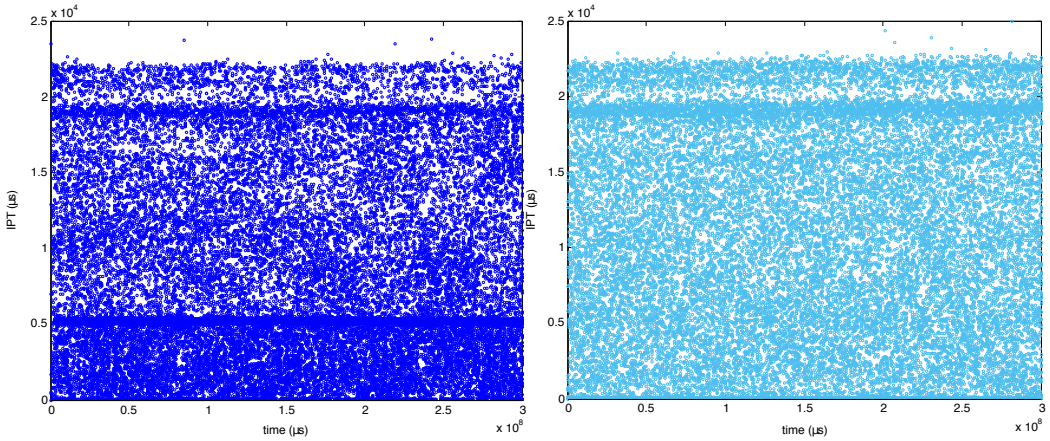
3.2.3. Analysis of traffic and latency calculation

Once it was confirmed that the videoconference application was suitable for our study in terms of hardware and bandwidth requirements, a deeper traffic analysis was performed. The purpose of this investigation was to observe the traffic so as to know how to compute the latency during the tests.

The very first and trivial approach to compute the latency is based on its definition. The end-to-end latency between Client A and Client B can be obtained by computing the time difference between when a data packet egress Client A and when it actually arrives at Client B. Unfortunately, the scenario under study does not consist of a single link where the data goes straight from Client A to Client B, there is the KMS in the middle, see Figure 5. This figure depicts that there are two links for the same data flow. The data flow consists of packets sent from Client A to the KMS and from the KMS to Client B. Notice that again, for the sake of simplicity, this study only refers to the communication from Client A to Client B but it is obviously reciprocal. In any case, the presence of the KMS in the middle of the communication is very challenging. Since the packets are sent through this entity, in order to compute the end-to-end latency, it is required detecting for KMS ingress packets the corresponding KMS egress packets. If this packet association is possible, it will be feasible measuring the time difference between when a packet departures one client and when arrives at the other, and this is, of course, the latency. This latency will be the sum of the latencies of both links observed in Figure 5 plus the processing time of the KMS.

After sniffing the traffic, it was observed that the KMS does just not forward packets but it processes them. By means of observing the traffic captures in both links, and by simply noticing that the number of packets in each link was different, it could be concluded that in some cases, the KMS directly forwarded the received packets but in other occasions, it processed and grouped them. Given the observed the behaviour, computing the latency from Client A to Client B would be only possible if the relationship of at least some of the ingress and egress packets could be detected. Ideally, we would like to compute the latency for all the packets, but it has been proven that it is not possible to follow the strict definition in this case. For this reason, a new methodology to compute the latency had to be proposed.

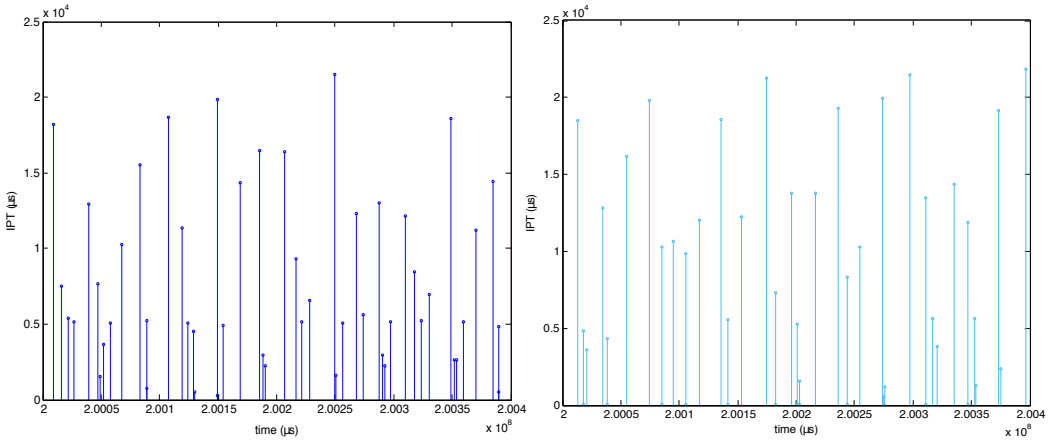
For the sake of knowing the traffic better, the inter-packet time (IPT) for both communications were examined, see Appendix 2. In Figure 9.a, we can observe that Client A generates most of its packets at certain rates, every 5ms and 20ms approximately. However, there is a considerable dispersion of the IPT that makes not possible to consider that the client was sending the packets at constant rates. On the other hand, in Figure 9.b, it is observed that the outgoing traffic of the KMS consists mostly of sent packets every 20ms but the IPT is really dispersed as well.



a) Client A→KMS b) KMS→Client B

Figure 9 – IPT

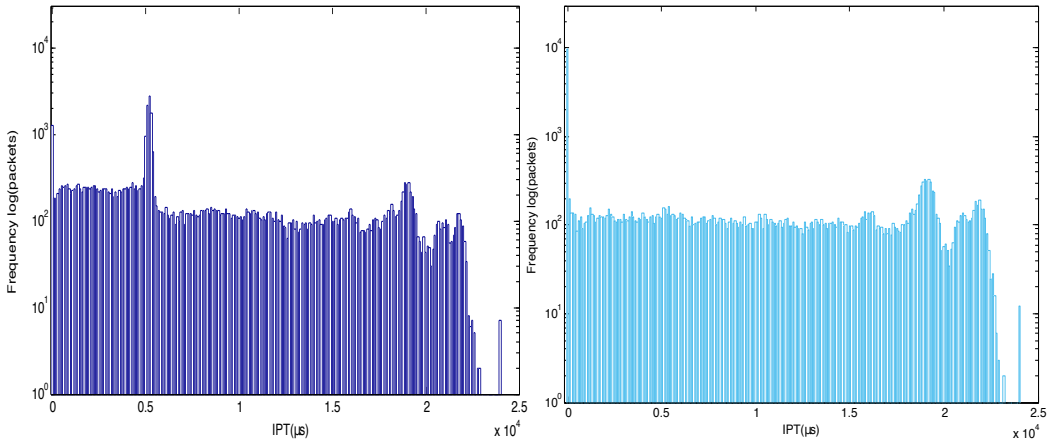
If the plots in Figure 9 are zoomed, an interesting outcome is discovered, see Figure 10. In this zoomed plot, a burst pattern is found for both data flows. There are different bunches of packets that travel together, i.e. small IPT between them, and some that have clearly higher IPT that would correspond to the first packet of each burst.



a) Client A→KMS b) KMS→ Client B

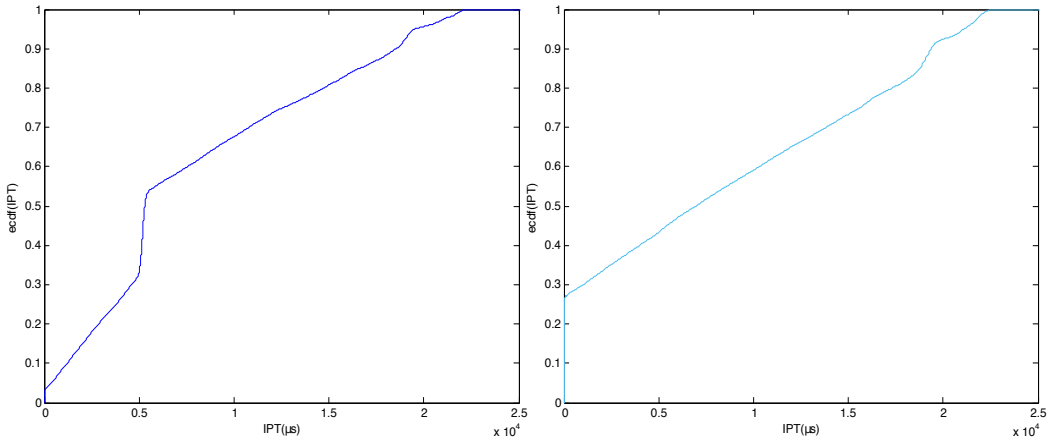
Figure 10 - IPT zoom

For the sake of corroborating this statement, the histogram and the empirical cumulative distributed function (ecdf) [56] of the IPT were figured out, see Figure 11 and Figure 12 respectively. Notice that in the case of the histogram, the y-axis is logarithmic.



a) Client A→KMS b) KMS→ Client B

Figure 11 - IPT logarithmic histogram



a) Client A→KMS b) KMS→ Client B

Figure 12 - IPT ecdf

In case of analysing the IPT of Client A→KMS communication, both, the histogram in Figure 11 and the ecdf in Figure 12, they show again that there is a considerable amount of packets that are sent at a rate of 5 ms and another peak at 20ms. However, the dispersion of the IPT commented above is again observed. When it comes to KMS→Client B communication, a peak in 20 ms is observed but the dispersion of IPT is depicted as well, both in the histogram and the ecdf. In any case, the most relevant conclusion we can obtain especially from the histograms of Figure 11 is the fact that there are numerous packets with a low IPT and numerous packets with high IPT. What this means is that both traffics can be modelled as a burst-based traffic. This phenomenon is depicted in Figure 13, where the IPT can be clearly differentiated from the inter-burst time (IBT) by the noticeable difference of its values.

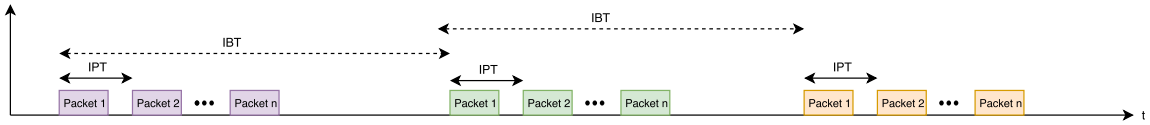


Figure 13 – IBT vs IPT

Notice that in Figure 11, the dispersion in the histograms indicate that the IBT and IPT are not constant at all. In other words, the bursts could contain more or fewer packets depending on the occasion because the IPT and IBT are diverse. Notwithstanding, knowing that this traffic behaves on a burst basis could give us the possibility of exploiting this pattern.

By virtue of modelling the traffic mathematically to ease the latency calculation, both data flows were evaluated trying to fit a well-known distribution. For this process, Easyfit [57] was used, a data analysis tool specifically developed for distribution fitting as performed in [58]. According to the results provided, none of the available distributions fitted with the data under study. Unfortunately, neither the traffic from Client A to the KMS nor the traffic from the KMS to Client B seemed to resemble a recognised pattern. This meant that the traffic could not be mathematically modelled on this wise. As proof, the results obtained are shown in Table 3. In this table, for each link, the best distribution for each statistical test is noted. For Client A to the KMS data, it can be observed that all of them differ, whereas the Normal distribution matches for two of the criteria in the case of the KMS to Client B data. Nevertheless, the results are far from being hopeful in any case since, for all them, the parameters indicate that the fits must be rejected. In brief, even for the best fit of each statistical test, the fits were not good enough to assume the data followed a well-known distribution.

Link	Parameters	Kolmogorov-Smirnov						Anderson-Darling						Chi-Squared					
		Distro	Values					Distro	Values					Distro	Values				
A → KMS	Sample size	Inv. Gaussian			39947			Gen. Extreme value			39947			Gen. Gamma			-		
	Deg. Freedom				-						-						15		
	Statistic				0.08826						616.88						13860		
	P-Value				0						-						0		
	Rank				1						1						1		
	α		0.2	0.1	0.05	0.02	0.01		0.2	0.1	0.05	0.02	0.01		0.2	0.1	0.05	0.02	0.01
	Critical value		0.00537	0.00612	0.00679	0.0076	0.00815		1.3749	1.9286	2.5018	3.2892	3.9074		19.311	22.307	24.996	28.259	30.578
	Reject?		Yes	Yes	Yes	Yes	Yes		Yes	Yes	Yes	Yes	Yes		Yes	Yes	Yes	Yes	Yes
KMS → B	Sample size	Normal			36564			Wakeby			36564			Normal			-		
	Deg. Freedom				-						-						15		
	Statistic				0.14003						1008.8						9077.6		
	P-Value				0						-						0		
	Rank				1						1						1		
	α		0.2	0.1	0.05	0.02	0.01		0.2	0.1	0.05	0.02	0.01		0.2	0.1	0.05	0.02	0.01
	Critical value		0.00561	0.0064	0.0071	0.00794	0.00852		1.3749	1.9286	2.5018	3.2892	3.9074		19.311	22.307	24.996	28.259	30.578
	Reject?		Yes	Yes	Yes	Yes	Yes		Yes	Yes	Yes	Yes	Yes		Yes	Yes	Yes	Yes	Yes

Table 3 – Distribution fitting results

In any case, thanks to the fact that the traffic behaves on a burst basis, it is possible to propose a method based on this pattern. Figure 14 shows the proposed method, where the first packet of each Client A→KMS burst can be corresponded to the first packet of each KMS→Client B burst and then the latency can be computed. This methodology

exploits the assumption that the data contained in the first packet of a KMS→Client B burst is the equivalent processed data of the data contained in the first packet of the associated Client A→KMS burst. Figure 14 also depicts that on some occasions, the KMS just forwards packets like the packets in burst 1 and in other cases, the KMS process and group packets like the packets in burst 2.

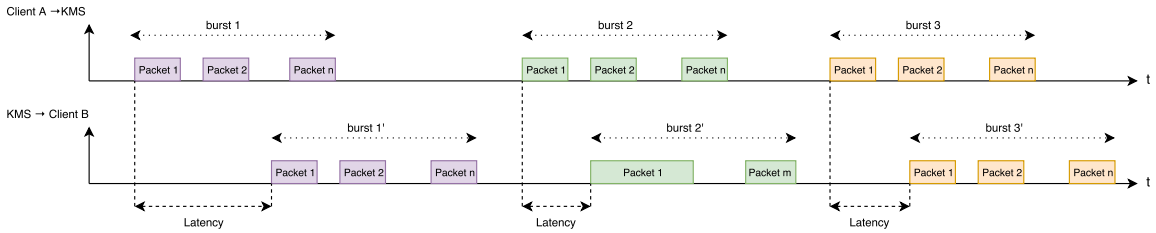


Figure 14 – Latency estimation model

Of course, with the methodology proposed, the latency cannot be computed for every single packet but at least for enough quantity of packets so as to obtain representative latency values.

4. Tests and results

In this section, the testbed implementation is detailed and the results for each of the experiments performed are presented.

The first experiment, Edge cloud vs different locations, aims to demonstrate the impact of the distance in the latency when the server is placed in different locations. This experiment pretends to motivate the placement of the server as close to the user as possible. In short, it was performed to analytically demonstrate the impact of the distance in the QoS, specifically in terms of latency, then justifying Edge cloud computing. The KMS was placed at different locations having a different relative distance to the client. Since the resources in the laboratory were limited and the server could not be hosted on a server physically located in an overseas country, the set of locations were emulated by means of tc and netem [59], Linux commands that enable configuring the network card in the server introducing the desired delay, see Appendix 3. As seen in [14], the emulated locations were chosen to take into account the real locations offered by Amazon Web Services (AWS) [60]. In order to use realistic latency values when emulating, the latency was computed by means of pinging these servers. In Table 4, where the distance is relative to the location of the laboratory, central London, the ping times obtained for each location are noted. Please, notice that the location London makes reference to servers that are in the same city as the laboratory but not in the edge since they were not placed in the office where the analysis was performed.

Location	Distance (Km)	Latency (ms)
Edge	0	0.451
London	10	6
Frankfurt	638	22
Virginia	6174	79
Oregon	7980	140
Tokyo	9569	276

Table 4 – Server locations and obtained ping latencies

For each location, a different videoconference session was established and the traffic between the clients and the KMS was captured for posterior analysis, particularly, to observe the latency in function of the distance. Before starting the tests, in order to corroborate the necessary session time, three different sessions of 5, 10 and 15 minutes were performed. When comparing those sessions, the traffic was barely the same. Consequently, all of them led to the same results so we could choose the most convenient, sessions of 5 minutes. Besides, notice that for each location the analysis was performed twice, with VMs and LXC.

The second experiment, Impact of Live migration, aims to show empirically the impact on the latency when migrating the server in an on-going session. The reason behind this interest is justifying how feasible the architecture proposed is since good results in this experiments would mean having a dynamic environment where the server can be easily relocated when necessary, even when the service is running. This migration pretends to be the kind of decisions that the SDN controller module could take in case of potential QoS improvement.

Like in the first experiment, the traffic captured consisted of a whole videoconference session between two clients during 5 minutes for each location where the tests were performed twice, for VMs and LXC. The difference relies on having a live migration of the server while the session is running. This means that the KMS is firstly hosted in one location and in the middle of the session, the server is live migrated to the edge. Again, in this second experiment, the focus relied on the latency. This experiment pretends to show how the latency can be reduced when the server can be located closer to the user, Edge cloud, instead of having it in a further location, the Cloud.

Aside from observing the latency, other QoS metrics such as bandwidth and packet loss were explored. The purpose of exploring the bandwidth was showcasing the importance of having enough network resources in the office in order to enjoy the benefits in terms of latency brought by Edge cloud computing. Not having enough bandwidth would vanish the effort of moving the servers around to improve the QoS. In section 3.2.2, the bandwidth consumed by the application in use was obtained. Having this value in mind, it was possible to define different available bandwidth during the videoconference. The idea behind this experiment was that the bandwidth of the application represents a certain percentage of the available bandwidth. To achieve it, *tc* and *netem* were used to configure the network card in the KMS, simulating the available bandwidth, see Appendix 3.

When there are network problems and some of the packets are lost, the user experience can be negatively impacted. For this reason, as an extra final analysis, a simplistic quality analysis is provided. The packet loss phenomena in the KMS for this experiment was emulated again by means of *tc* and *netem,* see Appendix 3. In this occasion, the network card deliberately dropped a certain percentage of packets as if the network was the one losing those. This last experiment highlights the impact the network can have on the quality and it details the behaviour of this particular application when struggling with packet loss.

For all the experiments, the analysis of the traffic had to be performed as follows. In order to capture the whole session and obtain relevant data, the capture always began before the videoconference had started and finished once the videoconference had ended. It must be highlighted that the data could not be obtained from raw captures but filtered ones, see Appendix 4. Then, in the case of the first and second experiment, where the latency was computed, the methodology exposed in section 3.2.3. was used to compute the latency, see Appendix 5.

4.1. Testbed implementation

The three subsections below describe how the scenario where the experiments were run was prepared with the assets required for their success.

The laboratory scenario consists of a set of Proxmox nodes, where all the VMs and LXC are hosted. In this case, with two nodes is enough to perform the experiments. Although these nodes should be installed in servers, on this wise, for the sake of having an economical setup, Proxmox nodes are installed in regular PCs. These machines are the analogy to the computing nodes that will be attached to the printers mentioned in section 3.1. In other words, these nodes are the edge-servers in the office.

One host per client was used, these are Client A and Client B in Figure 15. A different host had to be used for displaying the Proxmox Guest User Interface (GUI), just in case any administrating task was required without the disposal of a terminal. And finally, the Sniffer

consisted of a different host whose work was just capturing all the traffic that received through the interface connected to this network, see section 4.1.2. In this figure, it can also be seen that the machines exposed above were all connected to the same Local Area Network (LAN), in this case, under the same switch. Please, also notice that inside the Proxmox nodes, either the KMS or the Application, both could be VMs or LXC indifferently.

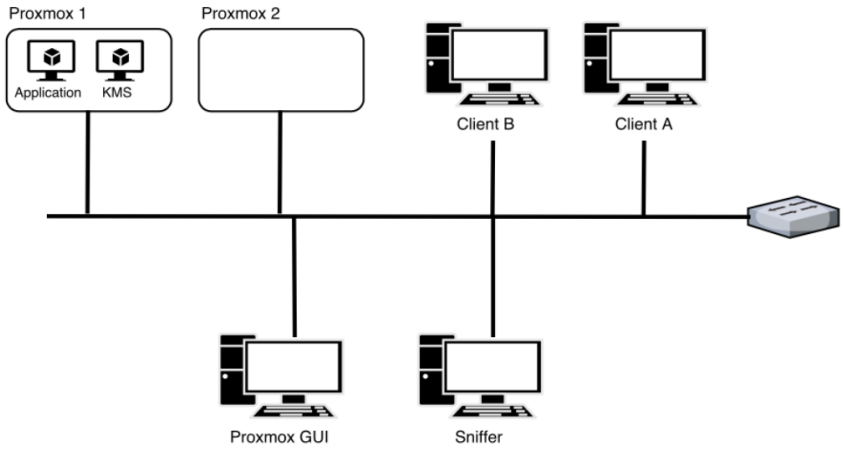


Figure 15 – Laboratory configuration

4.1.1. Resources needed

The setup of the testbed was simple and economical since there was no need of using high-performance devices, all of them were regular PCs and a switch. As a reference, check the specifications of the devices in Table 5. Additionally, the respective Ethernet cables are required since all the links are wired based.

4.1.2. Conditions and requirements

The most important condition to run the experiments correctly was having a completely controlled environment. Since the main goal of the experiments was to observe the latency performance, the impact of the network conditions over this parameter was critical. For this reason, the scenario had to be completely isolated from any other traffic, otherwise, the measurements obtained would be tremendously affected by external agents. In order to have a controlled scenario, a new network had to be set up. This network had to be completely independent of university premises so as to have a configurable and clean scenario where all the traffic was originated by the machines under study in the testbed.

Aside from having this self-contained scenario, it must be said that it would not have been possible to perform the traffic captures in the university network. Normally, public access points do not allow that any of the clients sniff the traffic in the network, obviously for

Device	Specifications
Switch	Cisco Catalyst 2960G Series
Proxmox nodes	i7-6700/3.4GHz 16GB RAM 500GB
Sniffer	i5-5257U @2.7GHz 8GB RAM 256GB
Client PCs	i7-3770 @3.4GHz 8GB RAM 500GB
Proxmox GUI	

Table 5 – Laboratory Hardware specifications

security concerns. Therefore, another relevant aspect of the testbed is its governance. In the interest of observing the traffic in the private network, the switch had to be configured in promiscuous mode [61] in one of the ports. Basically, the Sniffer connected to this configured port and could sniff all the traffic in the network, i.e. all ingress and egress packets in the switch.

4.1.3. Proxmox set up

In pursuance of having a controlled virtual environment, Proxmox was installed as the Operating System (OS) of the machines that would host the virtualized servers. The installation was pretty straightforward since the only input required was offering an IP address to the machine which, of course, had to be under the network of the testbed.

Once the nodes had Proxmox installed, both had to be configured as a cluster in order to be able to perform VMs and LXC migrations [62]. It was really important to configure the nodes properly to avoid having independent entities and attain having a single virtualization environment.

4.1.4. KVM and LXC preparation

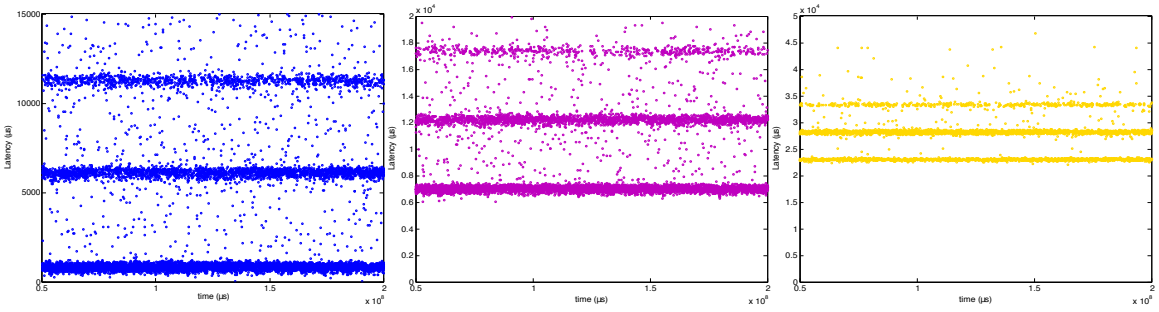
In order to perform the experiments presented in section 4, it was needed to set up the KMS and the application server. The videoconference service had to be run either in a VM or LXC and for this reason, the procedure described below was performed twice, once for each virtualization technology.

Firstly, in order to run a KMS in a virtualized environment, it had to be installed on this isolated machine following the instructions in [43]. For the sake of improving the security of the service, a certificate was created and added to the server following the recommendations in this instructions. Thanks to this last modification, the communications with the KMS were through WebSocket Secure (WSS) instead of WebSocket (WS).

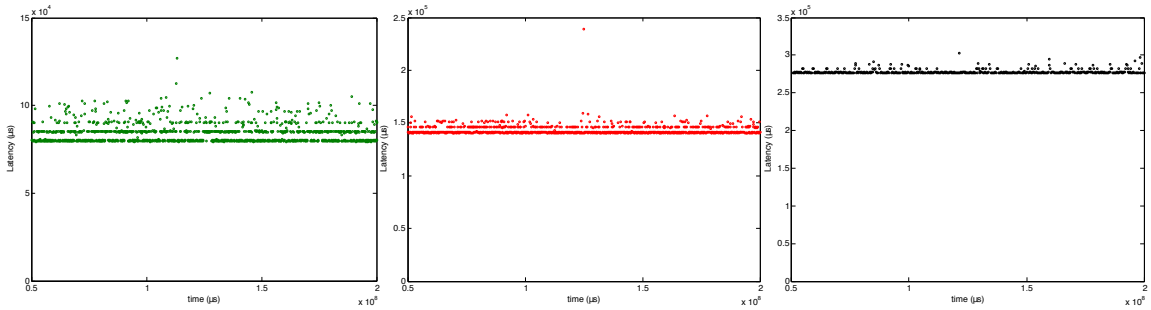
Secondly, the application server was configured on another isolated machine. The videoconference application was gathered from Kurento room projects [63], in particular, *kurento-room-demo* was cloned by means of git. In order to be able to compile the project, Java and Maven [64] had to be installed. Aside from this installation, a self-signed certificate had to be created and specified in *application.properties* file. Finally, the configuration required was as simple as adding in the *kurento-room-demo.conf.json* file the KMSs in use.

4.2. Edge cloud vs different locations

The results in Figure 16 demonstrate the huge difference of hosting the KMS in a close or far datacentre. Obviously, the location with the best performance in terms of latency is, as expected, the edge. Therefore, these plots analytically demonstrate that placing the KMS as close to the user as possible can provide significant benefits with regard to latency.



a) Edge b) London c) Frankfurt

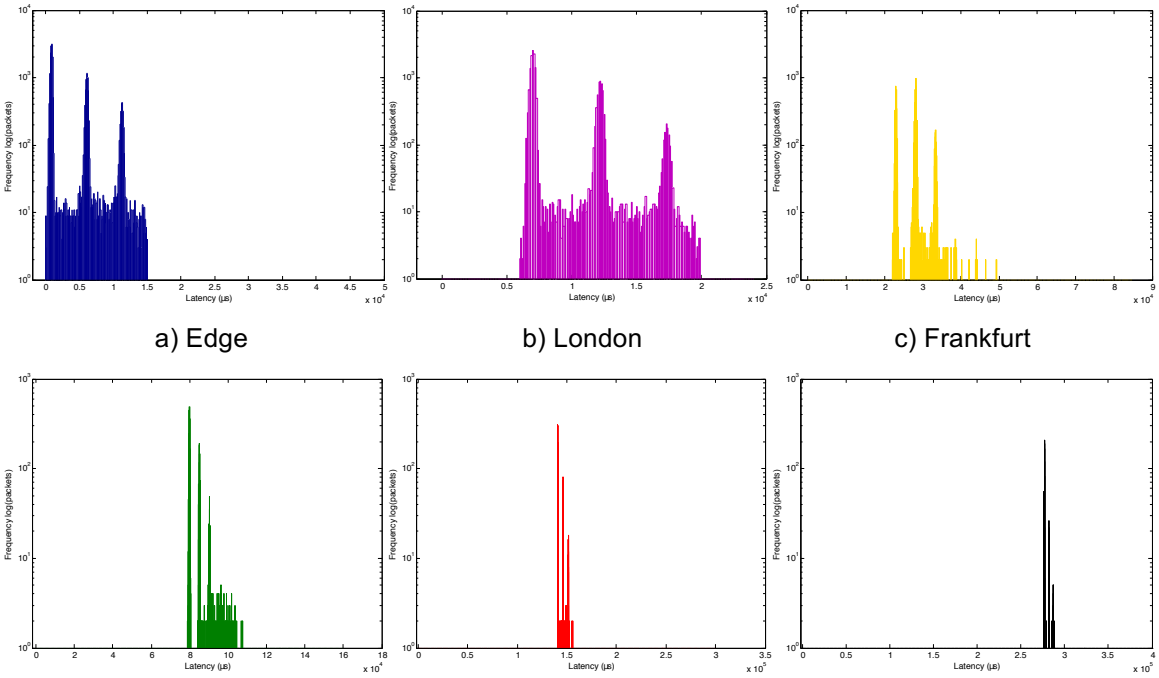


c) Virginia d) Oregon e) Tokyo

Figure 16 – Latency for different locations

In behalf of studying the latency in detail, the latency obtained for the edge case is further analysed, i.e. Figure 16.a. The observed latency is the overall end-to-end latency, i.e. the latency from Client A to Client B. Since the KMS is in the middle of this communication, the

latency is composed of the network latency plus the KMS processing latency. The former is the latency suffered by packets just for the fact of travelling through the network, while the latter corresponds to the time the KMS needs to process and forward these packets to the destination. Notice that in this figure, despite the dispersion in the values, three main latency values can be clearly identified. If we take a look at the logarithmic histograms in Figure 17, these three values are manifested as well. This pattern remains wherever the KMS is located. Of course, these peaks are shifted depending on the location since there is more latency as long as the KMS is moved away from the user.



a) Edge b) London c) Frankfurt

c) Virginia d) Oregon e) Tokyo

Figure 17 – Latency logarithmic histogram for different locations

By this means, it can be concluded that when the KMS processes the packets introduces three kinds of latencies depending on the task that it has to perform with them.

The analysis of the latency in function of the distance was performed with VMs but with LXC as well. In this case, the difference is almost inexistent since the only possible impact is the performance of the KMS when uses one technology or the other. The ecdf of the latency for each technology is depicted in Figure 18. The semblance between the figures is obvious.

An interesting outcome is the fact that the percentage of packets of each process might depend on the latency suffered. In that sense, the KMS behaves differently depending on when the packets arrive at it. Nonetheless, the scope of this project is not studying the specific behaviour of this particular media server but the overall comparison of different real locations. The study of the behaviour of the KMS regarding the latency for a wider range of latencies could open a future line of research.

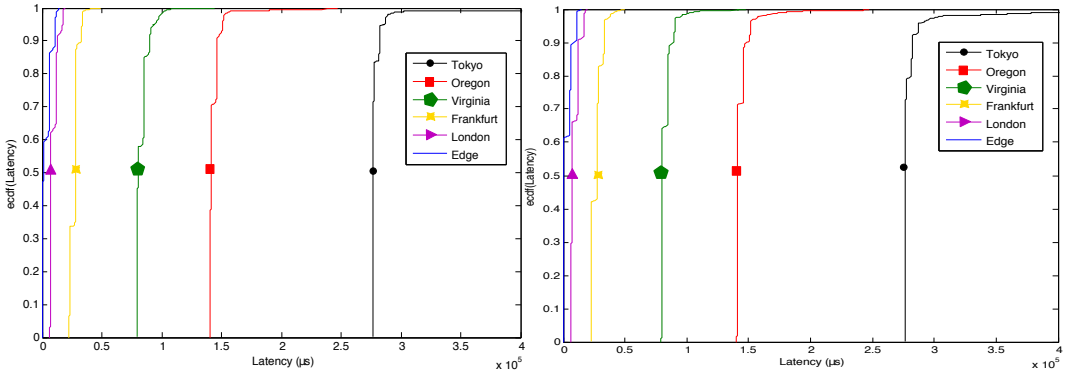


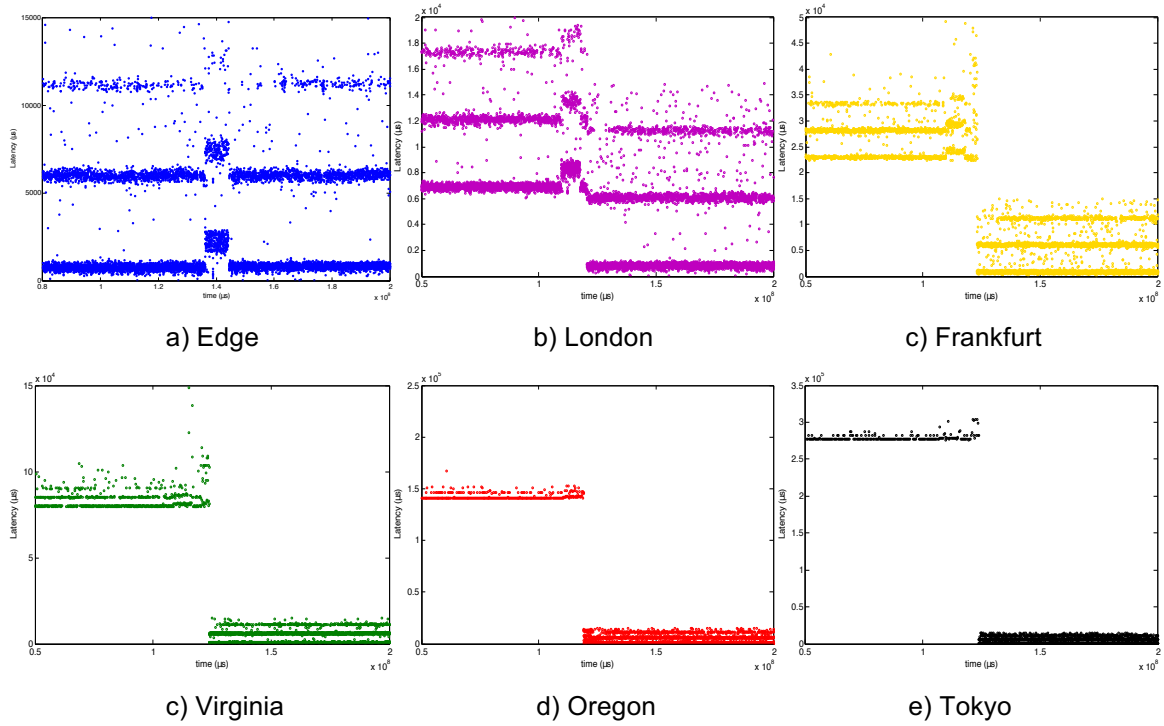
Figure 18 – Latency ecdf for all locations

4.3. Impact of Live migration

With regard to live migration of the KMS, the analysis can be clearly divided into two parts depending on the technology in use: VMs or LXC. In both cases, it is must be highlighted that the KMS was always migrated during an on-going session from each location to the edge.

The results for the VM are shown in Figure 19 where the migration exercise can be spotted at a glance. The migration from any location to the edge show the same phenomena, even when migrating within the same location, i.e. edge to edge. There is an increment in the latency during the migration process. Therefore, it can be immediately concluded that actually, there is an impact in terms of QoS. Nevertheless, the overall reduction of the latency is obvious when migrating from a further location to the edge. This should motivate the migration from the cloud to the edge even when the session is running and affects the latency. In the case of edge to edge migration, this study has analysed the feasibility of providing a videoconference service that is migrated as long as the user moves within the office. For instance, if the user moves from one part of the office where one of the nodes is placed to a new area with a closer node, it would make sense migrating the server to this new node. As seen, there will be an impact during the migration process but at the end of it, the videoconference service will be provided with the lowest latency possible.

In the interest of analysing the results further, one of the migration has been taken as an example in Figure 20, particularly the migration from London to the edge. In this figure, three phases are highlighted in the timeline. The first one is the latency when the KMS is placed in London, the second one is the latency during the migration process and, the last one corresponds to the latency once the KMS is in the edge. Using the indexes provided in the plot, i.e. A, B and C, the evolution of the latency is matched with A', B' and C'. These three values for the latency pattern was previously observed and analysed in Figure 16. The relevance of Figure 19 remains in observing how all of them are reduced by means of approaching the server to the user, i.e. migrating from the cloud to the edge. However, and as mentioned above, there is an impact in the latency during the migration time that cannot be ignored. It is clearly seen how during the migration time the latency increments during the process. Therefore, the slower the migration is the more worrying this concern is.



a) Edge b) London c) Frankfurt

c) Virginia d) Oregon e) Tokyo

Figure 19 – VM live migration latency for different locations

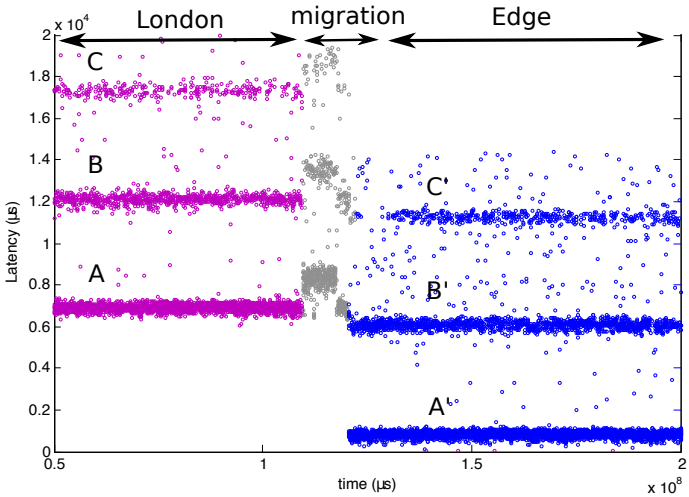
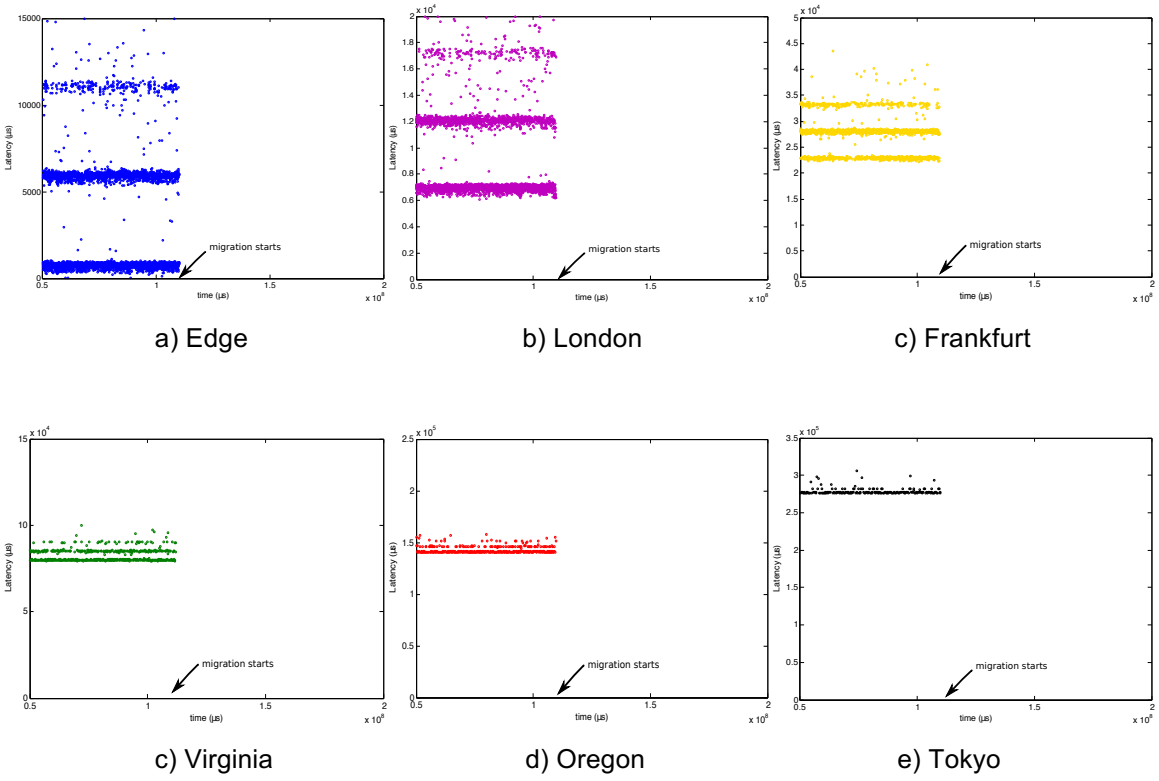


Figure 20 – VM Live migration London to Edge latency

Reducing the migration time depends on many factors, but especially on the size of the machine being migrated and the available bandwidth in the network. If the migration time can be reduced, the observed impact can be diminished. This also highlights the importance of the network configuration. In the testbed, these migrations were performed on the data plane because each machine unfortunately just provided one interface. However, in a production scenario that properly follows the SDN paradigm, a dedicated interface for the control plane must be provided. Then, the migration could be performed without impacting the data plane at all, where the videoconference packets are.

Anyhow, it is known beforehand that the migration of VMs is slower than LXC because of their nature [65]. Consequently, it was very interesting studying both cases in detail. Notwithstanding, the results for LXC live migration are not hopeful, see Figure 21. This figure shows how the latency could just be computed before the migration started. In other words, LXC live migration did not succeed. At the time of this study was performed, the environment described in section 4.1. offered LXC migration but not live. Nowadays, the live migration of LXC is being developed and hopefully, in the near future, this feature will be stable enough to be included in virtualization environments such as Proxmox. In order to follow the development of live LXC migration, consult CRIU [66], the current solution taking the lead.



a) Edge b) London c) Frankfurt

c) Virginia d) Oregon e) Tokyo

Figure 21 – LXC live migration latency for different locations

Despite the technology limitation exposed above, a further analysis had to be done to deeply understand the issue. Since the videoconference session was lost when the migration started, a parallel process had to be run to check the status of the server during the migration. A simple approach was pinging the server throughout all the experiment. In Figure 22, the latency for these pings is shown jointly with the videoconference data packets latency. This figure mainly shows how the server stopped replying pings when the migration started. Eventually, the KMS started replying the pings again but effectively, the KMS was shut down. Consequently, there is a noticeable migration downtime that concludes that the migration was not live. In other words, the KMS was shut down, migrated

and started in the new location. Certainly, this procedure was the cause of losing the videoconference sessions.

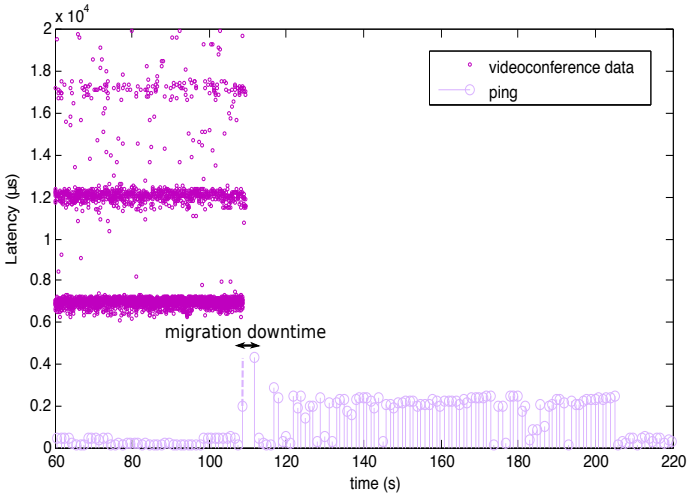


Figure 22 – LXC Live migration London to Edge latency

4.4. Available bandwidth and packet loss

The experience perceived as the videoconference user when the available bandwidth was changed is shown in Table 6. Notice that the focus of this project is not the Quality of Experience (QoE), therefore the information provided in this table was not obtained from multiple samples but from the experience of just one user. At any rate, this analysis was performed in view of knowing the performance of the videoconference service under study and providing a reference to the experience.

Available Bandwidth	QoE perception
4.8 Mbps	Very good
4 Mbps	Good
3.43 Mbps	Noticeable lag
3 Mbps	Unacceptable lag
2.67 Mbps	Unacceptable lag
2.4 Mbps	Unacceptable lag

Table 6 – User perception depending on the available bandwidth

When the available bandwidth was reduced along the session, an interesting outcome was obtained: the session suffered lag when the available bandwidth was reduced. In contemplation of this awareness, the latency for this session was computed, see Figure 23. In this figure, it can be observed how the dispersion of the latency values incremented as long as the session elapsed. In brief, the less bandwidth was available for the videoconference the more probable of obtaining higher latency values.

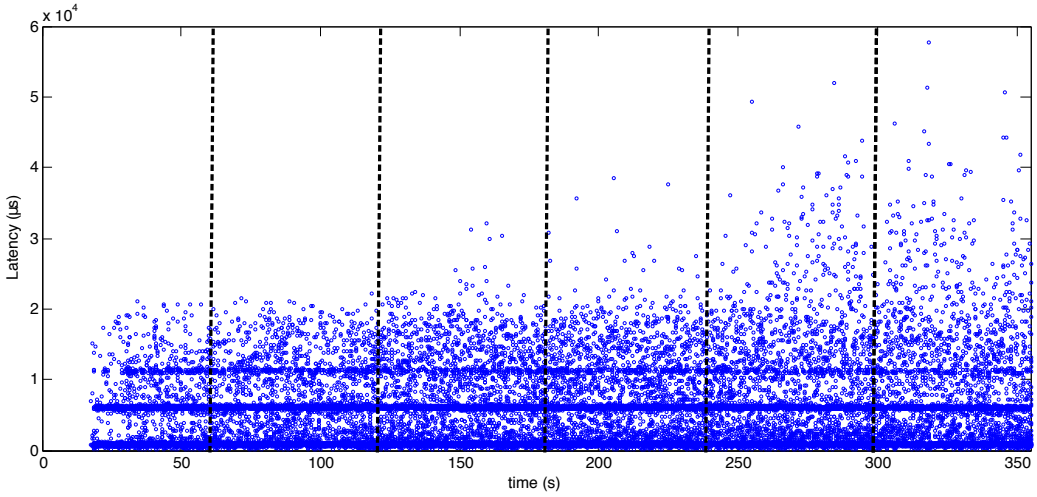


Figure 23 – Latency evolution depending on the available bandwidth

The reason for this increment in the latency can be explained with the help of Figure 24. In this figure, the overall bandwidth consumed by the videoconference was computed every 1000 packets and every 100 packets. In this conditions, the evolution of the average bandwidth and the instant bandwidth is represented. It can be observed how the average bandwidth is always below the available bandwidth and how the instant bandwidth fluctuates around the average. Theoretically, the application had the required bandwidth available in the whole session, on the worse occasion 2.4Mbps were available, i.e. the videoconference occupied 100% of the available bandwidth. Nonetheless, as long as the bandwidth occupation of the videoconference represented a higher percentage of the total available bandwidth, the experience worsened. Essentially, when the available bandwidth was closer and closer to the average bandwidth, there was more certainty that at some point the instant bandwidth would stand out of the available bandwidth and this would provoke a bad experience to the user.

As a summary, the results of this first analysis pretends to remark how important is quantifying the real resources needed when setting up the system. In this particular case, a minimum of 4 Mbps bandwidth must be available to offer a good service.

Regarding the packet loss analysis, it could be observed that the videoconference service could handle the packet loss particularly better. In Table 7, it is shown how the packet loss was not noticed until losing 15% of the packets. This means that up to this point, the application was able to handle losses. Naturally, the videoconference data was sent through UDP packets, therefore the recovery of lost packets had necessarily to be done at the application level. In either way, the analysis of the application itself was not under the scope of this project. To conclude this last analysis, a set of pictures for different packet loss ratio are displayed in Figure 25.

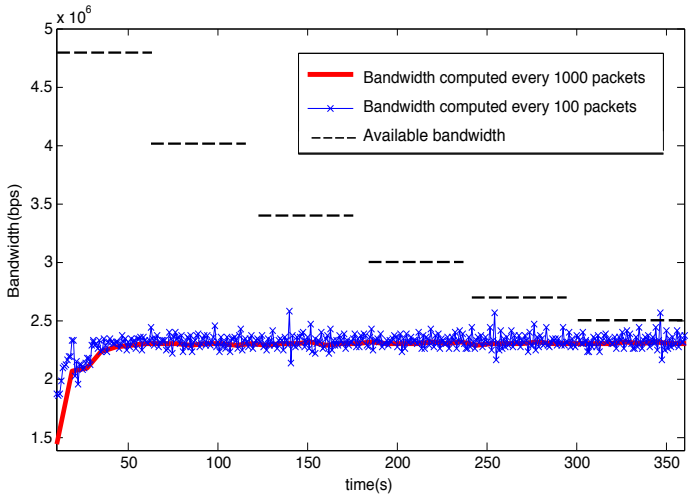
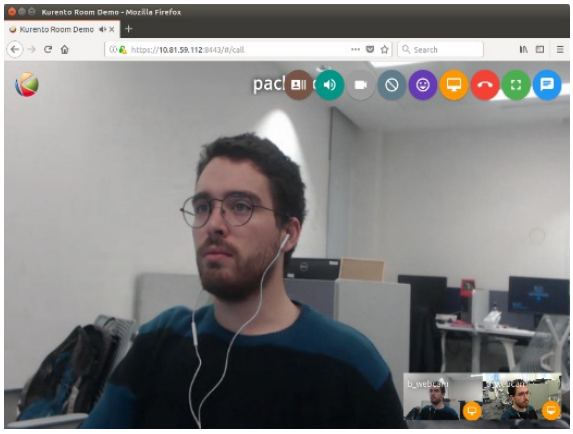


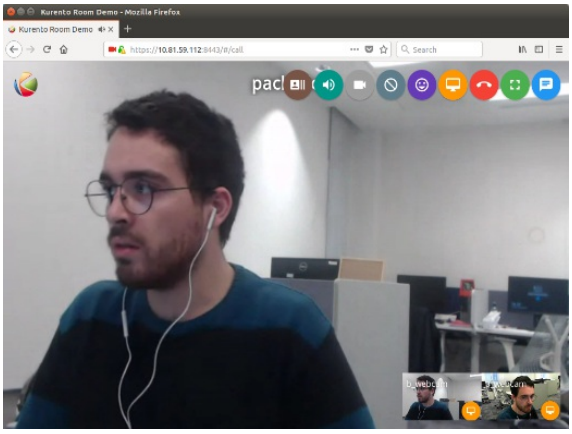
Figure 24 – Available bandwidth

Packet loss	QoE perception
0%	Very good
5%	Very good
10%	Good
15%	Acceptable
20%	Noticeable but annoying
25%	Noticeable but annoying
30%	Blurry image
35%	Blurry image
40%	Unacceptable

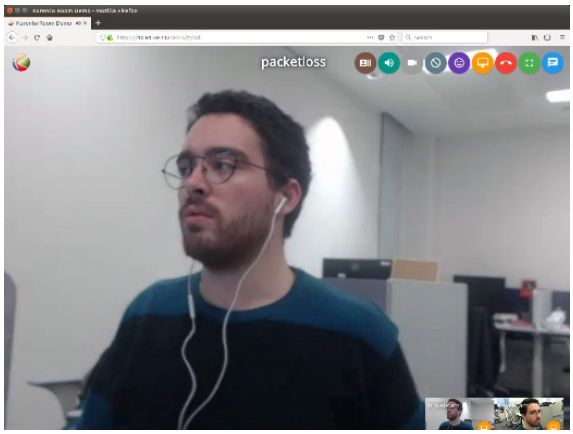
Table 7 – User perception depending on the packet loss



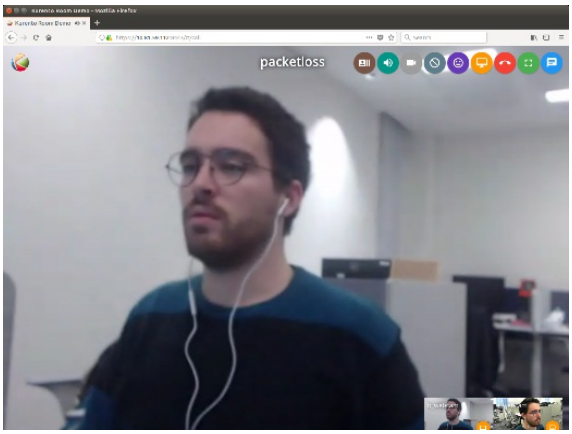
10% packet loss



20% packet loss



30% packet loss



40% packet loss

Figure 25 – Packet loss pictures

5. Budget

The budget presented below makes reference to the implementation of the testbed presented in section 4.1. Proposing a budget for a final production environment would result very erratic since it would highly depend on the scenario. Therefore, do not consider this budget as a proposal for a production environment.

The approximated budget to deploy a laboratory where to run the experiments exposed in section 4 is detailed in Table 8. Notice that the hardware exposed in this table can be substituted with similar devices that might be cheaper. Consequently, the total amount required is, of course, an approximation. Moreover, the setup used for this project did not need to purchase any of these items since the university could offer them.

Item	Price per unit (€)	Quantity	Total
Switch	750	1	750
Proxmox nodes	800	2	1600
Sniffer	600	1	600
Client PCs	700	2	1400
Proxmox GUI (PC)	700	1	700
Ethernet wires	6	6	36
Matlab license	800	1	800
		TOTAL	
		5886€	

Table 8 – Testbed budget

As proven, the implementation of the testbed is simple and economical in comparison to deploying a real service in a data centre. One one hand, the hardware is not especially demanding, see the specifications in Table 5. On the other hand, the virtualization environment, Proxmox, and the videoconference application, Kurento, both are open-source software and therefore, free. The only license to add to the budget is Matlab, used for post-processing data purposes.

6. Conclusions and future development

The aim of this dissertation has been to deliver a POC that motivates the implementation of an Edge cloud computing and SDN based solution, especially, for improving the quality of low sensitive and bandwidth exhaustive applications in enterprise environments. For this reason, firstly, an Edge network model has been presented and detailed. This model has been proven to be industry-accepted and its most exciting facet is having the ability to easily change the decision criterion depending on the business case.

The videoconference application used to perform the quality analysis was primarily evaluated. This exercise enabled understanding the traffic generated by this particular application which resulted to be burst based. Detecting this pattern was crucial at the time of proposing a methodology to compute the latency. In fact, this methodology had to be implemented so as to proceed with the quality analysis afterwards.

Essentially, the first experiment demonstrated how the latency could be significantly reduced by means of bringing the servers to the edge. Moreover, the second experiment proved the feasibility of having a dynamic environment where the service can be migrated. At the same time, this experiment highlighted the importance of following the SDN paradigm, where two different planes must be used for control and data purposes. And lastly, for the sake of providing a complete quality analysis, the impact of available bandwidth and packet loss in the edge was commented. One one hand, it was remarked the importance of having an extra margin of available bandwidth reserved for this kind of applications in enterprises infrastructures. It was observed how the lack of bandwidth incremented the latency, ruining the effort of bringing the servers to the cloud. On the other hand, it could be observed that the particular application under study behaved pretty good handling packet loss.

Ultimately, all of these tests together showed the capabilities of Edge cloud computing to improve the videoconference experience. Notwithstanding, several future milestones have been left for the future. The next step would be implementing a proper SDN module that could be integrated with an SDN controller. This would enable moving towards a real deployment in production analogous to the testbed presented in this work. In this case, it would be compulsory the disposal of enough interfaces to use different planes for control and data. Aside, it would be compelling the use of an advanced videoconference application that makes use of AR. In other words, an application that demands even more bandwidth for its correct functioning.

References

[1] Microsoft, “Microsoft hololens,” https://www.microsoft.com/en-us/hololens, Jan. 2018

[2] K.Minolta,“Workplacehub,”hhttp://workplacehub.konicaminolta.com/, Jan. 2018.

[3] M. Avram, “Advantages and challenges of adopting Cloud computing from an enterprise perspective,” *Procedia Technology*, vol. 12, pp. 529 – 534, 2014, the 7th International Conference Interdisciplinarity in Engineering, INTER-ENG 2013, 10-11 October 2013, Petru Maior University of Tirgu Mures, Romania. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S221201731300710X

[4] N. Antonopoulos and L. Gillam, *Cloud Computing: Principles, Systems and Applications*, ser. Computer Communications and Networks. Springer International Publishing, 2017. [Online]. Available: https://books.google.co.uk/books?id=ZmsmDwAAQBAJ

[5] Z. Xiao, Q. Chen, and H. Luo, “Automatic scaling of internet applications for Cloud computing services,” *IEEE Transactions on Computers*, vol. 63, no. 5, pp. 1111–1123, May 2014.

[6] S. Dhuria, A. Gupta, and R. Singla, “Review of pricing techniques in Cloud computing,” *International Journal of Advanced Research in Science and Engineering, 6 (10)*, pp. 1683–1690, 2017.

[7] L. Wang, R. Ranjan, J. Chen, and B. Benatallah, *Cloud Computing: Methodology, Systems, and Applications*. CRC Press, 2017. [Online]. Available: https://books.google.co.uk/books?id=8UbNBQAAQBAJ

[8] F. Sabahi, “Cloud computing reliability, availability and serviceability (ras): Issues and challenges,” *International Journal on Advances in ICT for Emerging Regions (ICTer)*, vol. 4, no. 2, 2012.

[9] R. Kalaiprasath, R. Elankavi, D. R. Udayakumar *et al.*, “Cloud. security and compliance - A semantic approach in end to end security,” *International Journal Of Mechanical Engineering And Technology (Ijmet)*, vol. 8, no. 5, 2017.

[10] O. Osanaiye, S. Chen, Z. Yan, R. Lu, K. Choo, and M. Dlodlo, “From Cloud to Fog computing: A review and a conceptual live VM migration framework,” *IEEE Access*, 2017.

[11] P. S. Khodashenas, C. Ruiz, M. S. Siddiqui, A. Betzler, and J. F. Riera, “The role of Edge computing in future 5G mobile networks: concept and challenges,” *Cloud and Fog Computing in 5G Mobile Networks*, p. 349, 2017.

[12] L. M. Vaquero and L. Rodero-Merino, “Finding your way in the fog: Towards a comprehensive definition of Fog computing,” *SIGCOMM Comput. Commun. Rev.*, vol. 44, no. 5, pp. 27–32, Oct. 2014. [Online]. Available: http://doi.acm.org/10.1145/2677046.2677052

[13] T. H. Luan, L. Gao, Z. Li, Y. Xiang, G. Wei, and L. Sun, “Fog computing: Focusing on mobile users at the edge,” *arXiv preprint arXiv:1502.01815*, 2015.

[14] P. J. Braun, S. Pandi, R. S. Schmoll, and F. H. P. Fitzek, “On the study and deployment of Mobile Edge Cloud for tactile internet using a 5G gaming application,” in *2017 14th IEEE Annual Consumer Communications Networking Conference (CCNC)*, Jan 2017, pp. 154–159.

[15] D. C. Klonoff, “Fog computing and Edge computing architectures for processing data from diabetes devices connected to the medical Internet of Things,” 2017.

[16] F. Bonomi, R. Milito, J. Zhu, and S. Addepalli, “Fog computing and its role in the Internet of Things,” in *Proceedings of the first edition of the MCC workshop on Mobile Cloud computing*. ACM, 2012, pp. 13–16.

[17] C. S. Magurawalage, K. Yang, and K. Wang, “Aqua computing: Coupling computing and communications,” *arXiv preprint arXiv:1510.07250*, 2015.

[18] M. Femminella, M. Pergolesi, and G. Reali, “Performance evaluation of Edge Cloud computing system for Big data applications,” in *2016 5th IEEE International Conference on Cloud Networking (Cloudnet)*, Oct 2016, pp. 170–175.

[19] L. Malhotra, D. Agarwal, and A. Jaiswal, “Virtualization in Cloud computing,” *J. Inform. Tech. Softw. Eng*, vol. 4, no. 2, 2014.

[20] G. P. Sharma, S. Singh, A. Singh, and R. Kaur, “Virtualization in Cloud computing,” 2016.

[21] P. VE, "Proxmox virtual environment wiki," 2017, [Online; accessed 17-January-2018]. [Online]. Available: https://pve.proxmox. com/mediawiki/index.php?title=Main Page&oldid=10007

[22] M. Raho, A. Spyridakis, M. Paolino, and D. Raho, "Kvm, Xen and Docker: A performance analysis for arm based NFV and Cloud computing," in *2015 IEEE 3rd Workshop on Advances in Information, Electronic and Electrical Engineering (AIEEE)*, Nov 2015, pp. 1–8.

[23] D.Bernstein,"Containers and Cloud: From LXC to Docker to Kubernetes," *IEEE Cloud Computing*, vol. 1, no. 3, pp. 81–84, Sept 2014.

[24] C.PahlandB.Lee,"Containers and clusters for Edge Cloud architectures – a technology review," in *2015 3rd International Conference on Future Internet of Things and Cloud*, Aug 2015, pp. 379–386.

[25] R. Jain and S. Paul, "Network virtualization and Software Defined Networking for Cloud computing: a survey," *IEEE Communications Magazine*, vol. 51, no. 11, pp. 24–31, November 2013.

[26] B. Han, V. Gopalakrishnan, L. Ji, and S. Lee, "Network Function Virtualization: Challenges and opportunities for innovations," *IEEE Communications Magazine*, vol. 53, no. 2, pp. 90–97, Feb 2015.

[27] D. Kreutz, F. M. V. Ramos, P. E. Verssimo, C. E. Rothenberg, S. Azodol- molky, and S. Uhlig, "Software-Defined Networking: A comprehensive survey," *Proceedings of the IEEE*, vol. 103, no. 1, pp. 14–76, Jan 2015.

[28] Q. Chen, F. R. Yu, T. Huang, R. Xie, J. Liu, and Y. Liu, "An integrated framework for Software Defined networking, caching, and computing," *IEEE Network*, vol. 31, no. 3, pp. 46–55, May 2017.

[29] N. McKeown, T. Anderson, H. Balakrishnan, G. Parulkar, L. Peterson, J. Rexford, S. Shenker, and J. Turner, "Openflow: Enabling innovation in campus networks," *SIGCOMM Comput. Commun. Rev.*, vol. 38, no. 2, pp. 69–74, Mar. 2008. [Online]. Available: http://doi.acm.org/10.1145/1355734.1355746

[30] R. Guerzoni, R. Trivisonno, and D. Soldani, "SDN-based architecture and procedures for 5g networks," in *1st International Conference on 5G for Ubiquitous Connectivity*, Nov 2014, pp. 209–214.

[31] N. Omnes, M. Bouillon, G. Fromentoux, and O. L. Grand, "A programmable and virtualized network it infrastructure for the Internet of Things: How can nfv sdn help for facing the upcoming challenges," in *2015 18th International Conference on Intelligence in Next Generation Networks*, Feb 2015, pp. 64–69.

[32] J. Costa-Requena, J. L. Santos, V. F. Guasch, K. Ahokas, G. Premsankar, S. Luukkainen, O. L. Prez, M. U. Itzazelaia, I. Ahmad, M. Liyanage, M. Ylianttila, and E. M. de Oca, "Sdn and nfv integration in gener- alized mobile network architecture," in *2015 European Conference on Networks and Communications (EuCNC)*, June 2015, pp. 154–158.

[33] R. Vilalta, I. Popescu, A. Mayoral, X. Cao, R. Casellas, N. Yoshikane, R. Martnez, T. Tsuritani, I. Morita, and R. Muoz, "End-to-end SDN/NFV orchestration of video analytics using Edge and Cloud computing over programmable optical networks," in *2017 Optical Fiber Communications Conference and Exhibition (OFC)*, March 2017, pp. 1–3.

[34] A. C. Baktir, A. Ozgovde, and C. Ersoy, "How can Edge computing benefit from Software-Defined Networking: A survey, use cases, and future directions," *IEEE Communications Surveys Tutorials*, vol. 19, no. 4, pp. 2359–2391, Fourthquarter 2017.

[35] T. Taleb, P. Hasselmeyer, and F. G. Mir, "Follow-me cloud: An Openflow-based implementation," in *2013 IEEE International Confer- ence on Green Computing and Communications and IEEE Internet of Things and IEEE Cyber, Physical and Social Computing*, Aug 2013, pp. 240–245.

[36] Y. Li, H. Wang, M. Liu, B. Zhang, and H. Mao, "Software Defined Networking for distributed mobility management," in *2013 IEEE Globecom Workshops (GC Wkshps)*, Dec 2013, pp. 885–889.

[37] L.L.Fernández,M.P.Díaz,R.B.Mejías,F.J.López,and J.A.Santos, "Kurento: a media server technology for convergent www/mobile real-time multimedia communications supporting WebRTC," in *World of Wireless, Mobile and Multimedia Networks (WoWMoM), 2013 IEEE 14th International Symposium and Workshops on a*. IEEE, 2013, pp. 1–6.

[38] Lynckia, "Licode documentation," http://licode.readthedocs.io/en/stable/, 2017.

[39] B. Garcia, L. Lopez-Fernandez, M. Gallego, and F. Gortazar, "Kurento: The swiss army knife of WebRTC media servers," *IEEE Communications Standards Magazine*, vol. 1, no. 2, pp. 44–51, 2017.

[40] B. Garca, M. Gallego, L. López, G. A. Carella, and A. Cheambe, "Nubomedia: An elastic PaaS enabling the convergence of real-time and Big data multimedia," in *2016 IEEE International Conference on Smart Cloud (SmartCloud)*, Nov 2016, pp. 45–56.

[41] O. Sefraoui, M. Aissaoui, and M. Eleuldj, "Openstack: toward an open-source solution for Cloud computing," *International Journal of Computer Applications*, vol. 55, no. 3, 2012.

[42] L. López, M. París, S. Carot, B. García, M. Gallego, F. Gortázar, R. Benítez, J. A. Santos, D. Fernández, R. T. Vlad, I. Gracia, and F. J. López, "Kurento: The WebRTC modular media server," in *Proceedings of the 2016 ACM on Multimedia Conference*, ser. MM '16. New York, NY, USA: ACM, 2016, pp. 1187–1191. [Online]. Available: http://doi.acm.org/10.1145/2964284.2973798

[43] Kurento.org, "Kurento documentation," https://media.readthedocs.org/pdf/doc-kurento/stable/doc kurento.pdf, May 2017.

[44] Kurento.org, "Kurento architecture," http://doc-kurento.readthedocs.io/en/ latest/mastering/kurento architecture.html, May 2017.

[45 Nir Yechiel, "SDN with Red Hat OpenStack platform: Opendaylight integration," https://redhatstackblog.redhat.com/2017/02/ 28/sdn-with-red-hat-openstack-platform-opendaylight-integration/, Feb. 2017.

[46] A.Mayoral,R.Vilalta,R.Muoz,R.Casellas,R.Martnez,andJ.Vlchez, "Integrated it and network orchestration using OpenStack, OpenDaylight and active stateful pce for intra and inter data center connectivity," in *2014 The European Conference on Optical Communication (ECOC)*, Sept 2014, pp. 1–3.

[47] H. Le *et al.*, "OpenStack and Software-Defined Networking: The enormous potential of open-source software collaboration," 2017.

[48] O. Tkachova, M. J. Salim, and A. R. Yahya, "An analysis of SDN-OpenStack integration," in *2015 Second International Scientific-Practical Conference Problems of Infocommunications Science and Technology (PIC S T)*, Oct 2015, pp. 60–62.

[49] J. Medved, R. Varga, A. Tkacik, and K. Gray, "OpenDaylight: Towards a model-driven SDN controller architecture," in *Proceeding of IEEE International Symposium on a World of Wireless, Mobile and Multimedia Networks 2014*, June 2014, pp. 1–6.

[50] L. Sequeira, J. L. de la Cruz, J. Ruiz-Mas, J. Saldana, J. Fernandez- Navajas, and J. Almodovar, "Building an SDN enterprise wlan based on virtual aps," *IEEE Communications Letters*, vol. 21, no. 2, pp. 374–377, 2017.

[51] L. Sequeira, J. Ferna´ndez-Navajas, and J. Saldana, "The effect of the buffer size in qos for multimedia and bursty traffic: When an upgrade becomes a downgrade," *KSII Transactions on Internet and Information Systems (TIIS)*, vol. 8, no. 9, pp. 3159–3176, 2014.

[52] Ubuntu, "htop - ubuntu manuals," http://manpages.ubuntu.com/manpages/zesty/man1/htop.1.html, Dec. 2010.

[53] Ubuntu,"bmon-ubuntumanuals,"http://manpages.ubuntu.com/manpages/ precise/man1/bmon.1.html, Dec. 2010.

[54] T. Levent-Levi, "How many sessions can a Kurento server hold?" https: //testrtc.com/sessions-kurento-server/, Sep. 2017.

[55] Microsoft, "How much bandwidth does Skype need?" https://support. skype.com/en/faq/FA1417/how-much-bandwidth-does-skype-need, Jan. 2018.

[56] Mathworks, "Empirical cumulative distribution function - Matlab documentation," https://uk.mathworks.com/help/stats/ecdf.html, Jan. 2018.

[57] K. Schittkowski, "Easy-fit: a software system for data fitting in dynamical systems," *Structural and Multidisciplinary Optimization*, vol. 23, no. 2, pp. 153–169, Mar 2002. [Online]. Available: https://doi.org/10.1007/s00158- 002- 0174- 6

[58] E. Garsva, N. Paulauskas, G. Grazulevicius, and L. Gulbinovic, "Packet inter-arrival time distribution in academic computer network," *Elektron- ika ir Elektrotechnika*, vol. 20, no. 3, pp. 87–90, 2014.

[59] T. L. Foundation, “netem and tc - linux man,” https://wiki. linuxfoundation.org/networking/netem, Jul. 2016.

[60] Amazon, “Amazon Web Services,” https://aws.amazon.com/es/, Jan. 2018.

[61] C. A. Headquarters, “Catalyst 2960 switch software configuration

guide,” https://www.cisco.com/c/en/us/td/docs/switches/lan/ catalyst2960/software/release/12-2 40 se/configuration/guide/scg.pdf, Sep. 2007.

[62] P. VE”, “Proxmox ve 4.x cluster,” https://pve.proxmox.com/wiki/ Proxmox VE 4.x Cluster, Jan. 2017.

[63] Kurento.org, “Kurento room documentation,” https://media.readthedocs. org/pdf/doc-kurento-room/latest/doc-kurento-room.pdf, Sep. 2017.

[64] A. M. Project, “Welcome to Apache Maven,” https://maven.apache.org/, Jan. 2018.

[65] C.Spoiala,A.Calinciuc,C.Turcu,andC.Filote,“Performance comparison of a WebRTC server on Docker versus virtual machine,” pp. 295–298, 05 2016.

[66] O. team at Virtuozzo, “Checkpoint/restore in userspace wiki,” https:// criu.org/Main Page, Dec. 2017.

[67] Van Jacobson and Steven McCanne, “tcpdump - manual”, https://www.tcpdump.org/tcpdump_man.html, Feb. 2017.

[68] Wireshark, “tshark - manual”, https://www.wireshark.org/docs/man-pages/tshark.html, Dec. 2010.

Appendix 1: SDN module example

The pseudo code available in this Appendix pretends to show an example of one Algorithm criteria seen in Figure 4. Notice that the aim is not providing a real ODL module but to depict the essence of the logic presented in section 3.1 with an easy example.

In this example, the module basically decides in which node the KMS has to be hosted depending on the demand in the network, i.e. the number of users. Each of the zones is physically identified by means of different subnetworks. This means that we would have a distribution of IPs as seen in Table 9.

Zone 1	Zone 2	Zone 3
10.81.59.X	10.81.60.X	10.81.61.X

Table 9 – IP networks distribution

The SDN module checks periodically the database, which is systematically updated thanks to SDN, and if it detects that there are more users requesting service in another zone rather the one where the KMS is hosted, the SDN controller will trigger the migration, as illustrated in Figure 26.

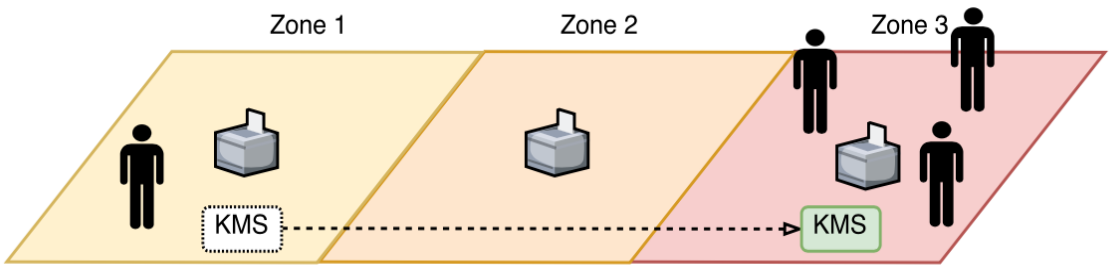


Figure 26 – User zones

The logic of this module is presented below in code format. It has been written in Java for being ODL compliant. In brief, the code updates a Map where stores the number of users per location and requests to migrate the KMS to the node with more demand if necessary.

```
 // This pseudocode showcases the logic behind the proposed OpenDaylight module
// Every x seconds, this module will recieve as an input:
//             - A .csv with all the packets captured in the network.
//             This .csv passed as an input contains the following four columns:
timestamp, IP source, IP destination and packet size
//             - The current server IP address
// The server will always have the same last octet in the IP@: X.X.X.111
```

```
public class odlEdgemodule{

	public Map<String,Integer> userMap = null; // This map will contain the number
users per subnet
	public String csvFile;
	public String serverIP;

	public odlEdgemodule(String captureCsv, String currentServerIP){
		userMap =  new Map<String,Integer>();
		csvFile = captureCsv;
	    serverIP = currentServerIP;
	}



    public static void main(String[] args) throws Exception{
		updateUserMap();
		Map.Entry maxEntry = null;
		for (Map.Entry<String,Integer> entry : userMap.entrySet())
		{
			if (maxEntry == null ||
entry.getValue().compareTo(maxEntry.getValue()) > 0)
		{
			maxEntry = entry; // Subnet with highest number of users
		}
		}

		String[] number = serverIP.split(".");
	String sn = number[3];

		if(!maxEntry.getKey().equals(sn)){ // Just in case the current location
has not the highest number of users then, migrate.
			String [] newServerIP = new String[4];
			newServerIP[1] = number[1];
			newServerIP[2] = number[2];
			newServerIP[3] = maxEntry.getKey();
			newServerIP[4] = "111";
			serverIP = newServerIP;
			migrate(newServerIP);
		}
	}

	/*This method update the map with the number of users in each subnet*/
	public void updateUserMap(){

		Set<Integer> ipSet = ipSet = new HashSet<Integer>();
		ipSet = getIps(this.csvFile);
		Set<Integer> subnetSet = new HashSet<Integer>();
		Iterator<Integer> iterator = ipSet.iterator();

		while (iterator.hasNext()) { // We want to identify the list of subnets

			String[] number = iterator.next().split(".");
			// Since subnetSet is a Set variable only unique values will be
added in each iteration
			subnetSet.add(number[3]);

		if (number[4].equals("111")) { // Additionally, when we detect the
server IP we remove it from the list from users
			iterator.remove();
		}
```

```
                }

                iterator = subnetSet.listIterator(); //reset the iterator

                while (iterator.hasNext()) { // Once the subnet Set is ready we can put
them on the Map
                        userMap.put(iterator.next());
                }

                iterator = ipSet.listIterator(); //reset the iterator

                while (iterator.hasNext()) { // Finally we want count the number of
users in each subnet and store it in the Map
                String[] number = iterator.next().split(".");
                String subnet = number[3]; // Subnet address
                userMap.put(subnet, userMap.get(key) + 1);
                }
        }

        /*This method parses the csvFile so as to get a list of the set of IPs that
appear in the capture*/
        public void getIps(String csvFile){

                String[] packet = null;
         String line = "";
         String cvsSplitBy = " "; // use space as separator

         try (BufferedReader br = new BufferedReader(new FileReader(csvFile))) {

             while ((line = br.readLine()) != null) {
               packet = line.split(cvsSplitBy);
               // Since ipSet is a Set variable only unique values will be added in
each iteration
                        ipSet.add(packet[2]);
                        ipSet.add(packet[3]);
             }

         } catch (IOException e) {
             e.printStackTrace();
         }
    }

}
```

Appendix 2: IPT script

The Matlab script code below computes the IPT of a given .csv file that contains the trace of a captured traffic.

```
% In order to use this script a .csv file has to be imported.
% This .csv has to contain four columns: timestamp, IPsource, IPdestination and packet
% size.

format long;
[nr,nc] = size(snifftable);

for i=1:nr
    snifftable{i,1} = snifftable{i,1}*1000000; % convert to microseconds
end

timeshift = snifftable{1,1};

for i=1:nr
    snifftable{i,1} = snifftable{i,1} - timeshift; % shift timestamp origin 0
end

ipt = zeros(nr-1,2);

for i=2:nr
    ipt(i,1)= snifftable{i,1};
    ipt(i,2)= snifftable{i,1}-snifftable{i-1,1};
end
```

Appendix 3: tc and netem commands

In this section, the shell scripts used to perform the experiments presented in section 4 are detailed. Namely, the simulations of delay, available bandwidth and packet loss.

Firstly, this tiny shell script shows how the delays were set in the KMS for the different locations, in this example, 6ms.

```
!#/bin/bash
# Set delay to the KMS
ssh kcl@10.81.59.111 tc qdisc add dev ens18 root netem delay 6ms
```

Secondly, this shell script was used to simulate available bandwidth:

```
!#/bin/bash
# Considering that the KMS requires 2.4Mbps
echo "Please, start capturing – No bandwidth limitation"
sleep 60
echo "Videoconference traffic occupies 50% of the available bandwidth"
ssh kcl@10.81.59.111 tc qdisc add dev ens18 root tbf rate 4800kbit latency 0.01ms burst 1540
sleep 60
echo "Videoconference traffic occupies 60% of the available bandwidth"
ssh kcl@10.81.59.111 tc qdisc add dev ens18 root tbf rate 4000kbit latency 0.01ms burst 1540
sleep 60
echo "Videoconference traffic occupies 70% of the available bandwidth"
ssh kcl@10.81.59.111 tc qdisc add dev ens18 root tbf rate 3430kbit latency 0.01ms burst 1540
sleep 60
echo "Videoconference traffic occupies 80% of the available bandwidth"
ssh kcl@10.81.59.111 sudo tc qdisc add dev ens18 root tbf rate 3000kbit latency 0.01ms burst
1540
sleep 60
echo "Videoconference traffic occupies 90% of the available bandwidth"
ssh kcl@10.81.59.111 sudo tc qdisc add dev ens18 root tbf rate 2670kbit latency 0.01ms burst
1540
sleep 60
echo "Videoconference traffic occupies 100% of the available bandwidth"
ssh kcl@10.81.59.111 sudo tc qdisc add dev ens18 root tbf rate 2400kbit latency 0.01ms burst
1540
sleep 60
echo "Experiment finished"
```

Lastly, this shell script was used to simulate packet loss:

```
!#/bin/bash
echo "Please, start capturing – 0% packet losses"
sleep 60
echo "10% packet loss"
ssh kcl@10.81.59.111 tc qdisc add dev ens18 root netem delay 0.01ms loss 10%
sleep 60
echo "20% packet loss "
ssh kcl@10.81.59.111 tc qdisc add dev ens18 root netem delay 0.01ms loss 20%
sleep 60
echo "30% packet loss "
ssh kcl@10.81.59.111 tc qdisc add dev ens18 root netem delay 0.01ms loss 30%
sleep 60
echo "40% packet loss "
ssh kcl@10.81.59.111 tc qdisc add dev ens18 root netem delay 0.01ms loss 40%
sleep 60
echo "Experiment finished"
```

Appendix 4: Capturing traffic

This section pretends to show the procedure followed by the Sniffer to obtain the data traffic. It could be divided into three parts: capturing, filtering and formatting.

The *tcpdump* [67] command shown below captures the traffic in the interface *en4.* Notice that this command is already filtering the packets of our interest. In this case, the correspondence of IPs is: 10.81.59.111 – KMS, 10.81.59.112 – Application server, 10.81.59.104 – Client A and 10.81.59.105 – Client B.

```
sudo tcpdump -i en4 -n "(src 10.81.59.105 and dst 10.81.59.111) or (src 10.81.59.104
and dst 10.81.59.111) or (src 10.81.59.111 and dst 10.81.59.105) or (src 10.81.59.111
and dst 10.81.59.104)" -w example.pcap
```

In the case of needing to split the traffic, for instance, obtaining just the traffic from Client A to KMS, another filter could be applied a posteriori by means of *tshark* [68] command:

```
tshark -r filtered-clean-cap1.pcap -Y "ip.src == 10.81.59.104 && ip.dst == 10.81.59.111"
-w client-a-to-kms.pcap
```

Finally, the shell script below was created so as to convert *.pcap* files to *.csv* format. This script creates a file where each row is a captured packet and contains four columns: timestamp, IP source, IP destination and packet size.

```
!#/bin/bash

sudo tcpdump -nn -tt -r $1 > ${1%%.*}.txt
awk '{print $1, $3, $5, $(NF)}' ${1%%.*}.txt > raw-columns.txt
awk '{print $1}' raw-columns.txt > timestamps.txt
awk '{print $2}' raw-columns.txt > dirty-ipsource.txt
awk '{print $3}' raw-columns.txt > dirty-ipdest.txt
awk '{print $4}' raw-columns.txt > packet-lenght.txt
cut -d. -f-4 dirty-ipsource.txt > clean-ipsource.txt
cut -d. -f-4 dirty-ipdest.txt > clean-ipdest.txt
paste -d' ' timestamps.txt clean-ipsource.txt clean-ipdest.txt packet-lenght.txt >
snifftable.csv
rm ${1%%.*}.txt raw-columns.txt timestamps.txt dirty-ipsource.txt dirty-ipdest.txt
packet-lenght.txt clean-ipsource.txt clean-ipdest.txt
```

Appendix 5: Latency estimation script

The Matlab script code below computes the latency of a given .csv file that contains the trace of a captured traffic. The logic of this code follows the already presented methodology in section 3.2.3. In a nutshell, this algorithm tries to find for each first packet of a burst departing from Client A to KMS, the corresponding first packet of a burst departing from KMS to Client B. If it succeeds in the finding, the latency can be computed and stored. If not, the algorithm tries with the following packet in the trace, skipping the ones already studied.

```
% In order to use this script a .csv file has to be imported.
% This .csv has to contain four columns: timestamp, IPsource, IPdestination
% and packet size.

format long;
[nr,nc] = size(snifftable); % snifftable is the name of the imported .csv file
sender = '10.81.59.104'; % sender IP
latency_threshold = 15000; % maximum accepted latency, depends on the location (us)
distance_threshold = 0; % lower threshold, depends on the location (us)

% This loop sets the timestamps to microseconds
for i=1:nr
    snifftable{i,1} = snifftable{i,1}*1000000;
end

timeshift = snifftable{1,1};
% This loop shifts the timestamps, i.e. setting the first to zero.
for i=1:nr
    snifftable{i,1} = snifftable{i,1} - timeshift;
end

latencies = []; % stores the computed latencies
latencies_pro = []; % stores the computed latencies and the corresponding timestamp
discarded_latencies = []; % stores the not valid latencies, i.e. too high/low
number_packets_burst = []; % stores the number of packets in each burst
latencies_index = 0;
cont_discarded = 0;
cont_burst = 1;
burst_flag = 0;

i = 1;
% This loop goes over all the rows in the capture
while i<=nr
    timestamp = snifftable{i,1};
    ip_source = snifftable{i,2};
    ip_destination = snifftable{i,3};
    size = snifftable{i,4};
    % Just in case of being a packet departing from the sender a latency
    % will be computed
    if strcmp(ip_source,sender) == 1
        k = i+1;
        % This loop goes over all the consecutive rows in the capture so as
        % to find the corresponding packet from the server to the receiver
        for j=k:nr
            ts = snifftable{j,1};
            ips = snifftable{j,2};
            ipd = snifftable{j,3};
            s = snifftable{j,4};
```

```
            % In case of being part of the same burst, it is posted in the
            % burst counter
            if (strcmp(ips,ip_source)==1) && burst_flag == 0
                cont_burst = cont_burst +1;
            else
                % The burst has finished
                burst_flag = 1;
            end
            % In case that the corresponding packet is found the latency
            % can be computed
            lat = ts-timestamp;
            if lat>distance_threshold
                if (strcmp(ipd,ip_source)==0) && (strcmp(ipd,ip_destination)==0) &&
(s>=size) % This if cause evaluates if this packet corresponds to the one under study
                 burst_flag = 0;
                  if lat<latency_threshold
                      latencies_index = latencies_index+1;
                      latencies(latencies_index) = lat;
                      latencies_pro(latencies_index,1) = timestamp;
                      latencies_pro(latencies_index,2) = lat;
                      number_packets_burst(latencies_index)= cont_burst;
                      cont_burst = 1;
                      i = j;
                      break;
                  else
                      cont_discarded = cont_discarded +1;
                      discarded_latencies(cont_discarded) = snifftable{i,1};
                      break;
                  end
                end
            end
        end
    end
    i = i+1;
end
```

Glossary

AR: Augmented reality

Ecdf: Empirical cumulative distributed function

GUI: Guest User Interface

HQ: High quality

IBT: Inter-burst time

IoT: Internet of Things

IPT: Inter-packet time

KMLE: Konica Minolta Laboratory Europe

KMS: Kurento Media Server

LAN: Local Area Network

LXC: Linux containers

NFV: Network function virtualization

ODL: OpenDaylight

OS: Operating System

PaaS: Platform as a Service

POC: Proof of concept

QoE: Quality of Experience

QoS: Quality of Service

SDN: Software-defined networks

VE: Virtual environment

VM: Virtual machine

VNC: Virtual network computing

WS: WebSocket

WSS: WebSocket secure