Telecommunication Services' Migration to the Cloud

Network Performance analysis

ISAAC ALBARRÁN
and
MANUEL PARRAS

KTH Information and Communication Technology

Degree project in
Communication Systems
Communication Systems, 30.0 HEC
Stockholm, Sweden

Telecommunication Services' Migration to the Cloud

Network Performance analysis

Isaac Albarrán
Manuel Parras

29 April 2012

Master's Thesis at the School of Information and Communication Technology (ICT)

KTH Royal Institute of Technology
Stockholm, Sweden

Telecommunication Services' Migration to the Cloud, a Network Performance Analysis
ISAAC ALBARRÁN
MANUEL PARRAS
Master Thesis
Academic supervisor: Gerald Q. Maguire Jr.
Industrial supervisors: Leif Y. Johansson and Nikhil Tikekar
Registration number: TRITA-ICT-EX-2012:54

Abstract

Nowadays, telecommunication services are commonly deployed in private networks, which are controlled and maintained by the telecommunication operators themselves, by co-location services providers, or, to some extent, by their hardware and software providers. However, with the present development of cloud computing resources, one might consider if these services could and should be implemented in the Cloud, thus taking advantage of cloud computing's high availability, geographic distribution, and ease of usage. Additionally, this migration could reduce the telecommunication operators' concerns in terms of hardware and network maintenance, leaving those to the Cloud computing providers who will need to supply a highly available and consistent service, to fulfill the telecommunication services' requirements. Furthermore, virtualization provides the possibility of easily and rapidly changing the Cloud network topology facilitating the addition and removal of machines and services, allowing telecommunication services providers to adapt to their demands on the fly.

The aim of this thesis project is to analyze and evaluate the level of performance, from the network point of view, that can be achieved when using Cloud computing resources to implement a telecommunication service, carrying out practical experiments both in laboratory and real environments. These measurements and analyses were conducted using an Ericsson prototype mobile switching center server (MSC-S) application, although the results obtained could be adapted to other applications with similar requirements.

In order to potentially test this approach in a real environment, a prior providers' survey was utilized to evaluate their services based on our requirements in terms of hardware and network characteristics, and thus select a suitable candidate environment for our purposes. One cloud provider was selected and its service was further evaluated based on the MSC-S application requirements. We report the results of our bench-marking process in this environment and compare them to the results of testing in a laboratory environment. The results of both sets of testing were well correlated and indicate potential for hosting telecommunication services in a Cloud environment, providing the Cloud meets the requirements imposed by the telecom services.

Resumen

Actualmente, los servicios de telecomunicaciones se implementan comúnmente en redes privadas, controladas y mantenidas por los operadores de telecomunicaciones, por proveedores de servicios de colocación o, hasta cierto punto, por proveedores de hardware y software. Sin embargo, con el presente desarrollo de la tecnología de '*Cloud computing*', se puede considerar la posibilidad de implementar servicios de telecomunicaciones en la nube, aprovechando su alta disponibilidad, distribución geográfica y facilidad de uso. Además, este cambio puede reducir las preocupaciones de los operadores en relación al mantenimiento del hardware y de la red, delegando en los proveedores del servicio de '*Cloud computing*', los cuáles deberán proporcionar un servicio consistente, cumpliendo así con los requisitos de los servicios de telecomunicaciones. Por otra parte, la virtualización propociona la posibilidad de cambiar rápida y fácilmente la topología de la red, facilitando la adición y supresión de maquinas y servicios, y, por tanto, permitiendo a los operadores adaptarse a sus necesidades sobre la marcha.

El objetivo de esta tésis es analizar y evaluar en nivel de rendimiento, desde el punto de vista de la red, que se puede conseguir usando recursos de '*Cloud computing*' para implementar un servicio de telecomunicaciones, llevando a cabo experimentos tanto en el laboratorio como en un entorno real. Estos análisis fueron realizados utilizando un prototipo de un servidor de conmutación móvil (MSC-S) de Ericsson, aunque los resultados pueden adaptarse a otras aplicaciones con unos requisitos similares.

Para probar esta propuesta en un entorno real, se realizó una encuesta de proveedores de servicios de '*Cloud computing*', con el objetivo de evaluar sus servicios teniendo en cuenta nuestros requisitos de hardware y red. Finalmente, un proveedor fue escogido y su servicio evaluado basándonos en los requisitos de la aplicación MSC-S. En este documento proporcionamos los resultados de esa evaluación y los comparamos con los obtenidos en el laboratorio. Los resultados de ambas evaluaciones fueron satisfactorios e indican la posibilidad de implementar servicios de telecomunicaciones en la nube, siempre que la nube cumpla los requisitos impuestos por dichos servicios de telecomunicaciones.

Sammanfattning

Nuförtiden är telekommunikationstjänster ofta uppsatta i privata nätverk, som kontrolleras och underhålls av teleoperatörerna själva, av samlokaliserande tjänsteleverantörer eller i viss utsträckning av deras hårdvaru- och programvaru-leverantörer. Med den nuvarande utvecklingen av Cloud Computing-resurser kan man dock överväga om dessa tjänster kan och bör genomföras i ett Cloud, vilket drar fördel av Cloud Computings höga tillgänglighet, geografiska spridning, och enkla användning. Denna migration minskar även teleoperatörernas oro angående hårdvaru- och nätverks-underhåll genom att överlåta detta till Cloud Computing-leverantörerna, som kommer att behöva leverera en hög tillgänglighet och konsekvent service för att uppfylla telekommunikationstjänsternas krav. Dessutom ger virtualisering möjlighet att enkelt och snabbt ändra ett Clouds nätverkstopologi, vilket underlättar tillägg och borttagning av maskiner och tjänster, vilket hjälper teleoperatörer att snabbt anpassa sig till deras krav.

Målet med examensarbetet är att analysera och uppskatta prestandan, från nätets perspektiv, som kan uppnås vid användning av Cloud Computing-resurser för att genomföra en teletjänst, genom praktiska experiment både i laboratorium och i verkligheten. Dessa mätningar och analyser utfördes med en prototyp av en Ericsson mobilomkopplingscentralserverapplikation (MSC-S), även om de erhållna resultaten skulle kunna anpassas till andra program med liknande krav.

För att potentiellt kunna testa denna metod i en verklig miljö användes en tidigare leverantörs undersökning för att utvärdera deras tjänster baserat på våra krav på hårdvara och nätverksegenskaper, och genom detta välja en lämplig kandidatmiljö för våra syften. En Cloud-leverantör valdes och dess tjänster utvärderades vidare baserat på MSC-Ss applikationskrav. Vi redovisar resultatet av vår testprocess i den här miljön och jämför det med resultaten av tester i laboratoriemiljö. Resultaten från båda uppsättningarna av tester var väl korrelerade och visar på potentialen av att implementera telekommunikationstjänster i en Cloud-miljö, om detta Cloud uppfyller de kraven som ställs av telekommunikationtjänsterna.

Acknowledgements

First of all, we would like to deeply thank our Ericsson's industrial supervisors Leif Johansson, Niklas Waldemar and Nikhil Tikekar. They entrusted us with this amazing and innovative project and supported us in every step. They also contributed to our work with their great expertise in the area, and taught us many things that will be essential in our future professional career as engineers.

We would also like to sincerely thank professor Gerald Q. "Chip" Maguire Jr. for accepting to be our academic tutor and examiner, and showing such a great interest in the subject of this master's thesis. We would also like to express our gratitude for all his support. He always answered our questions rapidly and precisely, and provided us with very useful feedback that improved the quality of our work.

Manuel:

My sincerest appreciation to my friends of Spain who have been supporting me as always in spite of being far away from home.

I additionally want to extend my thanks to all the people who have accompanied me in this great adventure that started in Sweden in 2010 and with who I have spent unforgettable moments, especially Itzi who has encouraged and supported me in every moment, even in the distance.

Last but not least, I want to express my greatest gratitude to my family, especially my parents Luis and María Victoria, because of their never-ending love and support; and my brothers Luis and Miguel Angel, and my sister María Victoria, since I am who I am because of them.

Without the support and encouragement of all these people, none of this would have been possible. ¡Muchas gracias! Thank you very much! Tack så mycket!

Isaac:

First of all, I would like to affectionately express my gratitude to my fiancee, Mercedes, who has been by my side from the beginning regardless of the distance between us, and has given me the strength to keep on every day.

In addition, I would like to thank my family, my parents Isaac and María Dolores, my brother Nacho, and my sister Marta, for the great support and love shown towards me, without which none of this would have been possible.

Finally, I would like to thank my friends, both from Spain and those I have met during my stay in Sweden. Specially, I would like to acknowledge Manuel Parras, Daniel García, Armando Gutiérrez, and Nacho Mulas all of whom shared this important stage of my life with me, and supported me through it.

Contents

List of Figures

List of Tables

List of Code Listings

List of Acronyms and Abbreviations

3GPP	Third Generation Partnership Project
API	Application Programming Interface
APG	Adjunct Processor Group
ARP	Address Resolution Protocol
AUC	Authentication Center
BGCF	Breakout Gateway Control Function
BGP	Border Gateway Protocol
BSC	Base Station Controller
BSS	Base Station System
BTS	Base Transceiver Station
BW	Bandwidth
CAPEX	Capital Expenditure
CPU	Central Processing Unit
CS	Circuit-Switched
CSCF	Call Session Control Function
CSMA/CD	Carrier Sense Multiple Access with Collision Detection
DEC	Digital Equipment Corporation
DoS	Denial of Service
EDGE	Enhaced Data rates for GSM Evolution
EIR	Equipment Identity Register
eNB	Evolved Node B

EoIP	Ethernet over IP
EPC	Evolve Packet Core
ETSI	European Telecommunications Standard Institute
GB	Gigabyte
Gbps	Gigabits per second
GHz	Gigahertz
GPRS	General Packet Radio Service
GSM	Global System for Mobile communications
GUI	Graphical User Interface
HLR	Home Location Register
HSS	Home Subscriber Server
IaaS	Infrastructure as a Service
IDE	Integration Development Environment
IEEE	Institute of Electrical and Electronic Engineers
IETF	Internet Engineering Task Force
IGMP	Internet Group Management Protocol
IMEI	International Mobile Equipment Identities
IMS	IP Multimedia Subsystem
IMSI	International Mobile Subscriber Identity
I/O	Input/Output
IP	Internet Protocol
ISO	International Standard Organization
IT	Information Technology
KVM	Kernel-based Virtual Machine
LAN	Local Area Network
LTE	Long Term Evolution
LTS	Long Term Support

MAC	Media Access Control
MB	Megabyte
Mbps	Megabits per second
MGCF	Media Gateway Control Function
MGW	Media Gateway
MIPS	Million Instructions Per Second
MME	Mobility Management Entity
MPLS	Multi-Protocol Label Switching
MRF	Media Resource Controller
ms	millisecond
MSC	Mobile services Switching Center
MSC-S	Mobile Switching Center Server
MSC-S BC	MSC-S Blade Cluster
MSISDN	Mobile Station Integrated Services Digital Network
MTBF	Mean Time Between Failures
MTTR	Mean Time To Repair
MTU	Maximum Transmission Unit
NAT	Network Address Translation
NGN	Next Generation Network
NIST	National Institute of Standards and Technology
NMC	Network Management Center
NMS	Network Management Subsystem
NSS	Network Switching Subsystem
OMC	Operation and Maintenance Center
OPEX	Operational Expenditure
OS	Operating System
OSI	Open Systems Interconnection

OSPF	Open Shortest Path First
OSS	Operation Support System
PaaS	Platform as a Service
PC	Personal Computer
PCP	Priority Code Point
PDN GW	Packet Data Network Gateway
PIM	Protocol Independent Multicast
PS	Packet-Switched
PSTN	Public Switched Telephone Network
QoE	Quality of Experience
QoS	Quality of Service
RAM	Random Access Memory
RAN	Radio Access Network
RIP	Routing Information Protocol
RFC	Request For Comments
RNC	Radio Network Controller
RTT	Round-Trip Time
SaaS	Software as a Service
SAE	System Architecture Evolution
SBC	Session Border Controller
SCTP	Stream Control Transmission Protocol
SGW	Serving Gateway
SIM	Subscriber Identity Module
SIP	Session Initiation Protocol
SIS	Site Infrastructure Support
SLA	Service Level Agreement
SMS	Short Message Service

SPX	Signaling Proxy
SS	Switching System
SSH	Secure Shell
TCP	Transmission Control Protocol
TDM	Time-Division Multiplexing
TLB	Translation Lookaside Buffer
UDP	User Datagram Protocol
UPS	Uninterruptible Power Supply
UMTS	Universal Mobile Telecommunications System
UTRAN	UMTS Radio Access Network
VAX	Virtual Address Extension
VLAN	Virtual Local Area Network
VLR	Visitor Location Register
VM	Virtual Machine
VPN	Virtual Private Network

Chapter 1

Introduction

The aim of this chapter is to describe the thesis project and the organization of the thesis itself. First, an overview of the subject is provided so that the reader can grasp the context of the study and completely understand the rest of the thesis. After that, the problems addressed in this project are described, followed by a statement of the goals and the purpose of the project. Next, the methodology that was adopted to solve the problems previously posed is described. Thereafter, the target audience of this thesis is presented, and the scope and limitations of the project are clearly stated. Finally, an outline of the thesis itself is presented to highlight the structure of the thesis in order to make it easier for the reader to utilize this thesis.

1.1. Overview

Nowadays, services such as telephony, Internet access, and Short Message Service (SMS) in mobile terminals are increasingly expected by end users to have the same Quality of Experience (QoE) as the user obtains when using wired devices [1]. However, the amount of resources necessary to provide such large scale services increases with the demands upon the service, and the capital expenditure (CAPEX) required to build and sustain such a deployment is becoming a major concern for telecommunication operators [2]. Furthermore, demands in terms of bandwidth are increasing, especially due to the emergence of new services and applications requiring Internet access [3], as it can be seen in Table 1.1. Therefore, an important research challenge is to develop new solutions so that the initial expenses are reduced and, at the same time, future needs can be met easily and without extensive modifications when scaling up resources. Currently, solutions to boost mobile networks' bandwidth are being addressed by the Long Term Evolution/Evolved Packet Core architecture (LTE/EPC) [4] [5]. LTE introduces sophisticated radio-communication techniques, enabling faster and more efficient access networks, while EPC involves the deployment of a packet-based core network capable of dealing with future traffic increases [6]. Additionally, the IP Multimedia Subsystem (IMS) [7] [8]

[9] is the main framework to provide voice and SMS services over IP.

Table 1.1. Global Mobile Data Traffic Growth. Adapted from [10]

Year	Annual Increment
2008	156%
2009	140%
2010	159%
2011 (expected)	131%
2012 (expected)	113%

The National Institute of Standards and Technology (NIST) [11] defines Cloud computing as "a model for enabling ubiquitous, convenient, on-demand network access to a shared pool of configurable computing resources (e.g., networks, servers, storage, applications, and services) that can be rapidly provisioned and released with minimal management effort or service provider interaction".

Cloud computing is an emerging industry with the active participation of many companies, including many telecommunication operators as illustrated by Figure 1.1. The set of services being offered by these companies is also becoming broader, thus reaching and satisfying a more diverse range of customer requirements – enabling these customers to outsource their IT operations [12]. Data warehousing, computing as a service, and virtual data centers are just a few examples of cloud services. At the same time, exploiting cloud computing enables customers to save hardware acquisition and network deployment costs. This thesis considers how to provide telecommunication services through cloud-based solutions. We are not the first to consider this idea, as it has already been pondered by Caryer et al. [13], who suggested the utilization of cloud computing resources to support the development of Next Generation Networks (NGN).

What remains to be seen is whether current cloud services meet the Quality of Service (QoS) requirements for implementing applications such as a Mobile Switching Centre Server (MSC-S), Home Location Register (HLR), or any other application on which mobile services are dependent. Our study has focused on a practical demonstration of whether a cloud-based implementation is possible for an MSC-S application, considering the difficulties that such a deployment entails, and on evaluating the results obtained. It is important to note that telecommunication services demand a QoS that is considered to be difficult to achieve in a normal environment, but despite this, cloud technology needs to be able to deliver the required QoS. Additionally, all the drawbacks that use of the public Internet may introduce when providing a service through it must be kept in mind (Oredope and Liotta also regarded this as an important concern in [14]). However, other issues also need to be addressed in our investigation. For instance,

Figure 1.1. Cloud computing companies

as it is mentioned in [15], interoperability among different cloud environments could be a concern, and tests should be performed to examine this possible complication.

1.2. Problem description

The main problems this thesis project deals with (and to which a solution is provided) are:

- Telecommunication applications generally have a weakness due to centralization deployment: These applications are commonly run on specialized hardware components, and these components are normally deployed together in racks. This implies the existence of a single point of failure if something harmful occurs in the facilities where these racks are placed. In addition, these centralised implementations limit services' scalability, and the specific geographic location of these servers leads to delay constraints upon some services.

- Large CAPEX and high risk in deploying a telecommunication network: The design and implementation of telecommunication networks entail large initial investments, which have become the main difficulty for new telecommunication operators - creating a high barrier to entry that favors incumbents.

Additionally, the level of risk associated with such a venture is very high for investors.

- Uncertainty of the suitablity of migrating telecommunication applications to the Cloud: We consider a solution where Cloud computing is used to address the problems stated above, as suggested by Caryer, et al. [13]. However, as of the time this thesis project was proposed in August 2011, Cloud environments had not yet been tested against the constraints that these applications require in terms of QoS, therefore thorough testing needed to be carried out to prove that a cloud based solution is actually suitable.

1.3. Goals

The main goals of this master thesis project are:

- Practically demonstrate that an Ericsson MSC-S Blade Cluster (MSC-S BC) can be implemented deploying prototype MSC-S blades with limited functionality in a current cloud computing environment. Carry out tests to observe whether this solution may have performance problems from the network perspective, especially in light of the QoS requirements imposed by telecommunication services.

- Extend the first goal to the case where the prototype MSC-S blades that form the Ericsson MSC-S BC are *geographically* distributed. Conduct tests to demonstrate that the performance of the implementation meets the application's QoS requirements.

In order to ease our task and to efficiently achieve our main goals we divided the work into several tasks. The tasks on which we focused are:

- Design a completely functional network environment so that the prototype MSC-S blades can be run on standard x86-based computers while meeting the application's requirements (we consider the environment's network configuration and limitations during this design process). After the configuration of the network and computer environment, we deployed the application prototype in order to test that it operates correctly (from a functional point of view). After ensuring that it operates correctly we will conduct performance testing to ensure that the solution meets all of the performance requirements, i.e., that the solution is suitable for our purposes.

- Utilize the previous solution in a test bed to simulate a cloud within the Ericsson network, so that the MSC-S application prototype can be tested in the presence of impairments that might be introduced by a real Cloud environment. These impairments were simulated by adding Cloud network penalties such as delay and packet loss. In addition, a vitualization layer was added over the native machine on which the application was run.

- Conduct a survey to evaluate current Cloud computing providers in the market to see which can meet our requirements in terms of computing capacity and network capabilities, while preparing for our last step: testing the application in a real Cloud environment.
- Verify that it is possible to exploit available commercial Cloud computing resources to host the MSC-S application prototype by performing thorough tests in a selected provider's Cloud.

1.4. Purpose

The purpose of this thesis is to describe the design, implementation, and evaluation of a network solution for the Ericsson MSC-S BC so that the problems stated in Section 1.3 can be solved using current commercially available Cloud computing services. Success in such a task would provide a firm foundation for future telecommunication network application deployments in a Cloud. Additionally, the testing will address important issues regarding this solution. Since Cloud services provide very simple billing schemes, the initial expenses needed to build a modern telecommunication network would be reduced substantially. In addition, the distributed resources that could be utilized via Cloud computing could increase redundancy in the network, potentially preventing major service disruptions from ocurring due to many different types of failures. Finally, a satisfactory Cloud solution for this sort of applications would introduce an inflexion point in the design and deployment of modern telecommunications networks and would certainly lead to further investigations concerning the migration of other telecommunication services and applications to a Cloud environment.

Furthermore, this thesis provides a thorough analysis of current main Cloud computing providers. The analysis focuses on the constraints of the Ericsson MSC-S BC application, such as the required computing performance (in terms of bounded service times), bandwidth, security, Service Level Agreements (SLAs), and appropiate access to the Could resources in order to have full control of the machines' configuration. This survey provides useful insights into which Cloud providers are suitable to host telecommunication services requiring high QoS.

1.5. Methodology

In order to fulfill the goals of this master's thesis project, both qualitative and quantitative approaches were utilized. Secondary research was used as a qualitative method, starting with a literature review. Since previous studies may have already considered Cloud computing as a solution to implement telecommunication services for mobile networks we began by seeking out existing solutions. Moreover, this review provided material for our background chapter and allowed us to obtain a full state-of-the-art overview of the subject. This literature review also provided a solid foundation upon which we could base our ideas.

In addition, another qualitative method was our survey. The purpose of this survey was to evaluate the current Cloud computing providers and analyze whether any of the offerings were suitable for our task. The first part of this survey, a thorough secondary search, was conducted to narrow down the number of Cloud provider candidates based on their websites and information in their white papers. We eliminate from further consideration any providers that did not meet our basic technical requirements. Secondly, personal enquiries were posed to the first providers to respond, enabling us to clarify our understanding of their offerings. These personal queries were conducted by e-mail and phone conversations. Finally, a final decision to select a single Cloud provider for testing was made based on all the data obtained from the survey and personal enquiries. In Chapter 5, a more detailed explanation of the survey process is provided.

Based upon the literature study, and the requirements of our employer, a set of test cases were developed to guide the design and implementation of our solution to the problems described in Section 1.2 for the Ericsson MSC-S BC. Those experiments enable us to quantify the performance of our solution and to compare this performance to our employer's QoS requirements for the telecommunications network application that was selected for our testing. Additionally, the Cloud provider selected as a potential test bed from the survey was evaluated on the basic requirements of the Ericsson MSC-S BC application. The measurements were conducted over a range of representative network parameters, computing capacity and memory latency.

1.6. Scope

The project's aim was to look for a stable solution, perform an analysis of the solution's performance, detect the hindrances that could appear from a network perspective, and overcome them when possible. Therefore, a comparison of the performance of other solutions is outside the scope of this project.

In addition, some issues that we detected, which had nothing to do with the network aspects of the implementation, could not be resolved due to the complexity of the Ericsson MSC-S BC system. Solving these issues was out of the scope of this project, which was focused on the network aspects of the implementation, and required more complete knowledge of the system.

Finally, the technical implementation decisions made for our solution are explicitly stated and further explained below:

- The practical experiment was carried out using a Ericsson proprietary MSC application prototype with limited functionality. In the future, further studies should be conducted to verify the correct behaviour of a completely functional Ericsson MSC-S BC application, as well as other (related) applications to see if the results of this study can be generalized to other (similar) applications.
- A Linux environment was used to run the application, as Linux was the target execution environment for both the virtual machines in the Cloud and the application's current host operating system. Ubuntu 10.04 LTS was the selected distribution due to its stability, ease of use, and our familiarity with it. While this was not the same Linux distribution used by the application in its commercial deployment (OpenSUSE), the performance differences between Linux implementations were not considered to be a significant issue.
- The tests were realized on a simulated Cloud in a laboratory environment and on a single Cloud provider's real Cloud environment. Comparison with other Cloud implementations is left as future work, but we expect different environments to give comparable results (after compensating fot the difference in configurations - such as processor speed, available memory, disk throughput, network bandwidth, etc.).
- Only two network tunneling implementations were considered in the experiments: Virtual Private Network (VPN) and Ethernet over IP (EtherIP), as the purpose of this project was not to evaluate all the options but to implement a solution that was functional.
- Kernel-based Virtual Machine (KVM) and VMware were the only virtualization hypervisors used in the project. These two hypervisors were used to simulate a real Cloud virtualization environment, and they were considered sufficient for our purposes. We were not concerned with the small difference in performance of different virtualization methods, since we were not looking for small differences, but rather want to see if the proposed solution could meet the basic QoS requirements.
- A specific set of measurement tools were selected to collect the desired performance data. The selection of these tools together with our test results and analysis are presented in Chapter 4.

1.7. Limitations

The first of the limitations we faced was our lack of background knowledge regarding the solution we were expected to implement. As far as we knew, there had not been another investigation with a similar purpose as ours, although the idea was suggested in a general manner by Caryer, et al. [13].

Moreover, we suffered several setbacks in our efforts to deploy the simulated Cloud within Ericsson's internal network. These complications were associated with security and privacy policies that, at first, prevented us from freely configuring the environment. Additionally, when we contracted for service from a commercial Cloud service provider, we had some problems communicating with this Cloud provider's machines with the Secure Shell (SSH) protocol, as SSH was filtered out by the gateway of the Ericsson network. However, these issues were quickly resolved by our project supervisor and our manager.

The last step of the project, which consisted of testing the Ericsson MSC-S in a real cloud environment was not fully accomplished due to Ericsson's privacy policies. Permission to upload the application to such external location was not granted, and therefore this last step was limited to a thorough bench-marking process – with the goal of demonstrating theoretically that the system would operate correctly in such environment.

1.8. Target audience

Researchers and telecommunication operators interested in Cloud computing's potential uses within the telecommunication services market are the main targets of this thesis, specially, those who are concerned about the network performance of this sort of solution. In addition to these readers, any person interested in acquiring knowledge about the testing environment used in this research project can take advantage of this thesis as additional documentation about how to realize such a test environment and how to conduct performance measurements.

1.9. Thesis outline

The thesis is structured in a linear manner, where earlier chapters provide a general overview of the subjects necessary to understand the remainder of the thesis. It is strongly recommended that the reader study the theoretical background to provide an appropiate context for the subsequent experimental work.

Chapter 1 provides a general introduction to the thesis. While Chapter 2 provides general background information. Chapter 3 describes how the Ericsson MSC-S BC prototype could be run in a Cloud environmnent, while Chapter 4 describes tests of it in a laboratory emulated Cloud environment. Chapter 5 describes the survey of Cloud service providers and the results of our survey and bench marking. Chapter 6 presents our conclusions and suggest future work.

Appendix A is a manual for use by others who might want to configure a laboratory Cloud environment and run the tests tools that we have used. Appendix B includes the tables that contain the samples taken in Chapter 4. Appendix C contains the details of the Cloud services prover survey. Finally, Appendix D presents the scripts that were used for the bench-marking processes conducted in this project.

Chapter 2

General background

The purpose of this chapter is to give a brief overview of the technologies and concepts involved in this thesis so that the reader can easily read the rest of the thesis and understand why and how the work has been done. In addition, the information provided here focuses on the essential ideas that are directly related to this project, without going into unnecessary details.

Since the purpose of this thesis project is to analyze whether telecommunication services, specifically mobile networks' applications, can be migrated to the Cloud in practice from a network perspective, a short summary of IP networking is provided in Section 2.1, followed by an overview of the two main technologies that are relevant to the project: Cloud Computing (Section 2.2) and mobile network architectures (Section 2.3). Finally, a brief theoretical description of the Ericsson MSC-S BC prototype is provided in Section 2.4, since this was the platform on which the first tests were performed.

2.1. Networking

This section presents and describes the networking concepts necessary to undertand this thesis. Initially, a brief explanation of the layering concept characteristic of many network architectures is provided. Following this, each protocol that will be relevant to this thesis is described. The order that we will follow is from the bottom to the top of the protocol stack.

2.1.1. Network Layering concept

A computer network consists of several computers interconnected with each other via a network. Different models of network stacks exist nowadays, but the models that are most widely used are: the Open Systems Interconnection (OSI) model and the TCP/IP model. The TCP/IP model is based on the Transmission Control Protocol (TCP) and the Internet Protocol (IP) family of protocols. Both of these models are based on the concept of protocol layers [16]. According to this concept each layer is responsible for some functions in the communication process, thus reducing the complexity of the total protocol by isolating information that is not relevant to other layers within a layer. This layered architecture makes it easy to adapt the network model as new technologies evolve, since in principle each layer can be modified independently of the others – as long as it continues to provide the same services to the higher layer and to require the same services from the lower layer. Each layer offers a set of services to its immediate upper layer, while the latter relies on the services provided by the former when realizing its own functions. When sending data, the user generally passess user data through all those layers, from the application layer on the top to physical layer in the sender and back up the stack at the receiver.

The TCP/IP model is a modification of the OSI model. The TCP/IP model is the basis of the Internet [17]. The TCP/IP network model specifies the set of rules and protocols that are necessary to communicate among hosts within a TCP/IP network. These rules define how data should be sent in terms of forming IP packets, addressing these packets to their destination, routing packets, and forwarding these packets to the correct destination [18]. The TCP/IP model has four layers, while OSI model has seven layers, as we can see in Figure 2.1. Although they differ from each other, they are very similar from the transport layer down to the physical layer.

2.1.2. Network Access layer

There are several network access layer protocols available. For example, Ethernet is the dominant local area network (LAN) technology and has been defined in various standards developed by the IEEE 802.3 working group [20]. The initial version of Ethernet used a physical shared media (a coaxial cable), hence there was a need for a media access control protocol to decide when a host could transmit a frame, how long it should wait before transmitting, and when it should re-transmit. Subsequently the Ethernet standard was broadened to include many different types of media. Increasingly Ethernet is implemented by connecting each individual host to a switch. An example of switched Ethernet network can be seen in Figure 2.2 (A).

The main elements of which the Ethernet technology is composed are the access protocol, the hardware components, and the Ethernet frame structure. When data (and potentially also a logical link layer header) reaches the Ethernet layer it is encapsulated in Ethernet frames that are forwarded across the network to its destination. The Ethernet access method is called Carrier Sense Multiple Access with Collision Detection (CSMA/CD). This access method provides a way of sharing the medium so that all the hosts in the network can transmit data to each other

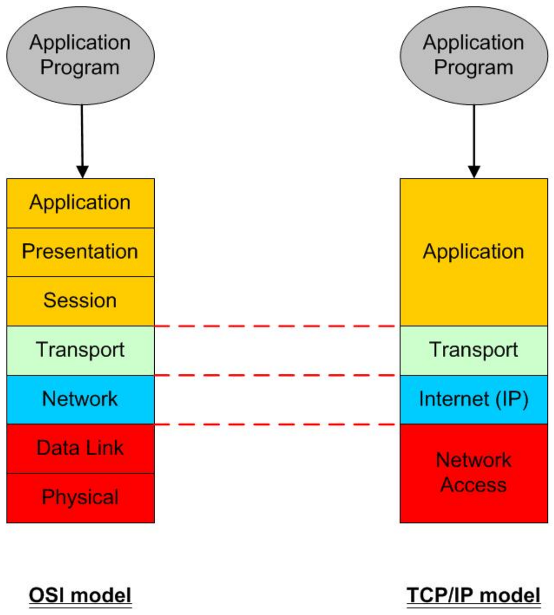


Figure 2.1. Comparison of OSI and TCP/IP models. Adapted from figure 1 of [19].

without monopolizing all the resources. However, in a switched Ethernet there is no reason to use the CSMA/CD protocol, hence later versions of the IEEE 802.3 standards define different media access and control protocols, for example, allowing full duplex communication and allowing elimination of inter-frame gaps. Finally, the hardware includes the cables, connectors, and electronic circuitry that allows the transfer of data in a LAN using Ethernet technology.

Furthermore, for other than point-to-point communication an addressing method is necessary so that the destination can know that a given frame is potentially for it and in some cases so that the receiver can know to whom a reply should be set. In the case of Ethernet we use a Media Access Control (MAC) address, which consists of a forty-eight-bit number. There are two main types of MAC addresses: unicast and multicast (which includes link layer broadcast). Additionally one bit of the MAC address indicates whether this address is globally unique or not. At the time of manufacturing an Ethernet interface the vendor assigns a unique address to each of their network interfaces. In order to assign globally unique MAC addresses the vendors buy blocks of 2^{24} addresses from the IEEE. In the case of Internet Protocol version 4 (IPv4) a translation from IP address to MAC address is possible using the Address Resolution Protocol (ARP). Details of the operation of ARP can be found in [21].

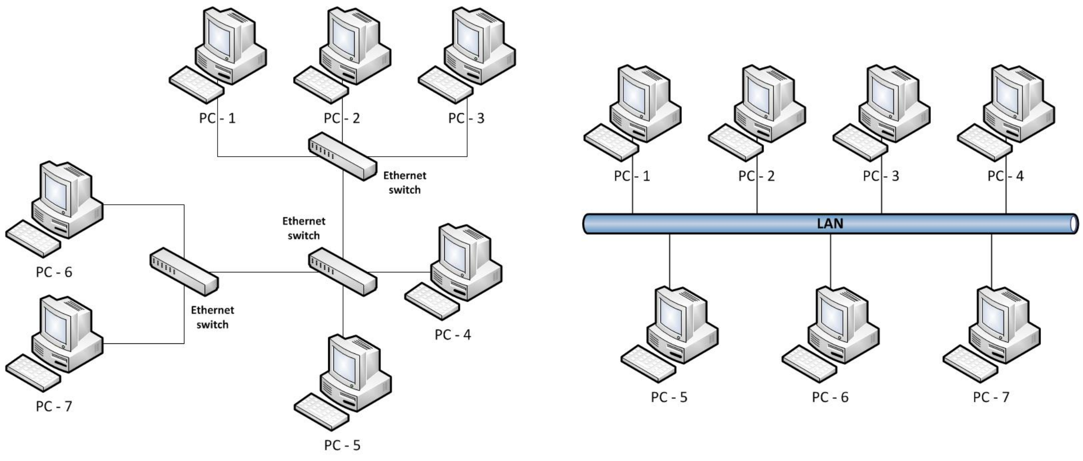


Figure 2.2. A switched Ethernet network topology (A) and the logically equivalent LAN representation (B).

In summary, today Ethernet refers to both the hardware and software that realizes the most common implementation of the lowest level communication layer and it delivers frames of bits between machines in an Ethernet network. Ethernet is the most common first layer of the TCP/IP model, thus any protocol built on top relies on the services provided by Ethernet to provide their own services.

2.1.3. Virtual Local Area Network (VLAN)

A Virtual Local Area Network (VLAN) [22][23][24] is composed by a group of hosts that communicate with each other as if they were in the same broadcast domain (a subnetwork) without nesessarily being in the same physical location, hence they forming a virtual subnetwork on top of the physical network. A VLAN has the same properties as a physical LAN, although VLAN membership configuration is implemented by software instead of reallocating physical devices and connections as would be required for physical LANs.

Additionally, if several VLANs share a physical connection, it is important to remember that the bandwidth is shared by all the Virtual LANs. VLANs are set up by tagging the traffic for each VLAN (for example using IEEE 802.1Q tags), by adding a tag field to the Ethernet control header, and by configuring layer 2 switch ports [25]. An example of the virtualization provided by this technology is illustrated in Figure 2.3. Note that it is also possible to use the IEEE 802.3Q three bit Priority Code Point (PCP) field to specify priorities, hence the different VLANs need not equally shared the bandwidth.

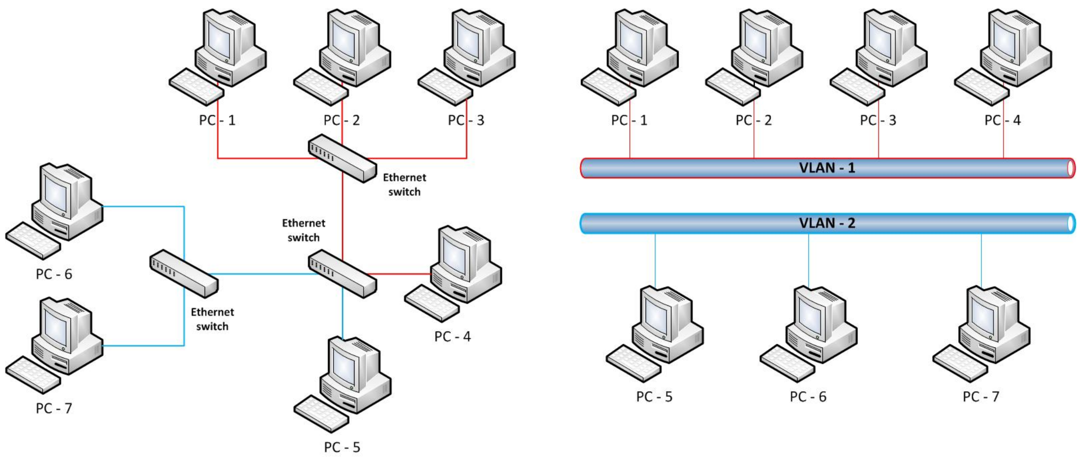


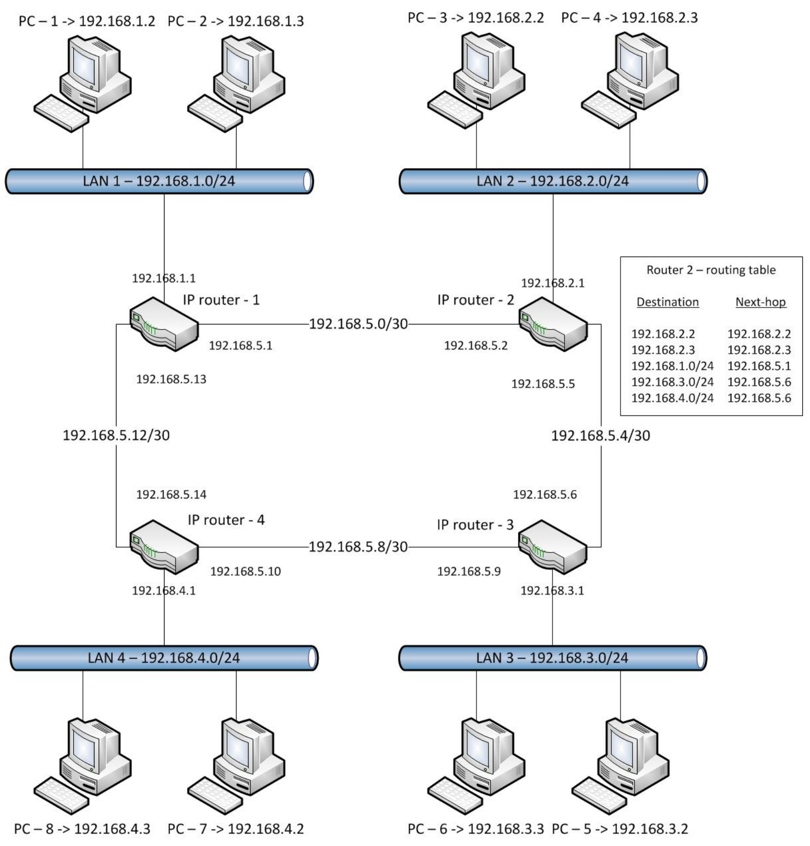
Figure 2.3. Deployment of two VLANs (1 and 2) over a physical LAN.

2.1.4. Internet Protocol (IP) layer

IP is the core technology upon which the global IP network, more commonly known as "The Internet", is based. As Request For Comments (RFC) 791 points out, IP is a protocol whose purpose is "to provide for transmitting blocks of data called datagrams from sources to destinations, where sources and destinations are hosts identified by fixed length addresses" [26]. IP is commonly considered to be a best-effort protocol, which means that it does not guaranteed that the datagrams reach their destinations.

The only IP concept we are going to detail is routing, is how IP packets reach their target. Routers are intermediate devices that forward datagrams to their correct destinations using the information provided in the IP header of the packet and the routing information which the router has. Each router keeps a routing table and makes forwarding decisions based on entries in this table. Routing tables can be manually configured, which normally only occurs in small networks, or dynamically configured by routing protocols when dealing with a large number of routes. Some well-known examples of routing protocols are Open Shortest Path First (OSPF), Routing Information Protocol (RIP), and Border Gateway Protocol (BGP). IP routes are commonly specified for blocks of IP addresses (subnets), which reduces the size of the routing table (as opposed to having to keep an entry for each possible destination IP address). The specific route determines the output interface to which a given IP datagram should be forwarded on. A route specification consists of the destinations that can be reached by the route and the next hop on this path (this is generally stored as a three tuple: destination IP address. network mask, and outgoing interface - but the table may also include weights, maximum segment size, maximum transport unit size, etc.). An example IP network is illustrated in

Figure 2.4, where the routing table for router 2 is represented.



Router 2 – routing table

Destination	Next-hop
192.168.2.2	192.168.2.2
192.168.2.3	192.168.2.3
192.168.1.0/24	192.168.5.1
192.168.3.0/24	192.168.5.6
192.168.4.0/24	192.168.5.6

Figure 2.4. IP network example. The routing table presented contains both single and multidestination routes.

In addition to globally routable IP addresses, there are blocks of IP addresses used only for private purposes, thus IP packets to these destination addresses are never forwarded to the global Internet; hence these addresses can be utilized repeatedly within separate private IP networks. These IP addresses are referred to as private IP addresses, while the globally routable Internet addresses are referred to as public IP addresses. Therefore, these private IP addresses are commonly used for internal company networks, testing environments, or any other network that is isolated from the Internet. However, Internet access can be provided to hosts that utilize a private IP address by introducing a gateway, such as a Network Address Translation (NAT). Packets arriving at the NAT can have a public destination address (hence they may be forwarded to the internet) or they may have a private destination IP address in which case they can only be forwarded within the private network. Note that in all cases before forwarding a packet with a private IP source address the NAT must replace this source address with one of its own source addresses (and perhaps modify the transport protocol source port number - so that it can return a response packet to the correct internation host's interface). A NAT that is connected to the global internet must of course replace a private IP source address with one of its own public IP addresses.

The Internet Protocol is more complex than presented here, and many of its characteristics have been omitted in this brief summary. However, a thorough explanation of IP can be found in any networking textbook, such as [18].

2.1.5. Transport layer

The transport layer provides an end-to-end communication protocol for use by applications. Application layer protocols can be built on top of a transport layer protocol. There are several commonly used transport protocols in the Internet, each offers a set of features that are suitable for different types of applications. Some of these features are: connection orientation or connectionless, flow control, congestion avoidance, reliability, in-order delivery, security, multiple streams, multi-homing, and byte or message oriented transmission. The transport protocols that have had an important role in our work are TCP, UDP, and SCTP. Each of these protocols is explained in more detail below [27] [28].

2.1.5.1 Transmission Control Protocol (TCP)

TCP is described by in RFC 793 [29] as a protocol for delivering bytes with a high reliability between hosts over an IP network. TCP's primary purpose is to provide a trustworthy, accessible logical connection between pairs of processes. The main characteristics of the protocol are that provides a connection orientation, bidirectional, byte-stream with in-order delivery and reliability. This reliability is provided through a variety of retransmission schemes. Additionally, flow and congestion control mechanisms are implemented so that neither the receiver nor the network becomes overloaded.

2.1.5.2 User Datagram Protocol (UDP)

The aim of UDP [30] is to make possible a datagram mode of packet-switched communication via an IP network. UDP offers a procedure to send data between applications with minimal protocol overhead. Its main traits are its connectionless orientation and unreliability, which implies that neither delivery nor duplicate prevention are guaranteed. Therefore, applications that require in-order packet delivery should not use UDP or they must build their own mechanism on top of UDP. However, the reduced overhead of UDP and its simplicity make it ideal for real-time applications.

2.1.5.3 Stream Control Transmission Protocol (SCTP)

SCTP was designed to transport Public Switched Telephone Network (PSTN) signaling messages through IP networks [31] [32]. However, SCTP has become a general purpose Internet Engineering Task Force (IETF) transport protocol. As with TCP, SCTP provides a reliable, connection oriented, bidirectional service with flow and congestion control mechanisms. Nevertheless, there are some differences [33]:

- SCTP uses a four-way handshake during session establishment to prevent the Denial of Service (DoS) attacks to which TCP is subject [34].
- SCTP implements message-based data transfer using a fragmentation and reassembly scheme.
- SCTP supports out-of-ordered data delivery in addition to the in-order delivery.
- A multihoming feature is offered to allow binding a single SCTP association to several IP addresses, providing redundancy and increased fault-tolerance.
- The level of reliability can be reduced by specifying a lifetime for the messages.
- An SCTP association can include multiple individual streams, each one with its own delivery ordering scheme, thus preventing head-of-line blocking [35].

2.1.6. Application layer

The application layer consists of programs that utilize the lower layers to communicate. It is important to note the importance the lower layers have for applications, as each one in turn provides an abstraction of all the layers below it. Therefore, any application can communicate with its counterpart via the Internet by utilizing the TCP/IP stack. Depending on the transport layer protocol selected, the transport could be reliable (for example, when using TCP or SCTP) or unreliable (when using UDP), thus the application designer will need to think about what level of reliability the application requires when selecting which transport protocol to use.

Furthermore, application software may directly be used by an end user or used to provide further functionalities to other applications built on top of this application layer protocol. Hence, many applications can isolate functions into a logical sub-layer, enabling easier later modifications while minimizing complexity.

2.1.7. Unicast, Multicast, and Broadcast

An additional functionality of IP is that a datagram can be sent to many recipients at the same time without the source needing to replicate the transmission for all these destinations. In such a "multicast" (or broadcast) the sender transmits a single packet and the network is responsible for replicating this packet as necessary to deliver it to all the targets. Today the majority of packets that traverse the wider Internet are sent to a single destination. This is called unicast transmission. When more than one receiver is involved we can distinguish between two varieties: multicast and broadcast. If the packet is going to be forwarded to all hosts in the network, then it would be broadcast. It should be noted that due to the effects of such a broadcast packet many networks filter out such packets and do not forward them beyond a certain boundary (for example a subnet or network). On the other hand, if this packet is to be delivered to a specific group of receivers, then the packet would be multicast. Since multicast has been used within our tests we will give a more detailed explanation of this type of traffic below.

In order to support multicast traffic [36] some protocols must be supported and multicast enabled routers must be present in the network. The Internet Group Management Protocol (IGMP) handles group memberships indicating which hosts want to listen to a given multicast address. This indication is sent via IGMP to the adjacent multicast-routers [37]. The Protocol Independent Multicast (PIM) protocol allows communication among multicast-routers so that a multicast packet can reach distant participants [38]. Every multicast packet has a multicast destination IP address, which is in the range of 224.0.0.0 and 239.255.255.255.

Basically, a multicast communication process consists of several steps: all hosts that wish to listen to a specific multicast destination IP address register using IGMP with its adjacent multicast-router identifying the multicast destination IP address that they want to listen to. The multicast router forwards the interest in a specific multicast destination IP address until a path is created from the source to the destination. Now multicast datagrams can traverse the network to the interested receiver(s). Finally, if a listener wants to leave the multicast group it notifies its adjacent multicast-router and the path will be eliminated when there is no one else connected to this multicast-router that is interested in this multicast destination address. An example of a tree created for a multicast communication can be seen in Figure 2.5.

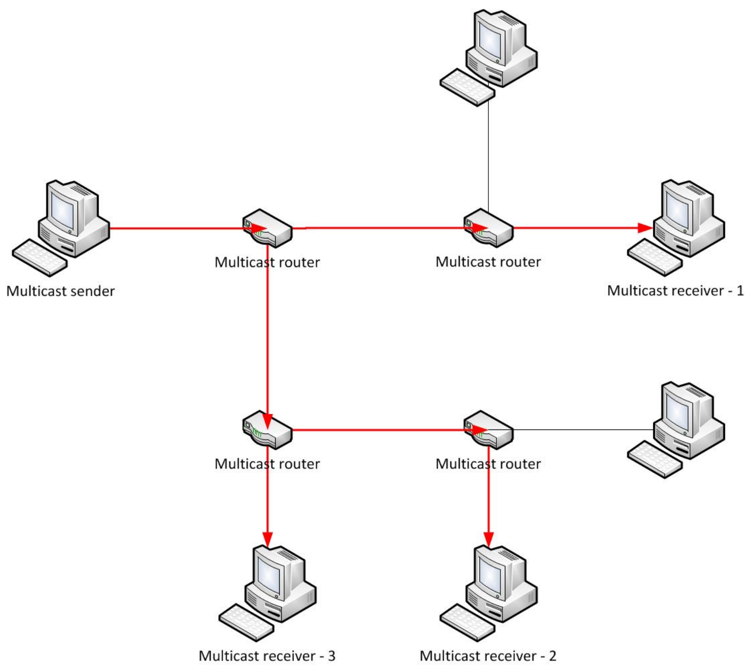


Figure 2.5. An example of a multicast tree.

2.1.8. Tunneling

The concept of tunneling is used when a level of virtualization over an underlying network (such as the Internet) is required in order to connect two or more networks (as if they were directly connected). An example is shown in Figure 2.6. There are many types of tunnel implementations. The choice of tunneling method depends on the network layer over which the tunnel will be set up and on requirements, such as the desired security. A tunnel encapsulates data units of the layer that is intended to be tunneled at one end of the tunnel and delivers them to the other end of the tunnel. At the receiver the encapsulated data units are decapsulated and the data is forwarded as ususal to its destination [39].

When tunneling Ethernet frames over IP [40], as was used in this project, the virtualization provides a virtual link layer connecting the involved networks. Therefore, any Ethernet frame that is sent in one network can go through the tunnel and reach the other end, as if both sites were directly connected to the same LAN. Note that the tunnel does not forward frames that do not need to be forwarded to the other site(s), hence Ethernet over IP (EoIP) acts as if the interconnecting tunnel is a distributed bridge.

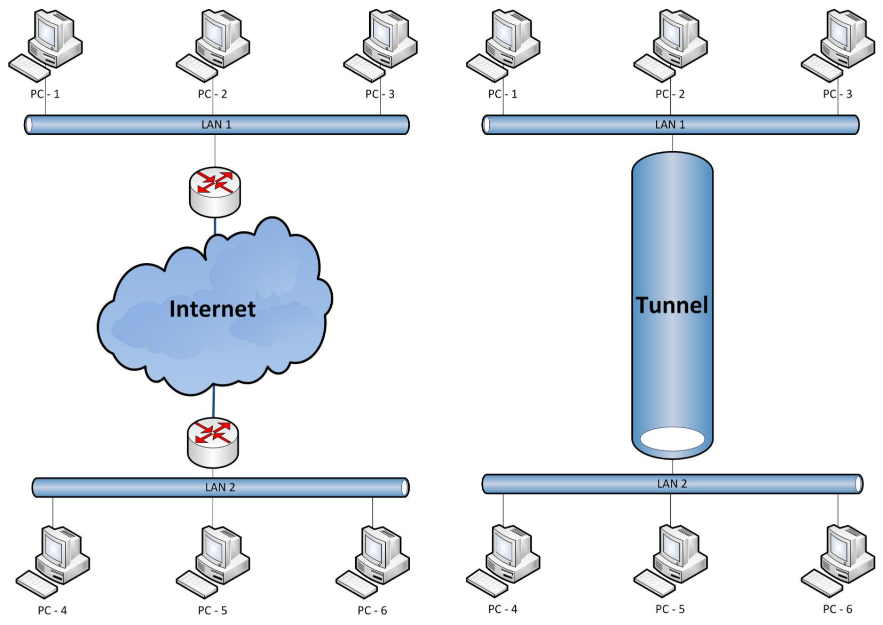


Figure 2.6. Tunneling can be used to virtualize the network connected PC-1 .. PC-6.

2.2. An approach to Cloud Computing

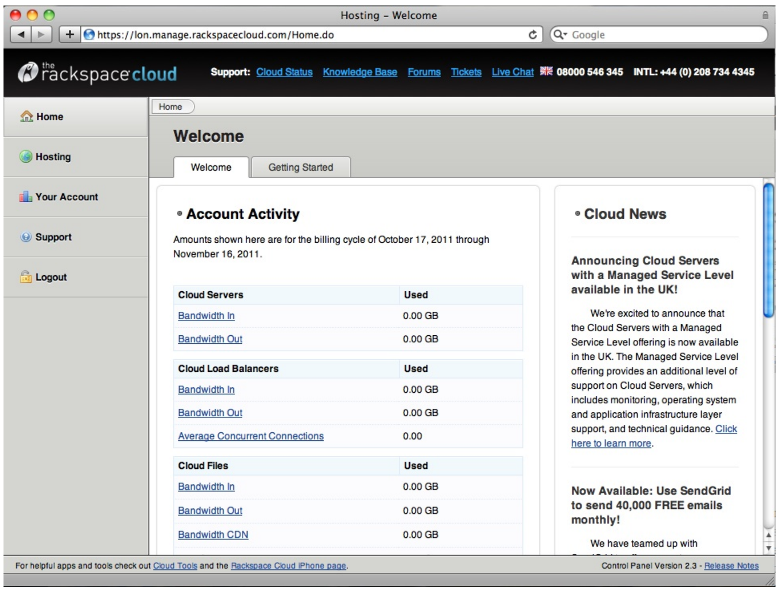
As stated in Section 1.1, Cloud computing can be realized in many different ways, which has caused its lengthwise and crosswise growth in the last years. However, every Cloud has some basic properties. This section summarizes the main characteristics of Cloud computing, then describes the different services and deployments that are found using this technology.

2.2.1. Cloud Computing features

The main features of Cloud Computing are on-demand provisioning, location independent resource pooling, Internet access, elasticity, network monitoring and metering, and outsourced maintenance [11][41][42][43][44]. Below we describe each of these in more detail.

2.2.1.1 On-demand provisioning

In any Cloud the computing resources are commonly ordered and accessed in an on-demand basis through the Internet (or in some cases through an intranet). This capability enables the customer to acquire whatever resources are desired whenever desired with almost complete independence, as if the resources were endless. To ease this process, the Cloud capacity is normally managed through a provider's specific Application Programming Interface (API). In many cases, the Cloud provider offers a very intuitive Graphical User Interface (GUI), such as the one shown in Figure 2.7. Using this GUI a customer simply specifies what is needed by typing and clicking on the interface, then the requested resources will be made available within a short amount of time. The freedom in terms of configuration depends on what level of virtualization the service offers, becoming more restrictive as the service goes up the logical ladder of services (from Infrastructure as a Service (IaaS) to Platform as a Service (PaaS) to Software as a Service (SaaS)). For further description see Section 2.2.7 and [43].


Hosting – Welcome
https://lon.manage.rackspacecloud.com/Home.do
Google
the rackspace cloud
Support: Cloud Status Knowledge Base Forums Tickets Live Chat 08000 546 345 INTL: +44 (0) 208 734 4345
Home
Hosting
Your Account
Support
Logout
Home
Welcome
Welcome
Getting Started

Account Activity

Amounts shown here are for the billing cycle of October 17, 2011 through November 16, 2011.

Cloud Servers	Used
Bandwidth In	0.00 GB
Bandwidth Out	0.00 GB

Cloud Load Balancers	Used
Bandwidth In	0.00 GB
Bandwidth Out	0.00 GB
Average Concurrent Connections	0.00

Cloud Files	Used
Bandwidth In	0.00 GB
Bandwidth Out	0.00 GB
Bandwidth CDN	0.00 GB

Cloud News

Announcing Cloud Servers with a Managed Service Level available in the UK!

We're excited to announce that the Cloud Servers with a Managed Service Level offering is now available in the UK. The Managed Service Level offering provides an additional level of support on Cloud Servers, which includes monitoring, operating system and application infrastructure layer support, and technical guidance. Click here to learn more.

Now Available: Use SendGrid to send 40,000 FREE emails monthly!

We have teamed up with

For helpful apps and tools check out Cloud Tools and the Rackspace Cloud iPhone page.

Control Panel Version 2.3 - Release Notes

Figure 2.7. Rackspace GUI [45]

2.2.1.2 Location independent resource pooling

Processing power, storage, and generally any computing resource are physically located within the Cloud owner's facilities, better known as data centers. The service provider is responsible for managing the available resources to serve multiple consumers at the same time, dynamically allocating and reallocating resources to these customers in an efficient manner depending on the customer's current needs. This dynamic allocation is performed by means of Cloud computing management software, such as Eucalyptus [46] or Open Nebula [47], which enhances the effectiveness of the resources and the elasticity of what resources can be provided at any given time.

Another trait which characterizes Cloud computing is that the consumers do not need to know where the resources are located **unless** it is strictly required for their purposes. However, in some cases, proximity is an issue - for example, to meet a given delay bound, then customers will select which service they get based upon the data center's geographical location or Internet connectivity conditions (e.g. delays, latency, etc.).

2.2.1.3 Internet access

All services provided by means of a Cloud are carried over a high speed Internet connection. The Cloud's Internet connectivity must be able to meet the aggregate instantaneous demands of all of the customers. This connectivity also provides ubiquity to any application or service that the Cloud computing customer may deploy using their allocated resources, thus extending the reach of the customer's application or service to almost any place in the world and to almost any number of end users. Cloud computing makes it possible for anyone with Internet access to consume or benefit from the offered service.

2.2.1.4 Rapid elasticity

Computing capacity is provisioned in an elastic manner, thus offering the customer the possibility of acquiring and releasing resources without any significant delay. Cloud providers are committed to carrying out clients' orders in a short amount of time, which is normally specified in their Service Level Agreement (SLA). If the provider fails to meet their SLA, then they must compensate the customer by reimbursing part of the customer's payment or perhaps even pay a penalty.

This capability enables customers to scale their own service according to their own needs in a fast and effective way, while at the same time avoiding the costs of unused computing capacity. Such an advantage becomes crucial, especially for start-up companies, as while it increases OPEX it reduces CAPEX. This lowers the risks to which these companies are exposed at the outset of their operations.

2.2.1.5 Pay-as-you-go billing model

Clouds are constantly monitored in order to offer the agreed service to the customers of the Cloud service. Thanks to the combination of this thorough monitoring and the elasticity offered by Cloud computing, customers pay for only the resource(s) that they are really using through a pay-as-you-go charging scheme. However, some providers also offer other charging plans (e.g. monthly/annual payments).

2.2.1.6 Outsourced maintenance

It is also worth mentioning that Cloud computing providers must maintain their equipment, this relieves their customers of this task. Additionally, data centers are upgraded on a constant basis with new equipment to enhance the Cloud service's performance. It is important to note that these efforts are essential for any company, they are also expensive, therefore delegating them to third parties who specialize in these tasks can lead to a significant cost reduction in this area.

2.2.2. Management responsibilities

As we have previously mentioned, the distribution of responsibilities depends on the service that is contracted for. Cloud providers offer managed solutions where all equipment and software, even when its correct operation does not depend on them, are monitored. These additional outsourced management options ease the user's Cloud experience while introducing extra costs. Therefore, it is up to the Cloud provider's customers to decide whether they will manage their own Cloud servers or they want someone else to do it for them (for which they will pay). Again one of the advantages of outsourcing this management is that it enables cost reductions due to increase scale, but it also means less control of the resources.

2.2.3. Customization

Especially when acquiring an IaaS service, the hardware, OS, and other specifications are issues for the customer. For that very reason many providers offer different solutions so that all their different customers needs can be satisfied. One of the main questions of a customer is whether the allocated equipment is dedicated to them as a single customer or shared, as they might have some concerns about security with shared environments, thus dedicated resources might be the only solution for them. Additionally, custom options such as running a specific release of a specific OS and details of the network configuration may be necessary in order to carry out specific tasks. In PaaS, the support for specific Integration Development Environments (IDE) and programming toolkits will ease the task for software developers as well as increase the number of potential clients.

2.2.4. Service Level Agreement (SLA)

When a Cloud service is acquired an important agreement is signed between the two contracting parties where all the details regarding the service are laid out. According to [48] and [49], where a trust model in order to evaluate Cloud services is offered, the main issues an SLA should address are:

- Fully describe the service so that the user knows exactly what to expect it and no unrealistic expectations are implied.

- Indicate the service's level of performance and the benchmarks, parameters, and metrics used to asses this performance.
- Present a problem management scheme that reduces the negative impact of possible incidents.
- Guarantee disaster recovery and business continuity for critical systems.
- Specify which are the customer's responsibilities within the business relationship.
- Describe the service's warranties and the penalties imposed when the requirements are not met.
- Provide the privacy and security policies.
- Provide details concerning a potential termination of the contract (by either party).

2.2.5. Cloud Security

As indicated by Zhou, et al. [50] security is the main customer concern, even above performance and availability, when considering contracting for a Cloud computing service. Therefore, many issues need to be assessed before uploading information to a Cloud, as the use of the Cloud implies the potential for various threats[51]. According to Jansen [52], the categories of security concerns attached to Cloud computing are:

Trust: When an organization makes use of Cloud resources it places a large amount of trust on the service provider, considering the fact that private data and processing are being handled outside of the customer's premises [53].

Architecture: Cloud infrastructure traits, such as virtualization, imply the existence of potential additional weak points that could be subject to attacks. Additionally, both sides of the processing (i.e., both client and server) need to be protected to guarantee secure operation.

Identity management: Authentication must be required when accessing information located within a Cloud's resources. Additionally, access control is important to provide the privacy required.

Software Isolation: The multi-tenancy concept with its sharing of physical resources generates new sources of threats among different users in a Cloud environment (see for example [43]).

Data protection: Data isolation is critical to address privacy concerns. Additionally, backup policies and data location are of great importance when dealing with valuable user data.

Availability: Outages in critical situations may lead to major trouble for the Cloud's users, as these users are relying on the service provided by the Cloud.

2.2.6. Private, public and hybrid Clouds

A Cloud can be private, public, or hybrid depending on its access restrictions [41][42][43]. In other words, whether a Cloud belongs to one of these groups is related to the boundaries of control and trust, rather than to who pays for the service or who actually owns the equipment [44][54].

Public Cloud

In this model the Cloud infrastructure is owned by an organization that sells services to third parties. The customer, which can be any company or individual with computing needs, takes advantage of the infrastructure abstraction that such a Cloud provides, thus saving the expenses of building a new private data center (that would require investing in hardware, Operating System (OS), network design, space provisioning, etc.)

The service diversity offered by public Clouds is increasing as Cloud computing evolves. Today there are many ways in which customers can exploit public clouds, depending on their needs and the service standard they require.

Private Cloud

The purpose of a private cloud is to serve just one organization, thus automatically providing the security and access restrictions that such an exclusive arrangement offers. A Private Cloud can be installed and managed directly by the organization itself, although a third party may also be in charge of operations, thus relieving shifting the operations to a third party (who might specialize in operating such clouds). The equipment can be located either at the customer's premises or under the supervision of a collocation provider.

Hybrid Cloud

Hybrid Clouds comprise a combination of public and private Clouds. These hybrids inter-operate in a common infrastructure managed by a single entity. Hybrid Clouds are limited by the technology which provides the interoperability among the different participating clouds.

Community Cloud

Some organizations with similar interests, concerns, and requirements share resources in a common infrastructure, thus reducing the individual organization's costs and taking advantage of the statistical multiplexing of the actual resources. The management of a community Clouds may be done by a third party or by the organizations themselves.

2.2.7. Cloud service types

Cloud services are divided in three big groups [41][42][43], depending on the level of abstraction the customer wishes to obtain [44][54]. These services can also be seen as a layered structure, where higher layers depend on lower layers to provide their service, as shown in Figure 2.8. However, cloud services do not always need to follow this pattern, as upper layer services can also exist **without** being implemented over lower Cloud layers. Further details of these different types of cloud services are provided in the following paragraphs.

Infrastructure as a Service (IaaS)

IaaS is the lowest level of abstraction offered by a Cloud service provider. It can provision storage, processing, network, and any computing resource over which the customer runs their own customized operating systems and software. The service provider controls and manages the underlying hardware and hypervisor layers while the customer is responsible for everything above the hypervisor. However, some providers offer a managed service where they perform monitoring and other maintenance activities for the customer.

Platform as a Service (PaaS)

PaaS provides an additional level of abstraction releaving the client from worrying about unnecessary tasks. This abstraction is provisioned by means of middleware programs running on top of the physical infrastructure. This middleware is specially customized to scale up specific types of applications. These platforms are commonly used by software developers in order to host their applications, so that they can be utilized by end-users in a flexible and effective way. The advantage of this type of Cloud service is that the customer does not need to worry about anything other than their own application, therefore the rest of the system becomes Cloud provider's concern and responsability.

Software as a Service (SaaS)

SaaS is currently the highest level of abstraction commonly offered within the Cloud market. SaaS refers to the direct consumption by end-users of applications supplied by software developers through thin client interfaces, such as a web browser. These applications may be running on top of a PaaS or over the developers' own servers, although the former option is becoming more and more frequently deployed.

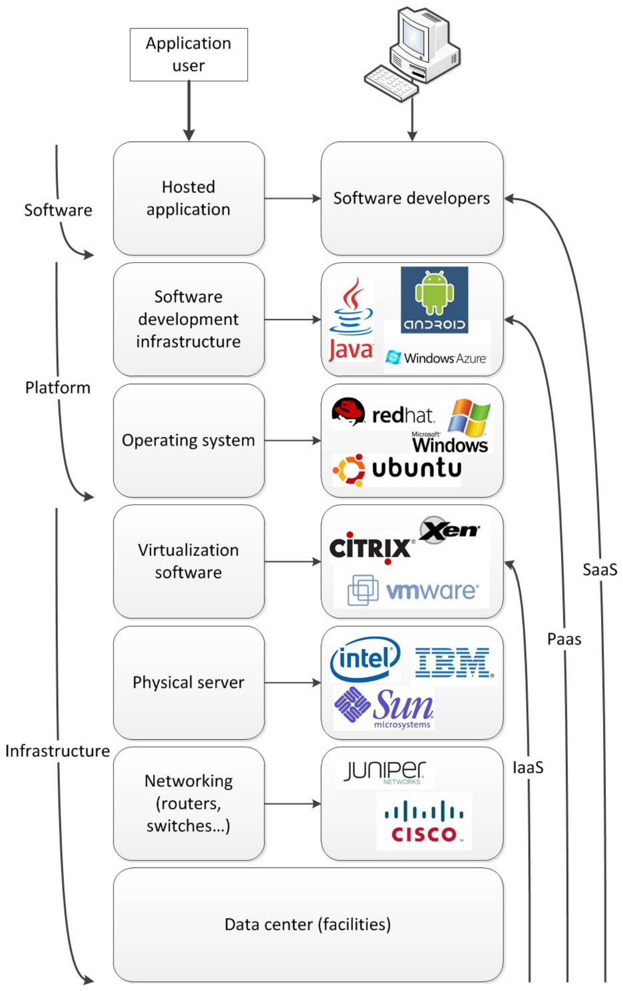


Figure 2.8. Cloud Layered Model Structure. Adapted from [55]

2.3. Telecom networks

The telecommunication application to be tested in this thesis work is a Mobile Switching Center Server (MSC-S), which is exclusively deployed in mobile core networks in order to implement call service's signaling functions. Therefore, in this section we will give a generic description of the most widely deployed mobile network architectures, the Global System for Mobile Communications (GSM) and the Universal Mobile Telecommunications System (UMTS). This section focuses specifically on the elements that intervene in the call service provisioning, thus data service network nodes are not covered. Finally, we will explain the evolution these architectures have experienced over the years, with a special focus on the core network.

2.3.1. Mobile network architectures

In this subsection, a brief description of the two main mobile network architectures' structure is given. This information provides a general overview of these two architectures' elements that are involved in providing a voice call service.

2.3.1.1 Global System for Mobile Communication (GSM)

GSM is a second generation (2G) digital mobile network architecture standard developed by the European Telecommunications Standard Institute (ETSI) [56]. GSM was created to replace the first generation analog mobile networks. The main function of GSM was initially to provide a circuit switched network optimized to support mobile telephony traffic. GSM is a cellular network, which means that the service area is divided into cells. A mobile terminal connects to the network through a cell's Base Transceiver Station (BTS) and, if the terminal is successfully authenticated and authorized, then the cell is responsible for providing network connectivity to this terminal.

GSM standard's topology is divided into three main systems: the Network Switching Subsystem (NSS), the Base Station Subsystem (BSS), and the Network Management Subsystem (NMS). These systems are composed of smaller units so that different functions are realized by different functional entities in the network. The Figure 2.9 illustrates the topology of a GSM network and the connections between the different functional units that comprise it.

The NSS is responsible for call control, billing data collection, mobility management, signaling, and subscriber data handling [57]. The NSS consists of the following units:

Mobile services Switching Center (MSC)

The MSC node is responsible for the setup, supervision, and release of calls in the GSM mobile network as well as for handling SMSs and managing terminals' mobility. It also collects call billing data in order to send it to the Billing Gateway, which processes this data to generate bills for the operator's subscribers. Additionally, a MSC may act as a bridge between its own GSM network and the Public Switched Telephone Network (PSTN), or another operators' GSM network, such a MSC is called a Gateway MSC.

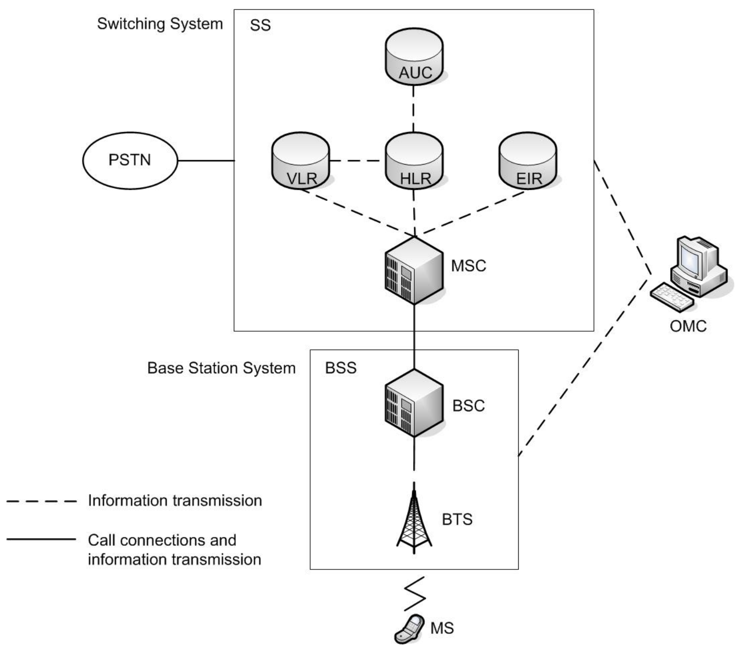


Figure 2.9. GSM network topology.

Home Location Register (HLR)

The HLR database stores and manages all mobile subscriptions belonging to a specific operator. The HLR is the main database containing permanent subscriber information for a mobile network. It stores all the information related to the subscriber: the International Mobile Subscriber Identity (IMSI), the Mobile Station Integrated Services Digital Network (MSISDN) which is the subscriber's phone number, information about the subscriber's supplementary services, authentication data, and location information.

Visitor Location Register (VLR)

The Visitor Location Register (VLR) is another database in a mobile communication network associated with a specific MSC. It contains information about all mobile terminals currently located in this MSC's service area. The VLR maintains the temporary subscriber information needed by the MSC to provide service (e.g. route a call to the correct base station) to visiting subscribers. The information in the VLR changes dynamically, as the subscribers move from cell to cell and network to network. In order to obtain service a mobile terminal registers with the network and this information enables the MSC to determine if it can be authenticated and if it is authorized for each specific service - and it also enables calls to and from this mobile terminal to be effectively handled and routed. The VLR can be seen as a distributed subset of the HLR as, when a mobile station roams into a new MSC's service area, the VLR connected to this MSC requests data about the mobile station from the HLR and stores the response in the local VLR. If the mobile station makes or receives another call without changing its service area, then the VLR will use the subscriber's information that it already has for authentication and call set-up. The database entry of the subscriber may be deleted once the subscriber leaves the MSC service area (note that the deletion may or may not occur quickly depending upon the probability that the subscriber may return to this service area).

Authentication Center (AUC)

The Authentication Center (AUC) is a database that is connected to the HLR. This database provides the HLR with the subscriber's authentication and encryption parameters that are used to verify the subscriber's identity and to ensure the confidentiality of each call. The AUC protects network operators from fraud and is used to authenticate each Subscriber Identity Module card (SIM card) when a terminal with this SIM card attempts to connect to the network (typically when the phone is powered on in the network operator's service area). Once the authentication is successful, an encryption key is generated, and this key is used in all communications between the mobile terminal and the network.

Equipment Identity Register (EIR)

The Equipment Identity Register (EIR) is an optional database that contains mobile equipment identity information which helps to block calls from stolen, unauthorized, or defective mobile stations. Operators can maintain three different lists of International Mobile Equipment Identities (IMEI) in their EIR: a white list containing valid mobile terminals, a grey list where dubious mobile equipment is included, and a black list containing the mobile devices to which the service is to be denied.

The Base Station Subsystem (BSS) is responsible for all radio-related features. Typically, a MSC controls several BSSs, covering a large geographical area that is divided into many cells [57]. This subsystem is composed of: Base Station Controller (BSC) and Base Transceiver Station (BTS).

Base Station Controller (BSC)

The Base Station Controller (BSC) provides all the radio-related functions and physical links between the MSC and the Base Transceiver Station (BTS). It also implements functions such as handover, cell configuration data, channel assignment, and control of radio frequency and power levels in each connected BTS.

Base Transceiver Station (BTS)

The Base Transceiver Station (BTS) handles the radio interface to the mobile station. It facilitates the communication between the mobile devices and the network. The BTS is the radio equipment (transceivers and antennas) needed to serve each cell in the network. Normally, a group of BTSs are controlled by a BSC.

Finally, the Network Management Subsystem (NMS) is the entity via which the network operator monitors and controls the whole system. The Operation Support System (OSS) can be divided into a two-level management function formed by a Network Management Center (NMC) responsible for the centralized control of the system, and subordinate Operation and Maintenance Centers (OMCs) focused on regional maintenance issues. The functions of the NMS can be divided into three groups [57]:

Fault management to ensure the correct operation of the network and a rapid recovery when failures occur.

Configuration management to maintain updated information about operation and configuration of the network elements.

Performance management The NMS collects data from the network elements so that the actual performance of the network can be compared to the expected performance.

2.3.1.2 Universal Mobile Telecommunications System (UMTS)

UMTS is a third generation (3G) mobile cellular technology standard developed by the Third Generation Partnership Project (3GPP) [58]. This standard defines a complete system composed of the radio access network and the core network. The core network initially preserved the GSM architecture in order to enable a graceful evolution of the GSM networks. Regarding the UMTS radio access network, two new elements were introduced to replace GSM's BSCs and BTSs respectively: Radio Network Controller (RNC) and Node B. The remaining GSM network elements are compatible with the UMTS network. Figure 2.10 illustrates the UMTS network architecture.

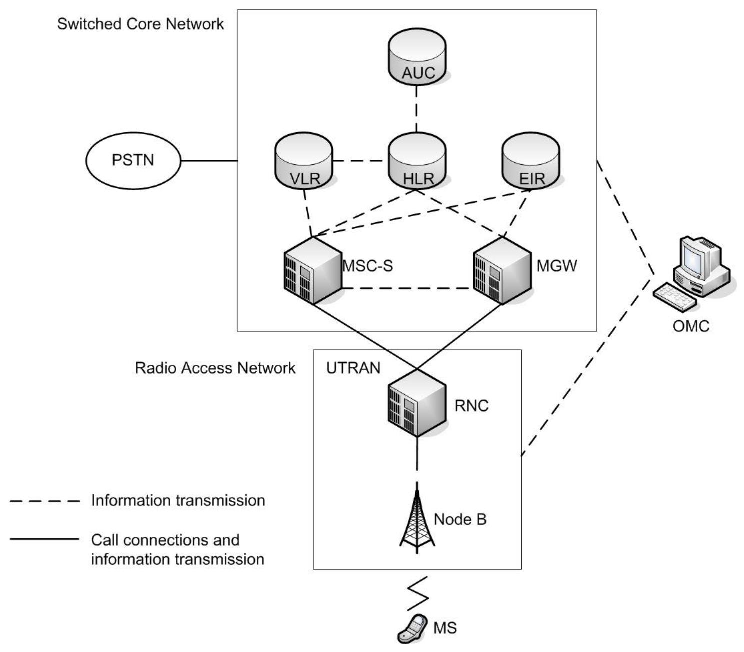


Figure 2.10. UMTS network topology.

2.3.2. Mobile core network's evolution

This subsection describes the evolution ofmobile telecommunication system's core networks with respect to their architecture and transport technology from the introduction of GSM to current third generation implementations.

2.3.2.1 Initial GSM architecture

The initial mobile core network architecture deployed along with the GSM network was composed by the NSS elements described in Section 2.9. The mobile core network provides connectivity among the mobile network nodes and terminals attached to other circuit switched telephony networks. The technology implemented in that initial GSM system utilized Time-Division Multiplexing (TDM) transmission techniques for both signaling and payload traffic [59]. Figure 2.9 illustrates a GSM network where both access and core networks are depicted.

Data service was later introduced in the GSM network through standards such as General Packet Radio Service (GPRS) [60] and Enhanced Data rates for GSM Evolution (EDGE) [61]. In order to provide this service the existing TDM backbone was exploited to implement a packet-switched (PS) domain together with the circuit-switched (CS) domain that supported the voice call service.

2.3.2.2 UMTS deployment

When the UMTS standard started to be deployed the GSM legacy network was preserved. The first UMTS standard release, 3GPP Release 99, introduced modifications in the access network by adding the UMTS Terrestrial Radio Access Network (UTRAN) nodes mentioned in Section 2.3.2.2. UTRAN nodes provided support for a variety of radio technologies such as Wideband Code Division Multiplexing [62] and High Speed Packet Access [63]. This release also introduced modifications in the core network, splitting the CS and PS domains. The CS domain continued to use TDM technology, while the PS domain started being implemented with IP technology [59]. The resulting network structure after these modifications is depicted in Figure 2.11.

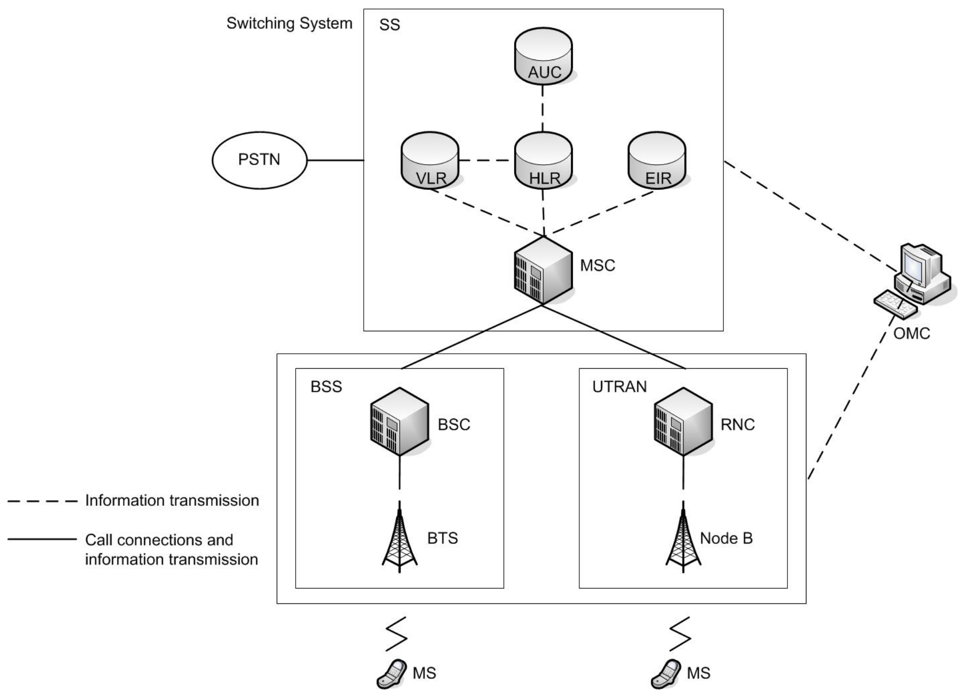


Figure 2.11. UMTS and legacy GSM network topology.

2.3.2.3 Layered architecture and softswitch solution

The main change included in the 3GPP Release 4 standard was the introduction of a layered architecture in the CS domain [8]. This architecture split, both physically and logically, the functions responsible for network signaling, control and management, and the functions concerning the transport of service data [64]. This division involves the replacement of the legacy MSC by two new nodes: the Mobile Switching Center Server (MSC-S) and the Media Gateway (MGW). The purpose of the MSC-S is to handle the call signaling and control, while the MGW realizes the actual circuit switching [59]. In addition, due to the different requirements associated with control and voice traffic, MSC-S nodes can be centralized whereas MGW nodes should be placed close to the radio access network nodes in order to optimize utilization of resources.

The implementation of this layered architecture lead to the softswitch solution [64] commonly deployed by telecommunication operators. In addition, the softswitch solution involves the transition to a common all-IP backbone for the CS and PS domains. In this shared backbone the Multi-Protocol Label Switching (MPLS) protocol is used to provide fast packet transport and traffic separation through the network. Figure 2.12 illustrates the mobile network architecture after applying the softswitch solution to a mobile core network.

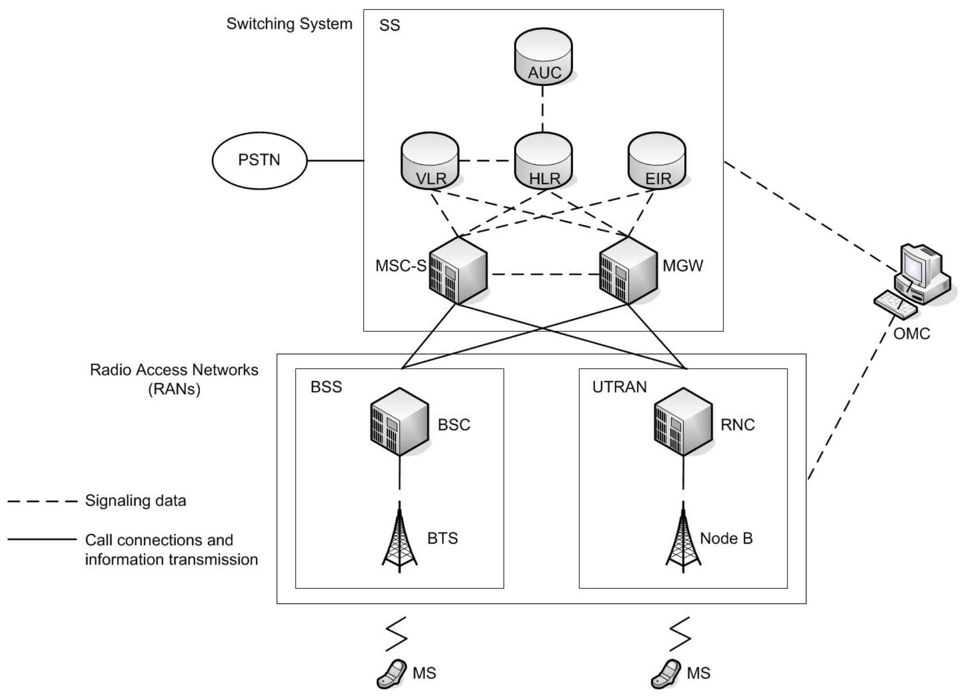


Figure 2.12. UMTS and legacy GSM softswitch network topology.

2.3.2.4 Packet-switched core

The most recent evolutionary step for the mobile core network is the transition to the System Architecture Evolution (SAE) core, also known as the Evolved Packet Core (EPC), as specified in the 3GPP Release 8 [59]. This core network implementation, together with the Long Term Evolution (LTE) radio technology, provides support for higher traffic rates through a flat all-IP architecture [5]. This combination is also known as the fourth generation (4G) of mobile networks. Another purpose of SAE/EPC is to provision a full IP-based PS backbone, thus eliminating the legacy CS domain.

In addition, the SAE architecture reduces the number of nodes in the network, hence reducing the latency, simplifying network management, and enabling an easier future evolution [59]. This reduction is accomplished by upgrading the network with the following new nodes [65]:

Evolved Node B (eNB) This node manages the radio interface to the mobile terminals.

Serving Gateway (SGW) and Packet Data Network Gateway (PDN GW) The SGW node routes and forwards user data packets while the PDN GW is responsible for providing connectivity to external packet networks. These two nodes can be implemented as a single element in the network.

Mobility Management Entity (MME) This network element implements the control functions of the access network, such as terminals' mobility management and handover handling, and is also responsible for selecting the SGW.

These elements are compatible with different access network technologies such as 2G, 3G, WiFi[1], and WiMax [6], as well as with architectures based on legacy CS domains. In addition, the most widely deployed option to implement SAE/EPC is an overlay deployment in order to take advantage of this compatibility.

Several solutions have been proposed to implement the real-time services over this PS domain [6]. One of the most important is the implementation of an IMS platform. To realize such a solution, the IP core network must be upgraded in order to meet the end-to-end QoS requirements characteristic of real-time services, such bounded latency and appropriate minimum throughput [65]. In addition, the IMS architecture provides support for different multimedia services in addition to real-time applications and allows interworking across both fixed and mobile access networks by defining an horizontal control plane that decouples the service layer from the access network [7]. The elements from which an IMS implementation is composed are:

Call Session Control Function (CSCF) It is responsible for processing the Session Initiation Protocol (SIP) signaling in order to provide session control for IMS terminals and applications.

[1]WiFi is the branding for interoperability of IEEE 802.11 devices.

Home Subscriber Server (HSS) This database contains the subscriber information necessary to provide a service.

SIP application server These servers host the applications through which the services are offered in the network.

Breakout Gateway Control Function (BGCF) It decides how to route sessions from the IMS network to a legacy mobile CS network or to a PSTN.

Media Resource Function (MRF) This network element provides media services in the local network and is responsible for processing streams in a media service.

Session Border Controller (SBC) This node is an IP to IP gateway between the IMS network and other operators' networks.

Media Gateway Control Function (MGCF) It provides control functions over the media resources when there is a traffic transaction between the IP network and other technologies' networks, such as TDM.

Media Gateway (MGW) This node is controlled by the MGCF and enables the data flow between different networks, providing translation of transport formats and transcoding when necessary.

2.4. Ericsson MSC-Server Blade Cluster

The Ericsson Mobile Switching Center Server (MSC-S) [66] is one of the main parts of Ericsson's Mobile Softswitch solution [64]. The MSC-S controls all circuit-switched call services, the user plane, and the MGWs. However, the MSC-S has been replaced by a state-of-the-art solution, called the MSC-S Blade Cluster (MSC-S BC) [66]. In the MSC-S BC the MSC-S's functionality is implemented on several generic processor blades that work collectively as a cluster. All the components of the Ericsson MSC-S BC are implemented as a racked architecture which consists of one or two cabinets, with the second cabinet being optional. The first cabinet hosts the mandatory elements, while the second contains an optional expansion of MSC-S blades in order to support greater capacity. In addition, every MSC-S blade has a locally attached VLR register (see Section 2.3.1.1).

The MSC-S BC has great benefits in terms of power consumption, as it reduces power consumption by up to 60 percent compared to the earlier MSC-S, and reduces the physical footprint of an MSC by up to 90 percent (again as compared to the MSC-S). This is accomplished thanks to the MSC-S BC's advanced components. This system also achieves high reliability with commodity components by exploiting redundancy.

2.4.1. Architecture

As can be seen in Figure 2.13 the main components of the MSC-S BC are the MSC-S blades, a signaling proxy (SPX), an IP load balancer, an I/O system, and a Site Infrastructure Support (SIS) system:

SPX: This element is responsible for distributing external SS7 signaling traffic over the MSC-S blade cluster so that it can be processed. The traffic distribution to the blades is done on an algorithmic basis (e.g. Round Robin scheduling algorithm).

IP load balancer: The IP load balancer is an interface for non-SS7-based IP-signaling traffic and it distributes external IP signaling to the MSC-S blades. This element was not used in this thesis project, as all the tests were conducted using SS7-based IP-signalling traffic as requested by our employers.

I/O system: The I/O system is connected to the operation support system (OSS), and handles all the data transfers to and from the MSC-S blades and SPXs.

SIS system: The SIS system is also connected to the OSS, and it provides the I/O system to Ericsson's Infraestructure components.

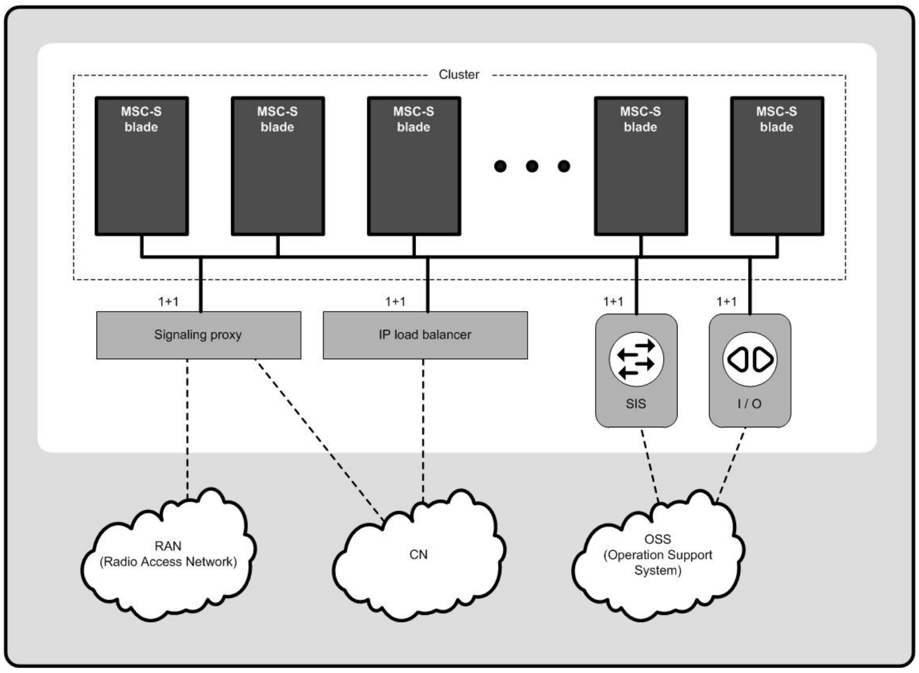


Figure 2.13. MSC-S Blade Cluster. Adapted from [66]. Note that the notation 1+1 means that the device is configured such that there is one primary device and at least one secondary device (to provide redundancy).

2.4.2. Redundancy

Inside the MSC-S BC, the redundancy is implemented in different ways. Some subsystems have a 1+1 redundancy configuration, such as the network signaling interfaces and the I/O system. This form of redundancy consists on having two components of each type, but only one is working at a time. If the component that is working becomes unavailable, the other takes over to ensure that the service is not adversely affected. The standby element takes over whether the primary component's unavailability is planned or unplanned.

In contrast, the redundancy scheme in the MSC-S BC is n+1. This means that every blade can assume every blade's role inside the cluster. The subscriber records are kept in two blades to ensure access to them, if any of the blades become unavailable this data will not be lost and will be available to continue operation. Even in the case of multiple blades' failures, the cluster will remain fully operational, with only a reduction in capacity (due to the failed blades), and potentially a minor temporary loss of subscriber records in the VLR registers (those that have not yet been synchronized from one of the two blades maintaining these records to another blade), which would need to be fetched from the HLR.

Additionally, to achieve this redundancy in the MSC-S BC the system utilizes fault-tolerant middleware that ensures that the blades share a consistent view of the cluster configuration at all times. Furthermore, the static configuration data (needed to execute a requested service) is replicated in every blade, while dynamic data (subscriber records or the state of external traffic devices) is replicated in two or more blades.

One of the most important benefits of n+1 redundancy is the potential to isolate one MSC-S blade from traffic to perform maintenance activities on it, enabling zero planned cluster downtime.

2.4.3. Scalability

The MSC-S BC was developed with scalability in mind. In order to increase system capacity one simply adds MSC-S blades to the cluster. This is possible as the shared cluster components have been designed to support a wide range of cluster capacities, from very small to very large.

The specific MSC-S blades that are deployed are not visible to neighboring network nodes, such as the HLR, RNC, or BSC. Because of this, blades can be added without interrupting the cooperation between these network nodes. Moreover, the MSC-S BC has the ability to adapt its internal distribution to a new blade configuration without human intervention. This means that few manual steps are needed to add a blade to a running system. The blades automatically organize themselves into a new internal distribution scheme because of a new cluster configuration, and they replicate all the necessary data to the newly added blade. All these configuration and redundancy activities run in background, so they have no effect on the normal cluster capacity or availability. After several minutes of preparation and testing, the blade is available for activating to support user traffic and becomes part of the cluster.

Chapter 3

Ericsson MSC-S BC in a Cloud environment

This chapter contains the fundamental elements of the thesis project. First, the topologies of the Ericsson MSC-S BC cloud solution are described. Second, the motivation and feasibility of practical implementations are analyzed by taking into consideration both their theoretical advantages and disadvantages, and the impact on performance they might cause. Finally, based upon the previous step, a conclusion was drawn regarding the realization of the different designs.

3.1. Topologies

The Ericsson MSC-S BC is traditionally implemented in a racked architecture, as stated in Section 2.4. The Ericsson MSC-S BC cloud solution deployed in this thesis work project can be realized with several different topologies. However, a common characteristic of these topologies is that all of these topologies utilize external servers to implement the Ericsson prototype MSC-S blades. Another design decision was to locate the rest of the Ericsson MSC-S BC elements (SPX, SIS, etc...) on-premises in a rack without modifications from the system racked implementation. In addition, all the remaining elements necessary for the simulated mobile network to work (HLR, MGW, BSC, RAN, etc...) were also located on-premises.

This section describes the different generic topologies that were designed, configured, and tested during the project. In order to simulate the network impairments characteristic of real Cloud environment, tests were conducted introducing network penalties, such as packet loss and delay. In addition, a virtualization layer was added in the external servers. The test details, configurations, and results are in Chapter 4. Additionally, the different designs were implemented following a bottom-up approach in terms of complexity so that issues that might arise could be tackled in a simpler manner. The tested topologies are:

- A hybrid cluster consisting of both on-premises racked MSC-S blades, and off-premises prototype MSC-S blades.
- A completely external cluster composed of just prototype MSC-S blades.
- An external cluster composed of cluster partitions geographically remote from each other, and formed by prototype MSC-S blades.

3.1.1. The hybrid cluster

The purpose of this design was to become more comfortable with the Ericsson MSC-S BC system, and to demonstrate the correct operation of the system when placing a prototype MSC-S blade in an emulated Cloud environment (outside the racked architecture). Tests in a simulated traffic case were conducted in order to verify the system's correct performance with this network configuration (see Section 4.2.1). Figure 3.1 depicts the topology of an Ericsson MSC-S hybrid blade cluster, where a prototype MSC-S blade is implemented on an external server located outside the rack.

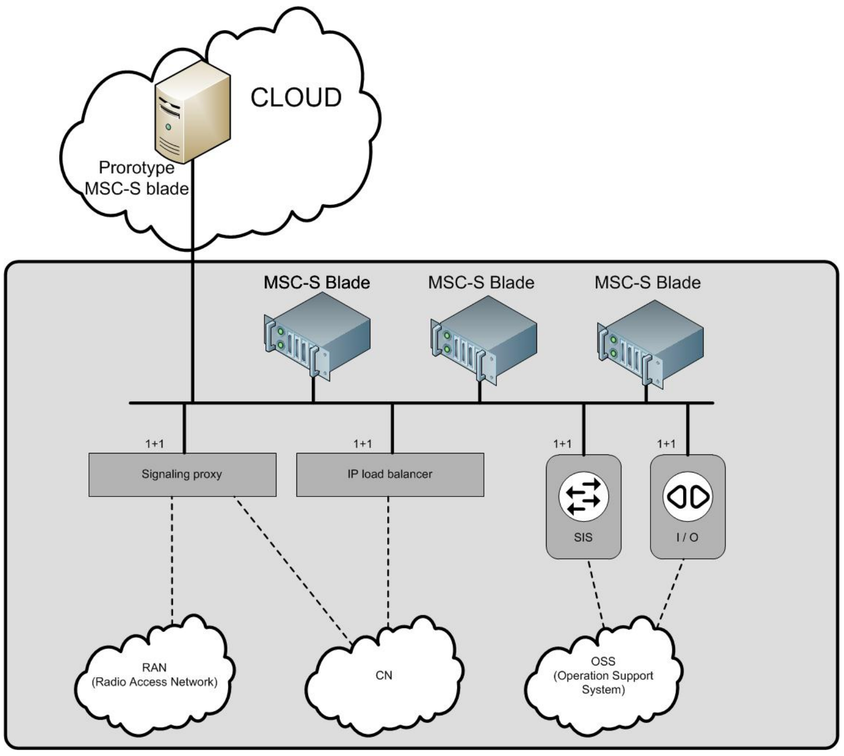


Figure 3.1. Ericsson MSC-S hybrid cluster topology.

3.1.2. The external cluster

This design consists of an Ericsson MSC-S BC implementation whose only MSC-S blades are prototype MSC-S blades located in a emulated Cloud environment. A simulated traffic test case was conducted while network impairments were introduced between the prototype MSC-S blades and the system. The purpose of these tests was to verify the correct operation of the cluster protocols in presence of network impairments as well as the system's stability with this network configuration (see Section 4.2.2). In addition, the Ericsson MSC-S BC fault recovery schemes were tested using this topology (see Section 4.3.2).

The correct operation of this topology would assure that this cluster externalization could be replicated in a real Cloud environment despite the network impairments. The external cluster topology is represented in Figure 3.2.

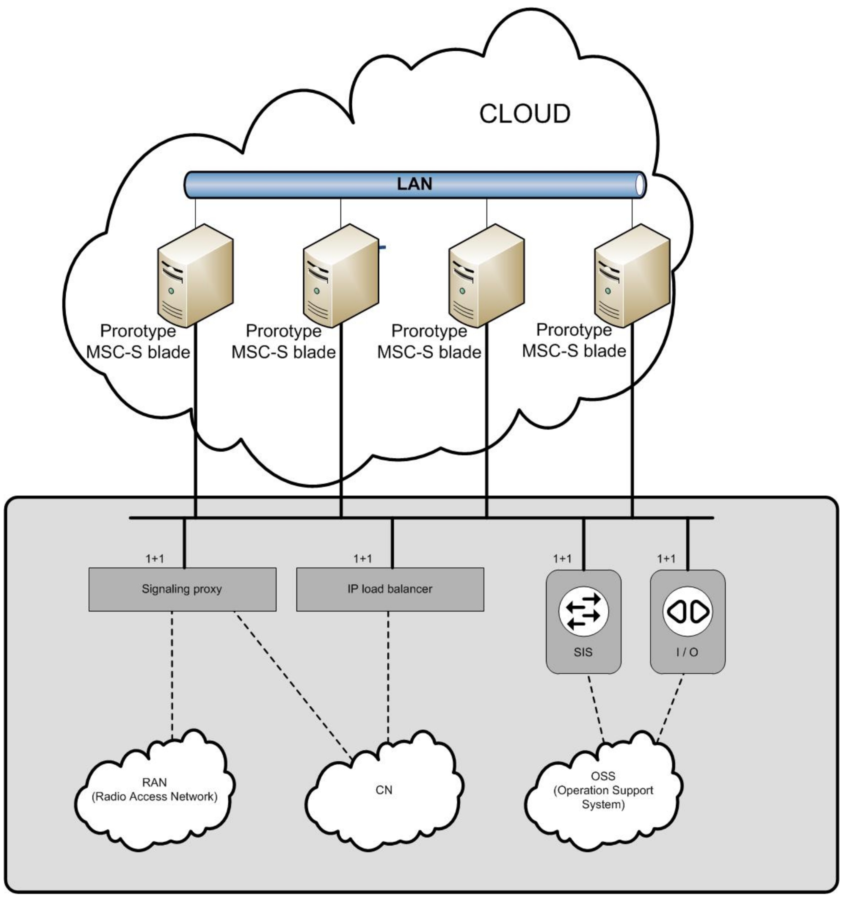


Figure 3.2. Ericsson MSC-S external cluster topology.

3.1.3. The split cluster

The split cluster configuration consists of an Ericsson MSC-S BC implementation whose only MSC-S blades are prototype MSC-S blades located in several geographically remote emulated Cloud computing environments. A simulated traffic test case was conducted while network impairments were introduced between the prototype MSC-S blades and the system, and between the different geographically remote locations where the prototype MSC-S blades were located. The purpose of these tests was to verify the correct operation of the cluster protocols in presence of this combination of network impairments in the system, as well as the system's stability with this network configuration (see Section 4.2.2). In addition, the Ericsson MSC-S BC fault recovery schemes were tested using this topology 4.3.2.

The correct performance of the system using this topology would lead to address the centralization issue stated in Section 1.2. Figure 3.3 illustrates the Ericsson MSC-S split cluster topology.

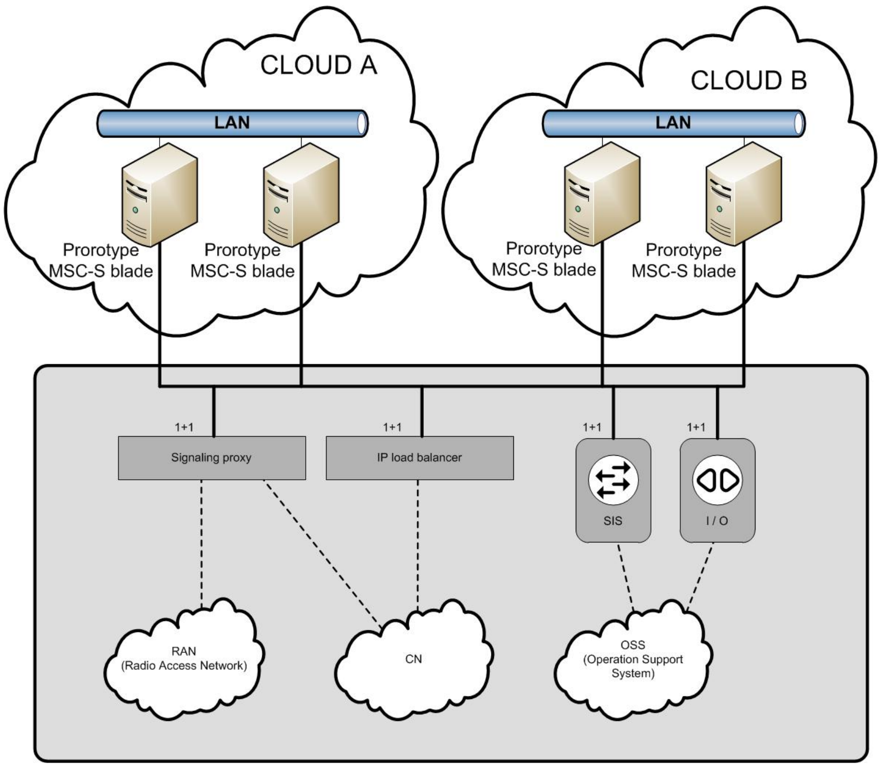


Figure 3.3. Ericsson MSC-S split cluster topology.

3.2. Design considerations

The purpose of this section is to present the motivations associated to the different implementations of the Ericsson MSC-S BC using prototype MSC-S blades located in a Cloud environment presented in Section 3.1. Additionally, the drawbacks that might occur in such an implementation are also presented. Finally, a conclusion regarding the feasibility of a practical realization of these topologies is drawn.

3.2.1. Motivations

Some of the advantages of a Cloud based implementation directly follow from the benefits provided by cloud technology. We have concluded that current Cloud developments enable telecommunication operators to use cloud resources for telecom services provisioning. The Ericsson MSC-S BC is just one of the many components of a telecom network that could be deployed using this technology. In addition, due to the increasing Internet capacity, through which cloud services are provisioned, some of the initial QoS concerns that prevented operators from utilizing this approach have been relaxed. Table 3.1 enumerates the main motivations for a Cloud based implementation, and divides them in two groups depending on their origin: cloud computing and Internet related. Additionally, a short clarification of these motivations is given below the table.

Table 3.1. Motivations to implement an Ericsson MSC-S BC in the Cloud

Motivations	
Cloud computing	High computing capacity High memory instances Geographical redundancy Elasticity and scalability
Internet	High Internet throughput Network redundancy

High computing capacity: In order to run applications, such as a MSC-S, a lot of computing capacity is necessary. In order to run prototype MSC-S blades of the Ericsson MSC-S BC 4-core x86-based machines were necessary to provide a good performance. As it can be seen later on in Chapter 5, cloud computing providers offer sufficient computing capacity for our purposes.

High memory instances: The requirements of telecom applications in terms of memory are substantial. For example, the prototype MSC-S application tested in this project needs 16 GB of memory to perform correctly. Current cloud services offer up to 16 GB of memory, although some cloud companies provide even larger amounts of memory or even allow customized memory options (see Section 2.2.3).

Geographical redundancy: As mentioned in Section 3.1.3, the implementation of the prototype MSC-S blades of an Ericsson MSC-S BC in the cloud could provide geographical redundancy, as the cluster could be split geographically, thus preventing a complete breakdown if a fault were to occur in any single site (such as fire, earthquake or power supply outage). This geographical redundancy, together with the Ericsson MSC-S blades' redundancy and fault-tolerant middleware (see Section 2.4.2), provides greater resilience against information loss, and minimizes the risk of operation downtime.

Elasticity and scalability: The elasticity provided by a cloud environment enables the possibility of rapidly creating new servers so that new blades can be added to the cluster if necessary (see Section 2.2.1.4). This capability can be leveraged by the Ericsson MSC-S BC, as blades can be added to the cluster without disturbing the rest of the system (see Section 2.4.3).

High Internet throughput: The improvement of the Internet backbone and access networks enables high throughput connections, thus making it possible to carry real-time traffic through the Internet to and from the cloud environment (see Section 2.2.1.3).

Internet network redundancy: The fully meshed connectivity provided by the Internet avoids the existence of single link points of failure between the blades and the rest of the Ericsson MSC-S BC located on-premises, and between the blades themselves. This of course require multiple physical links from the premises to multiple Internet service providers (or a single provider that provides a redundant architecture).

3.2.2. Drawbacks

Such Cloud based implementations leads to several drawbacks in comparison to the traditional deployment. These drawbacks are mainly related to the cloud-basis of the resources and the reliance on the Internet. A classification of these drawbacks is given in Table 3.2. A brief description of them follows the table.

Table 3.2. Drawbacks to implement an Ericsson MSC-S BC in the Cloud

Drawbacks	
Cloud computing	Cloud servers performance
Internet	Internet indeterminist behaviour

Cloud servers performance: The cloud servers are located in the cloud service provider's premises; therefore it is their responsibility to assure these servers' correct operation. For this reason, strict SLAs must be signed so that the servers' uptime is guaranteed by the cloud service provider(s).

Non-deterministic Internet behavior: Although we consider the enhancement of the Internet capacity as a strong motivation, its non-deterministic behavior is an important drawback for a real-time telecom application such as a MSC-S. The latency and packet loss, as well as the throughput variation are out of the user's control and, therefore, the Ericsson MSC-S BC prototype or any other telecom application will need to cope with them. The main concerns that the non-deterministic Internet behavior arise with regards to the Ericsson MSC-S BC prototype are:

- The possible incompatibility of the cluster protocols with Internet impairments: The protocol that keeps the Ericsson MSC-S blades (racked and prototype) running as a single entity might be affected by Internet delay and packet loss.
- The possible incompatibility of the MSC-S application with Internet impairments: Network impairments might cause a violation of the QoS requirements that real-time voice requires.

3.2.3. Conclusion

The additional elasticity and redundancy that this implementation could provide would enhance the reliability and scalability of telecom network applications. However, any of the drawbacks considered above (or any other) could hinder the successful functioning of the Ericsson MSC-S BC prototype.

In order to conclude whether the implementation is stable, or whether anything negatively affects the correct operation of the system, we needed to test this solution in several simulated environments. If any issues are detected, then we need to analyze if there is a troubleshooting process that could successfully resolve the situation. If any of the limitations stated in Section 1.7 prevents the cloud solution from addressing any issue, and we are unable to find a way to mitigate the problem, then this issue will be indicated as future work for the researchers who continue this investigation. The discovery of hindrances was one of the main purposes of this project.

Additionally, a complete evaluation of a selected cloud service provider needed to be done so that the tests results obtained in the lab environment could be replicated in a real environment. Finally, a test needed to be done so that the correct operation of the Ericsson MSC-S BC prototype in a real cloud environment could be practically demonstrated. However, this last step was not carried out in this thesis project due to the lack of permission to utilize the prototype MSC-S blade software outside of the Ericsson internal network, as stated in Section 1.7.

Chapter 4

Ericsson MSC-S BC lab tests

This chapter describes the different laboratory tests that were carried out in order to prove the correct operation of the Ericsson prototype MSC-S blades in a simulated Cloud environment despite the impairments introduced by the network. First, an introduction to all of the elements utilized to conduct the tests is given. Following this, the network configuration and laboratory environment setup are provided. Subsequently, the test cases and their purpose are explained in detail. Every test case is introduced with a description of the environment's topology and setup. Then, the results obtained from the test are stated. Finally, an analysis of every result is made and a conclusion is drawn. These tests are divided in two groups:

- Steady state test cases and
- Fault recovery test cases.

4.1. Test environment

The purpose of this section is to enumerate and describe the different elements that were used in order to conduct the tests. These elements were scattered over different Ericsson labs, two located in Stockholm and another in Montreal. The three labs were connected to each other through the Ericsson internal network. Additionally, the network configuration and setup are briefly described, further details are contained in Appendix A.

4.1.1. Stockholm laboratory A

The Stockholm laboratory A was the main laboratory used during the tests. This laboratory contained the actual racked Ericsson MSC-S BC implementation with all its components, several machines that run traffic generators to emulate the mobile network elements necessary for the tests, and two additional machines were available in order to implement the different topology designs for the tests.

4.1.1.1 Ericsson MSC-S BC

The Ericsson MSC-S BC located in the Stockholm laboratory A includes all the elements from which the system is composed. These elements were enumerated and described in Section 2.4. This MSC-S BC is the traditional racked implementation; therefore, it contains several racked blades that could be used in the tests.

4.1.1.2 HLR Traffic Generator

The HLR mobile network element was implemented through an Ericsson proprietary traffic generator application that simulated its behavior. The purpose of this component was to maintain the subscriber information necessary to support call traffic as well as performing the HLR functions in the test network.

The HLR traffic generator application was run on a dedicated Intel Xeon 3050 machine with the following features:

Table 4.1. HLR traffic generator machine's features.

Processor frequency	2.13 GHz
Number of processors	2, with 2 cores each
RAM memory	2 GB
Operating System	OpenSUSE 11.3, 64-bits version

4.1.1.3 RAN Traffic Generator

Both RNCs and BSCs were implemented through a single Ericsson proprietary traffic generator application. These elements generated the call traffic used to simulate a real mobile network, although the only type of traffic that was utilized were location updates and voice calls, either GSM or UMTS based.

This traffic generator application was run on a dedicated Intel Xeon 3065 machine with the following features:

Table 4.2. RAN traffic generator machine's features.

Processor frequency	2.33 GHz
Number of processors	2, with 2 cores each
RAM memory	2 GB
Operating System	OpenSUSE 11.2, 64-bits version

4.1.1.4 Additional machines

In order to realize the topologies described in Section 3.1 additional machines were necessary. Two more machines were provided by Ericsson. The names used for these machines in the rest of this thesis are: *Bridge* and *Cloud* machines.

The *Bridge* machine is a Genuine Intel computer with the following features:

Table 4.3. *Bridge* machine's features.

Processor frequency	2.83 GHz
Number of processors	4, with 4 cores each
RAM memory	12 GB
Operating System	Ubuntu 10.04.3 LTS, 64-bits version

The *Cloud* machine is a Genuine Intel computer with the following features:

Table 4.4. *Cloud* machine's features.

Processor frequency	2.53 GHz
Number of processors	16, with 4 cores each
RAM memory	32 GB
Operating System	Ubuntu 10.04.3 LTS, 64-bits version

4.1.2. Stockholm laboratory B

The Stockholm laboratory B contained two physical machines that were used to implement external blades in the tests. Both were Intel Xeon machines with the following features:

Table 4.5. Stockholm laboratory B machines' features.

Processor frequency	2.4 GHz
Number of processors	24, with 6 cores each
RAM memory	60 GB
Operating System	OpenSUSE 11.4, 64-bits version

4.1.3. Canada laboratory

The Canada laboratory consisted of four virtual machines set up using VMware [67]. All of these were Intel Xeon X5650 virtualized machines with the following characteristics:

Table 4.6. *Canada* machine's features.

Processor frequency	2.67 GHz
Number of processors	2, with 2 cores each
RAM memory	20 GB
Operating System	Ubuntu 10.04.3 LTS, 64-bits version

4.1.4. Network configuration

This section provides a detailed view of the test environment utilized in the tests. First, the Stockholm laboratory B setup is explained. Second, the Canada laboratory setup and bench-marking process are described. Third, the solutions considered to interconnect the three laboratories used in the tests are described, and the reasons for the final selection are exposed. Then, an explanation of the Ericsson MSC-S BC's constraints is provided. Finally, the tools utilized throughout the tests are presented.

4.1.4.1 Stockholm laboratory B setup

A virtualization layer was added to the test machines located in the Stockholm laboratory B. This was done in order to simulate more closely the virtualization utilized in typical cloud implementations. Additionally, this virtualization allowed the creation of two virtual machines running on each physical machine, thus using the existing computing resources more efficiently.

Given the resources (as described in Section 4.1.2), the virtual machines that were created had the following characteristics: **24** GB of RAM memory, and **4** processors.

Kernel-based Virtual Machine (KVM) [68] was the virtualization software used to create the virtual machines. In addition, the QEMU [69] program was used as a CPU emulator on top of KVM. A more detailed explanation of how these virtual machines were setup can be found in Appendix A.2.1.

Once the virtual machines were created, the Stockholm laboratory B network topology was modified so that all of the four new machines were able to communicate locally with each other, and through VPN with the *Bridge* machine in Stockholm laboratory A, and, by extention, with the whole test network. Figure 4.1 illustrates the Stockholm laboratory B network topology. As can be observed in this figure, the blade numbers chosen for the machines were 13, 14, 15, and 16, although they could be modified as needed. In addition, details of this network configuration are described in Appendix A.2.2.

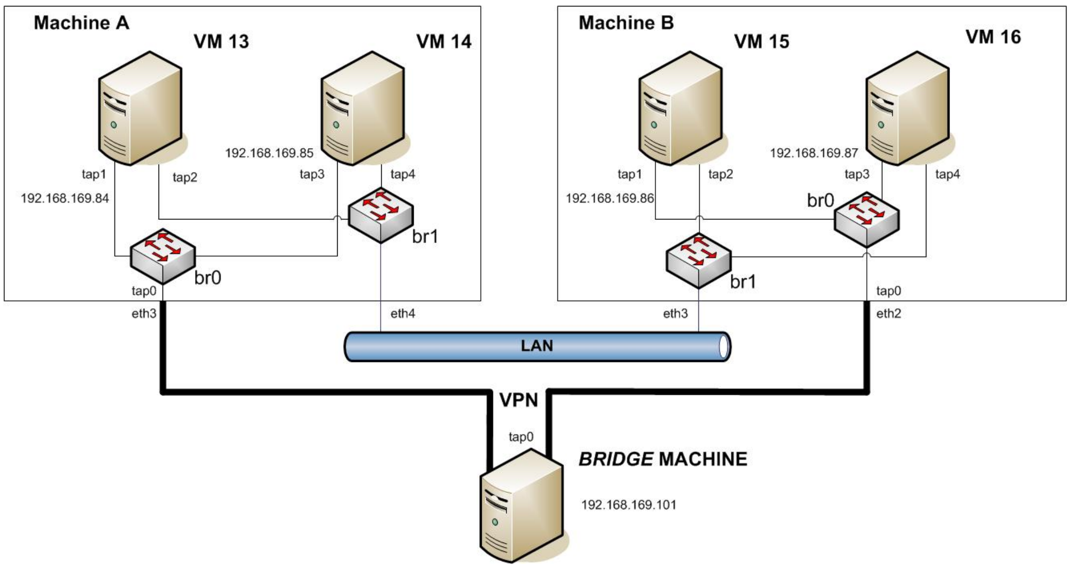


Figure 4.1. Stockholm laboratory B network topology.

4.1.4.2 Canada laboratory setup and bench-marking

The network configuration adopted in the Canada laboratory virtual machines is illustrated in Figure 4.2. This configuration enabled a local communication between the virtual machines. Additionally, The virtual machines connected through VPN with the *Bridge* machine in Stockholm laboratory A, and, by extention, with the whole test network.

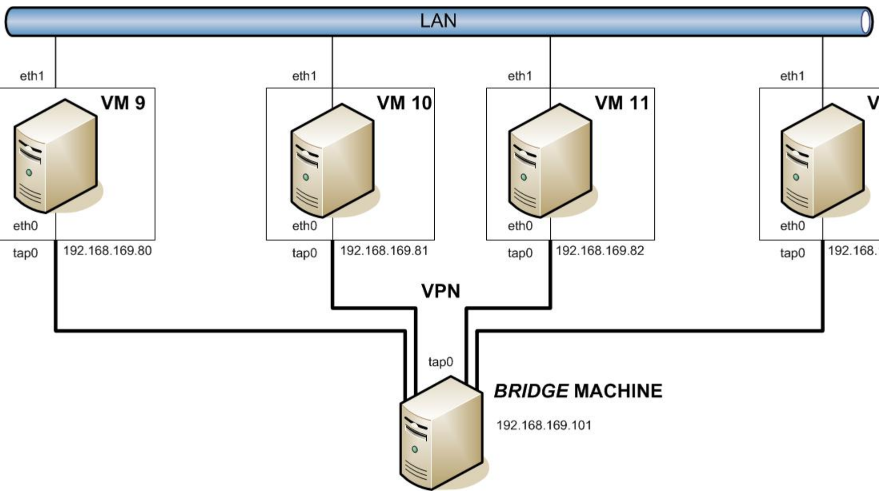


Figure 4.2. Canada laboratory network setup.

Additionally, network bench-marking tests were conducted in order to know what network impairments were present between the Canada laboratory and the Stockholm laboratories. This bench-marking process consisted of a UDP test (see Section 5.5.1.1) and a PING test (see Section 5.5.1.3) that measured the round-trip time[1] (RTT), and the packet loss rate. A sample was taken every 5 minutes, where every sample was the result of a 10 seconds test using either *ping flooding* (in the PING test) or *iperf* (in the UDP test).

The average RTT obtained in the 24-hour test was **118.5 ms**. Figure 4.3 represents the *ping flooding* average RTT obtained for every sample. In addition, the maximum RTT value obtained in the test was **142.4 ms**. Figure 4.4 represents the *ping flooding* maximum RTT obtained for every sample.

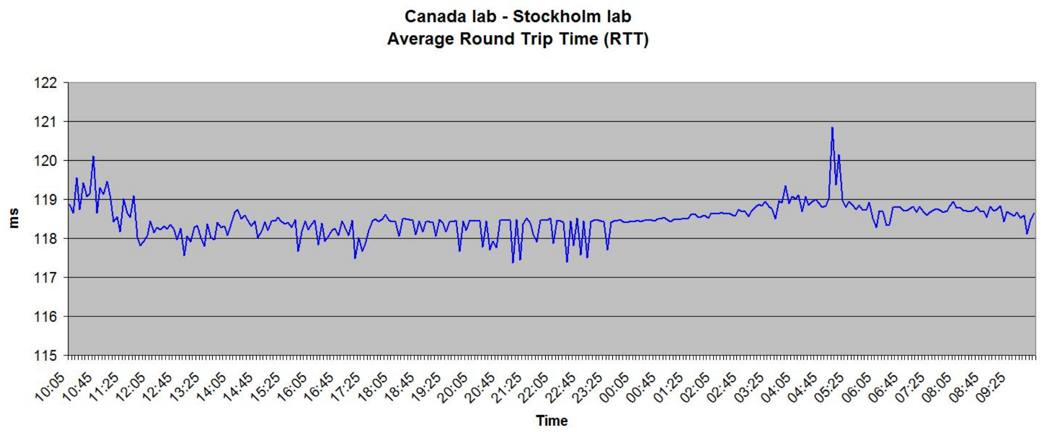


Figure 4.3. Average RTT between the Canada laboratory and the Stockholm laboratories.

[1]The round-trip time is defined as the time it takes for a packet to be delivered from its source to its destination, plus the time it takes to acknowledge the packet back from the destination to the sender.

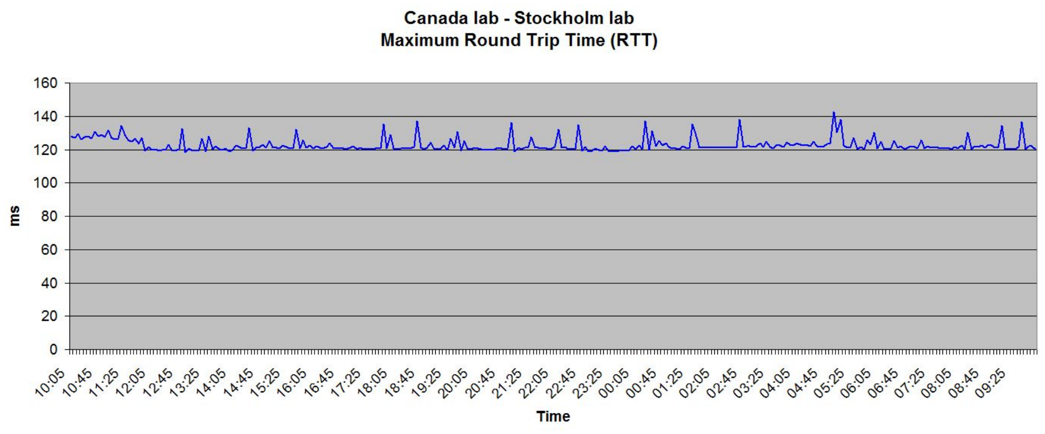


Figure 4.4. Maximum RTT between the Canada laboratory and the Stockholm laboratories.

The average packet loss rate obtained in the 24-hour test was **0.022%**. Figure 4.3 represents the *iperf* average packet loss rate obtained for every sample.

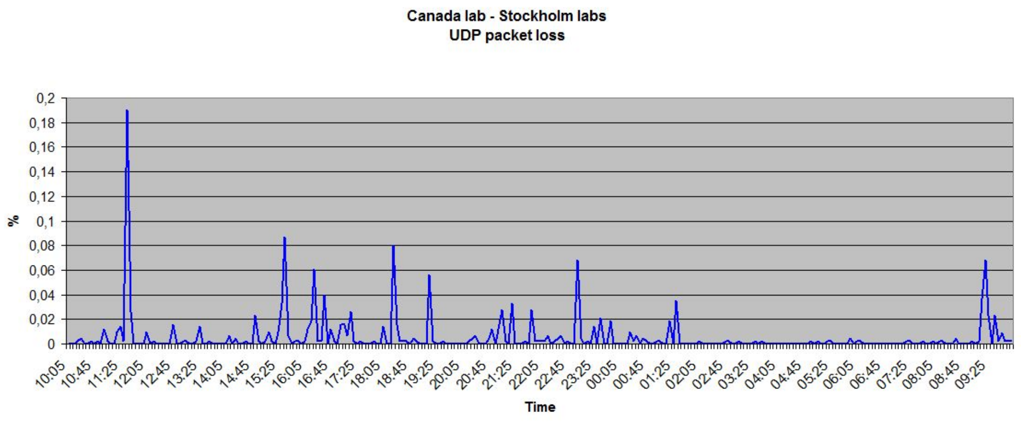


Figure 4.5. Average packet loss rate between the Canada laboratory and the Stockholm laboratories.

4.1.4.3 Tunneling solution

In order to provide full connectivity between all the elements in the network, and considering that the traffic carrying private IP addresses is filtered in many parts of the Ericsson intranet as well as in the Internet, a solution needed to be found to allow this traditional configuration to work. Two solutions were considered before one of them was selected as the definitive choice: EoIP tunneling and Level 2 Virtual Private Network (VPN) tunneling.

Basic EoIP tunneling: This solution entailed the implementation of an unencrypted EoIP tunnel, where the Ethernet packets (including its VLAN tag) were sent through the intranet to the other end of the tunnel by means of IP packets using public IP addresses. Figure 4.6 depicts how the Ethernet frames were encapsulated in IP packets.

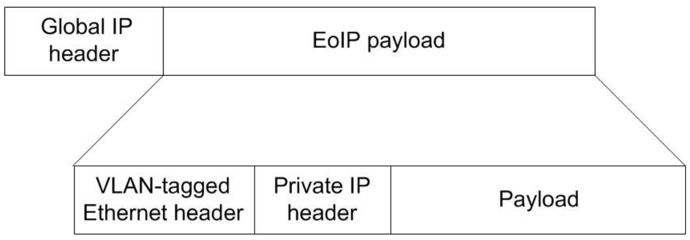


Figure 4.6. Ethernet over IP encapsulation example.

The practical implementation of the EoIP tunnel was tested using an existing Linux driver based on the RFC 3378 [40]. The open-source code of this implementation was obtained from [70], and the tutorial on how to install and use the driver was followed.

Level 2 Virtual Private Network (VPN) tunneling: The VPN tunneling solution consists of a VPN server to which the VPN clients connect to setup a tunnel between them and the server. This tunnel can be used to send IEEE 802.3 frames to the other end of the pipe. In addition, this VPN offers several different types of encryption, thus making the tunneled communication more secure. Figure 4.7 illustrates the structure of a VPN packet. Another important VPN feature is that the VPN server can be configured so that every Ethernet frame the server receives from a client is replicated and sent to the other clients, thus simulating a real LAN. This functionality is called client-to-client. In this LAN the VPN server acts as bridge to enable communication between the clients. This concept is illustrated in Figure 4.8.

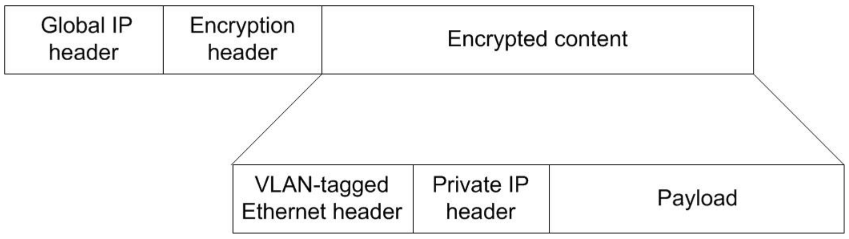


Figure 4.7. VPN encapsulation example.

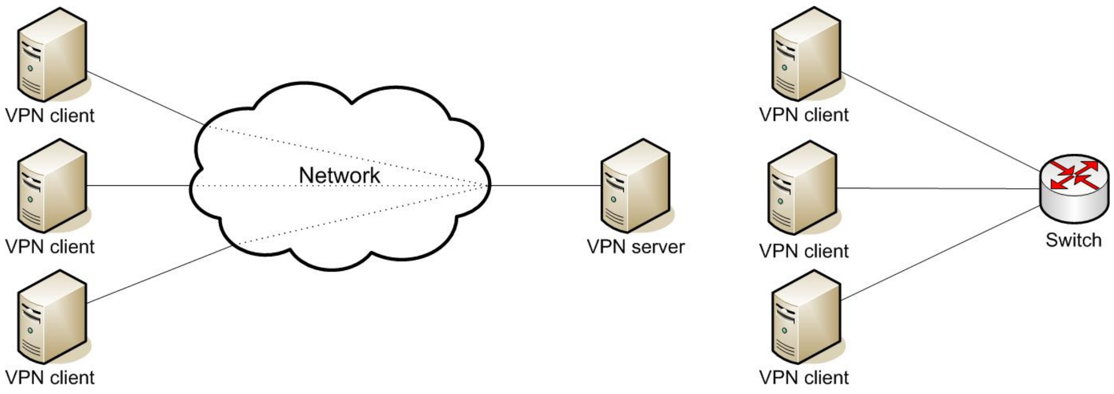


Figure 4.8. Client to client bridging concept with the tunnels shown on the left and the logical view of the network shown on the right.

The VPN was an open source implementation, OpenVPN 2.1.0 [71]. Additionally, two tutorials, [72] and [73], were followed to configure the VPN server, the clients, and the bridge necessary for the server to work as an Ethernet bridge. The complete process of the VPN configuration is detailed in Appendix A.1.

Considering the additional functionalities that the VPN offers, as well as the ease of installation and configuration, OpenVPN was the final choice to implement the Ethernet tunnel necessary for the tests. The *Bridge* machine was used as the VPN server to which the external blades could connect as OpenVPN clients.

4.1.4.4 Ericsson MSC-S BC network constraints

The tests conducted in this thesis work entailed the setup of external prototype MSC-S blades to be run as if they were internal to the racked Ericsson MSC-S BC's implementation. Therefore, the traditional network configuration was utilized. This configuration requires us to define private IP addresses for the 8 virtual interfaces which a blade may use. However, in the tests carried out in this thesis project only 6 of the 8 interfaces were utilized, as the functionalities provided by the other 2 interfaces were not necessary for our purposes. Each of these 6 interfaces is connected to a different VLAN, thus each blade can communicate through the VLAN with the rest of the system elements that are connected to this same VLAN. In addition, each VLAN is used to send a specific sort of traffic, thus separating the different functions of the system.

The 6 virtual interfaces traditional IP configuration uses the following rules:

Table 4.7. Ericsson MSC-S BC network contraints.

Interface connected to VLAN 169	IP address 192.168.169.**X**
Interface connected to VLAN 170	IP address 192.168.170.**X**
Interface connected to VLAN 171	IP address 192.168.171.**X**
Interface connected to VLAN 172	IP address 192.168.172.**X**
Interface connected to VLAN 173	IP address 192.168.173.**X**
Interface connected to VLAN 1001	IP address 192.168.1.**Y**

Where **X** is equal to "71 + N" and **Y** is equal to "103 + N", where N is the number assigned to the blade in the system. Additionally, the functionalities provided by the traffic of each of these VLANs are:

- VLANs 169 and 170 are used for communication between the blades and the Adjunct Processor Group[2] version 43 (APG43) [66] for operation and management functions.

- VLANs 171, 172 and 173 are used for communication between blades to maintain the state of the cluster.

- VLAN 1001 is used to communicate between the blade cluster and the SPX. Therefore, all the signaling traffic handled by the cluster MSC-S blades goes through this VLAN.

The virtual interfaces could be configured using as many physical interfaces as required. However, in the tests conducted in this thesis project either one or two physical interfaces were utilized in each blade to implement all of the virtual interfaces for this blade.

[2]The Adjunct Processor Group is the main component of the Ericsson MSC-S BC I/O system, and is in charge of the cluster statistics and the *man-machine language* (mml) commands.

4.1.5. MSC-S blades states

This section clarifies the three different states in which the MSC-S blades can be during the development of this master thesis. The MSC blades state indicates the status of a blade within the MSC-S BC. It is a part of the consistent cluster view. These possible three states are: *active*, *passive*, and *recovery*.

Active: An "active" MSC-S blade is a cluster member and traffic load will be distributed to this blade.

Passive: A "passive" MSC-S blade is a cluster member, but the traffic is not distributed to this blade.

Recovery: A MSC-S blade in "recovery" state is a blade with transient fault conditions, prevented from entering the "active" state.

4.1.6. Testing tools

In order to conduct the tests with network impairments characteristic of a cloud environment, several tools were necessary. These tools are briefly described below along with a reference to further details.

Netem: Netem was used to introduce network impairments which the Ericsson MSC-S BC was to be tested against. In this master's thesis project only delays and packet losses were simulated. A more detailed description of how to use this tool is provided in Appendix A.10.

Traffic generators: Two traffic generators were used in this project to simulate real mobile network signalling traffic cases. One emulated an HLR while the other emulated a Radio Access Network (RAN). Further details for these traffic generators is provided in Appendix A.9.

4.2. Steady state

This section focuses on the steady state tests. Therefore, these tests include cases were no deliberate blade faults of any sort were introduced in the system. One of the purposes of these tests was to evaluate the operation of the Ericsson MSC-S BC prototype in several different topologies: the hybrid cluster (see Figure 3.1), the external cluster (see Figure 3.2), and the geographically split cluster (see Figure 3.3). The external cluster, and the geographically split cluster topologies were tested in presence of network impairments. In addition, cluster scalability tests were conducted in presence of network impairments. These tests were performed following a bottom-up approach in terms of topology complexity.

4.2.1. Initial hybrid cluster test

The first tests conducted consisted of testing the Ericsson MSC-S BC configured with the hybrid cluster topology shown in Figure 3.1. The external prototype MSC-S blade was set up in two different ways. First, using a physical computer as the hardware platform. Second, using a virtual machine as the virtual blade platform. This step by step approach was meant to simplify the troubleshooting if any complication(s) arose.

4.2.1.1 Laboratory setup

In this setup, three of the racked blades located in Stockholm laboratory A form a cluster together with an external prototype MSC-S blade configured as described in Appendix A.3 using just one physical interface. The racked blades used in this test had 1,2, and 3 as their assigned blade numbers while the external prototype blade had a different blade number assigned in each of the two following tests. The tested topology is detailed in Figure 4.9. In addition, the cluster activation was done as described in Appendix A.4.

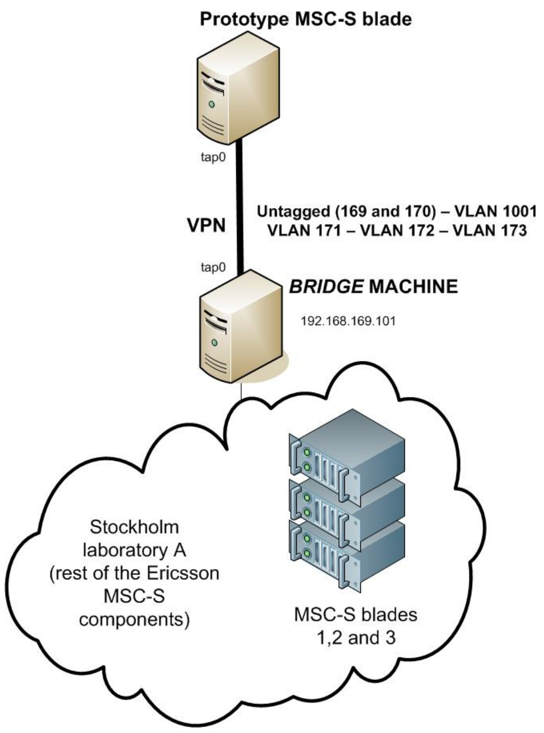

Prototype MSC-S blade
tap0
VPN
Untagged (169 and 170) – VLAN 1001
VLAN 171 – VLAN 172 – VLAN 173
tap0
BRIDGE MACHINE
192.168.169.101
Stockholm laboratory A (rest of the Ericsson MSC-S components)
MSC-S blades 1,2 and 3

Figure 4.9. Network topology when testing the hybrid Ericsson MSC-S BC.

4.2.1.2 Physical machine as external blade

The purpose of this test was to demonstrate that the prototype MSC-S blade of the hybrid Ericsson MSC-S BC could be implemented outside the rack and with the IP configuration designed by the authors of this thesis. The test consisted of setting up the cluster using this external prototype MSC-S blade together with the three racked MSC-S blades located in Stockholm laboratory A. The external prototype MSC-S blade had 17 as its assigned blade number. Afterwards, the system was tested with a high rate of GSM and UMTS calls (200 calls per second of each type) using the traffic generator introduced in Section 4.1.6.

The result of this test was satisfactory both in terms of the cluster activation and the traffic case. All the MSC-S blades remained in "active" state throughout all the testing, thus demonstrating the cluster protocols correct operation in a cluster with an external prototype MSC-S blade. Additionally, the traffic case ran as expected, with no dropped calls nor any sort of failure regarding cluster stability. Therefore, the next step was to configure the external prototype blade as a virtual machine.

4.2.1.2 Virtual machine as external blade

In this test, the external prototype MSC-S blade was implemented using a virtual machine. Blade 15, already configured in Stockholm laboratory B (Figure 4.1), was chosen as the external prototype MSC-S blade for the test. This test aimed to prove that an additional layer of virtualization did not negatively affect the correct operation of a prototype MSC-S blade running on top of the VM. Once that was achieved, traffic (GSM and UMTS calls) was emulated in the network in order to verify the stability of the system.

The cluster setup was satisfactory, and all the MSC-S blades were in "active" state after the cluster activation. In addition, the operation of the cluster while serving traffic was correct, with no dropped calls nor any incidence in the cluster.

4.2.1.3 Discussion

Given the satisfactory results in the tests described above, it was proved that a prototype MSC-S blade with our setup conditions could be externalized without causing any problem to either the cluster stability (cluster protocols) or to the traffic served by the MSC-S application. Furthermore, this correct operation was also demonstrated when the external prototype MSC-S blade was running on a virtual machine instead of running directly over the host machine OS. Therefore, the foundations were established that a whole Ericsson MSC-S BC prototype could potentially be built outside the rack using virtual machines to implement the prototype MSC-S blades.

4.2.2. External cluster test

Once it was demonstrated that a prototype MSC-S blade could be implemented out of the rack using a virtual machine as its platform, the next step consisted of building the Ericsson MSC-S BC composed only of prototype MSC-S blades outside the racked architecture.

In the beginning, this topology was deployed using the four virtual machines that were set up in Stockholm laboratory B (Figure 4.1). This external cluster was tested with traffic in different situations. In the beginning, we needed to prove that the cluster protocols were stable in this network configuration, with no other setback than the fact that the cluster was completely externalized. From there on, network impairments were added to the cluster so that its performance could be evaluated in a simulated Cloud environment from a network perspective. Different combinations of delay and packet loss were introduced between the MSC-S prototype blades and the rest of the system (located in Stockholm laboratory A, as explained in 4.1.1), while traffic was run in the network. The traffic case chosen consisted of GSM and UMTS calls.

Then, considering the data obtained in the Stockholm laboratory B test, a conclusion was drawn with regards to whether an external Ericsson MSC-S BC could be set up with its prototype MSC-S blades located in the Canada laboratory. The non-simulated network impairments between the Canada laboratory and the Stockholm laboratory A were also considered in this evaluation. If the results were favourable, a traffic test would be conducted with the external cluster located in the Canada laboratory.

Several issues arose both in the prototype MSC-S external blades and in the RAN traffic generator with regards to the network impairments present in the tests. The description of these challenges and their solutions, when possible, are reflected in this section.

4.2.2.1 Stockholm laboratory tests

First, the virtual machines available in Stockholm laboratory B were used to test the Ericsson MSC-S external cluster topology and demonstrate its correct operation. This section includes the laboratory setup, the tests, and the results obtained.

Laboratory setup

Every prototype MSC-S blade was configured with two interfaces following the Appendix A.3 manual (see Figure 4.10). The cluster activation was done as described in Appendix A.4. The network impairments were introduced in the *tap0* and *eth1 Bridge* machine's interfaces using the Netem tool (see Section 4.1.6), thus making the network impairments bidirectional.

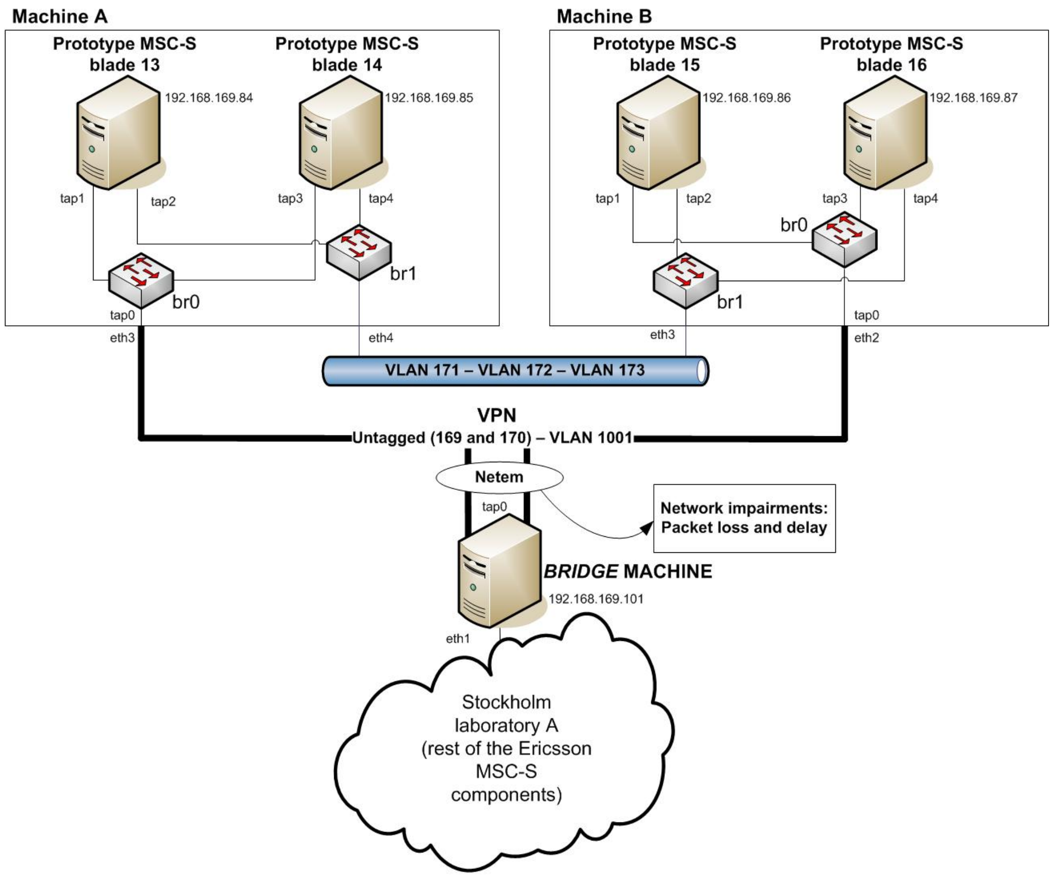


Figure 4.10. Stockholm laboratory B network topology when testing the external Ericsson MSC-S BC.

Tests

During our external cluster tests, we encountered a partial issue in the cluster activation which was not related to the network configuration nor to the impairments introduced in the network (packet loss and delay). This prototyping issue is described in Section 4.2.5. However, in the majority of cases the cluster activation worked correctly.

Once the cluster was activated and all its MSC-S blades were in "active" state, traffic was run in the system. First, the cluster was tested without network penalties

(Ericsson Stockholm internal network impairments were negligible). Secondly, delays were introduced between MSC-S blades and the rest of the system. Finally, several combinations of packet losses and delays were tested. For all these combinations, the cluster was left serving traffic until a large amount of calls was correctly completed, so that failures could be detected. When inconsistencies appeared, the test was left a longer time in order to verify the unstable operation of the system. The purpose of this test was to find out an approximation for the maximum delay, packet loss, and combination of both that could be borne between the prototype MSC-S blades and the rest of the system. The criteria by which the delay and packet loss were increased was very conservative in the beginning, given that there was no knowledge of how the system would react.

The system was intended to be tested with a high traffic rate in order to seek out worst case scenario limitations in the system. However, as it will be explained in Section 4.2.5, an issue regarding the RAN traffic generator behavior was encountered, which compelled us to set the traffic rate to a low value: 50 calls per second for both types of traffic (GSM and UMTS), thus 100 calls per second in total.

The results of all the external Ericsson MSC-S BC test samples with additional delay, packet loss, and combination of both are compiled in Appendix B.1. A brief summary of those results, dividing the three different test cases (only delay, only packet loss, and a combination of delay and packet loss), is offered below:

Delay: In the case where only delay was added between the external prototype MSC-S blades and the rest of the system, a round-trip time (RTT) of **167 ms** was detected as the limit, as it can be seen in Table B.1. When this RTT was surpassed, a great amount of calls were dropped due to an overflow in the SCTP buffers, thus displaying an unacceptable behavior. This buffer overflow was caused by the large amount of packet retransmissions originated by the exceeding number of SCTP timeouts due to the high delay. The SCTP retransmission timeout configuration values were predefined by our employers to the following values:

```
sysctl net.sctp.rto_initial=200
sysctl net.sctp.rto_min=100
sysctl net.sctp.rto_max=400
```

Packet loss: With packet loss as the only network impairment, a maximum packet loss rate of **12%** was obtained as limit, as shown in Table B.2. Whenever this rate was overcome, the same SCTP buffer overflow observed in the delay case occurred. The high packet loss rate originated an untenable number of SCTP retransmissions that overloaded the buffer.

Packet loss and delay: Several combinations were tested where both packet loss and delay influenced the communication between the external cluster and the rest of the Ericsson MSC-S BC system. The approach we followed consisted of testing several high delay values while varying the packet loss rate, aiming to obtain the highest packet loss rate supported with the actual delay. As shown in Table B.3, it can be observed that the limits of both packet loss and delay are reduced when both of them came into play at the same time. The different limit combinations that were obtained are:

- A RTT of **100 ms** with a packet loss rate of **2%**.
- A RTT of **120 ms** with a packet loss rate of **1%**.
- A RTT of **140 ms** with a packet loss rate of **0.5%**.
- A RTT of **160 ms** with a packet loss rate of **0.1%**.

Canada laboratory analysis

One of the purposes of this test was to evaluate whether the external cluster topology could be implemented in the available Canada laboratory virtual machines. With the results presented above, and having obtained an average **118.5 ms** RTT in a 24-hour test between Canada laboratory and the Stockholm laboratories (see Section 4.1.4), the limit that would apply would be a packet loss rate of **1%**. The average packet loss measured in a 24-hour test was **0.022%** (see Section 4.1.4), which is far enough from our limit. Therefore, an external cluster in the Canada laboratory was theoretically possible.

4.2.2.2 Canada laboratory test

As demonstrated above, an external cluster in the Canada laboratory was theoretically possible. Therefore, the virtual machines available in the Canada laboratory were used to test the external cluster and demonstrate its correct operation. This section includes the laboratory setup, the tests, and the results obtained.

Laboratory setup

Every prototype MSC-S blade was configured with two interfaces following the Appendix A.3 manual (see Figure 4.11). The cluster activation was done as described in Appendix A.4. No network impairments were introduced in this test, as delay and packet loss rate were already present in the network.

Test

The activation process had the same issue as in the Stockholm laboratory B test (see Section 4.2.5). Then, traffic was run in the network (50 GSM calls per second, and 50 UMTS calls per second). The performance of the cluster while serving traffic was correct, and no calls were dropped nor any incidence regarding cluster stability ocurred. Therefore, this test verified that an Ericsson MSC-S BC can be deployed while having the prototype MSC-S blades in a remote location.

4.2.2.3 Discussion

The results obtained with the external cluster topology are good, as the Ericsson MSC-S BC performance is correct even with large delays and packet loss rates between the external prototype MSC-S blades and the rest of the system. The encountered limits are high enough to enable a external cluster implementation in a determined location far from the rest of the Ericsson MSC-S BC elements. Besides, SCTP parameters could be tuned to increase this limit, although this would entail longer times of call setup and tear down, thus reducing the QoS in the calls.

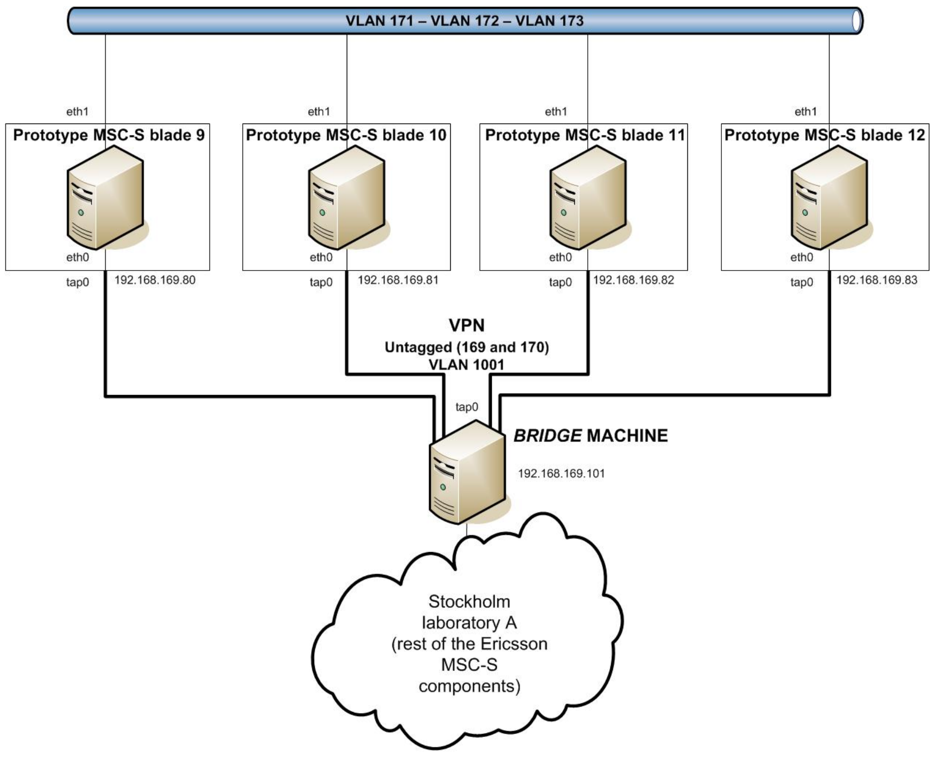


Figure 4.11. Canada laboratory network topology when testing the external Ericsson MSC-S BC.

It is important to note that the limitations in all the cases (delay, packet loss, and a combination of both) are directly attached to the SCTP connections that carry the signalling traffic between the SPX and the MSC-S blades, as the cluster protocols are stable even when the traffic is not. Therefore, this demonstrates that the cluster protocols behavior is good in presence of this sort of network impairments between elements in the system.

In addition, the external cluster topology was tested deploying its external prototype MSC-S blades in the Canada laboratory, thus verifying that the system performance is indeed good when locating the prototype MSC-S blades in a remote location. However, a new traffic generator would need to be developed in order to demonstrate a good performance with a higher traffic rate.

4.2.3. Split cluster test

In order to solve the single point of failure issue (see Section 1.2) for the Ericsson MSC-S BC, we needed to demonstrate the correct operation of a geographically split cluster. Following the bottom-up approach adopted in this thesis project, a local laboratory test (in the Stockholm laboratory B) was conducted prior to a real test using the Canada laboratory virtual machines.

The local laboratory test was carried out introducing different combinations of delays and packet loss rates between a prototype MSC-S blade and the rest system (including the other prototype MSC-S blades) while traffic was run in the network. The traffic case selected consisted of GSM and UMTS calls.

Considering the results of the Stockholm laboratory B test, a conclusion was drawn in regards to whether the geographically split topology could be deployed running the prototype MSC-S blades in the Canada lab virtual machines, taking into account the non-simulated network impairments between the Canada laboratory and the Stockholm laboratory A. If the results were favorable, several tests would be conducted varying the geographically split configuration to verify a uniform behavior of this topology.

No new issues were encountered throughout the geographically split cluster tests apart from those mentioned in Sections 4.2.1 and 4.2.2. However, it is important to note that all the modifications performed in order to solve previous challenges were present in the tested system.

4.2.3.1 Stockholm laboratory tests

First, the virtual machines available in Stockholm laboratory B were used to implement the prototype MSC-S blades in the geographically split cluster in order to demonstrate its correct operation. This section includes the laboratory setup, the tests, and the results obtained.

Laboratory setup

Every prototype MSC-S blade was configured with two interfaces following the Appendix A.3 manual (see Figure 4.12). The cluster activation was done as described in Appendix A.4. The network impairments were introduced in the *tap3* and *tap4* interfaces of the host machine B, and in the *eth0* and *eth1* interfaces of the prototype MSC-S blade 16 using the Netem tool (see Appendix A.10), thus making the network impairments bidirectional.

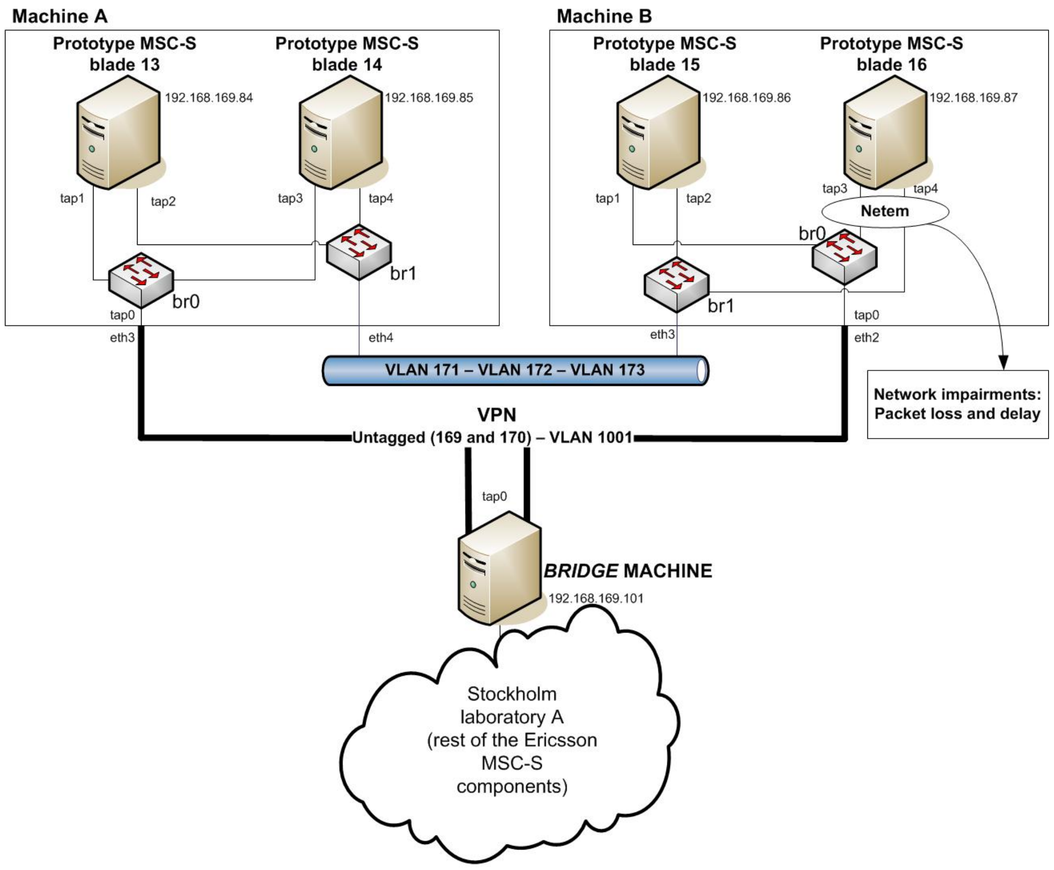


Figure 4.12. Stockholm laboratory B network topology when testing the geographically split Ericsson MSC-S BC.

Tests

During the geographically split cluster tests, we still encountered the activation issue described in Section 4.2.5. However, there were no incidents in the cluster activation attached to the network configuration nor to the network impairments the system was tested with.

Several combinations of packet loss rates and delays were tested in this step, while the geographically split Ericsson MSC-S BC was serving traffic. For all these combinations, the cluster was left serving traffic until a large amount of calls was correctly completed, so that any incident could be detected. When inconsistencies appeared, the test was left a longer time in order to verify the undesirable operation of the system. The purpose of this test was to obtain an approximation for the maximum delay, packet loss, and combination of both that could be borne between a prototype MSC-S blade and the rest of the Ericsson MSC-S BC system (including the rest of the prototype MSC-S blades forming the cluster). As the behavior was expected to be similar to Section 4.2.2 tests, the criteria by which the delay and packet loss were increased was more effective.

As the RAN traffic generator issue (see Section 4.2.5) remained, the traffic rate was set to a low value: 50 calls per second for both types of traffic (GSM and UMTS), thus 100 calls per second in total. The results of all the geographically split Ericsson MSC-S BC test samples with additional delay, packet loss, and combination of both are compiled in Appendix B.2. A brief summary of those results, dividing the three different test cases (only delay, only packet loss, and a combination of delay and packet loss), is offered below:

Delay: In the case where only delay was added, a RTT of **167 ms** was detected as the limit, as it can be seen in Table B.4. When this RTT was surpassed, an unacceptable amount of calls were dropped due to an overflow in the SCTP buffers of the prototype MSC-S blade with added delay. This buffer overflow was caused by the large amount of packet retransmissions originated by the exceeding number of SCTP timeouts due to the high delay in the affected blade.

Packet loss: With packet loss as the only network impairment, a maximum packet loss rate of **11%** was obtained as limit, as shown in Table B.5. Whenever this rate was overcome, the same SCTP buffer overflow observed in the delay case occurred in the prototype MSC-S blade with added packet loss. The high packet loss rate originated an untenable number of SCTP retransmissions that overloaded the buffer.

Packet loss and delay: Several combinations were tested where both packet loss and delay influenced the communication between a prototype MSC-S blade and the rest of the Ericsson MSC-S BC (including the other prototype MSC-

S blades). The approach we followed consisted of testing several high delay values while varying the packet loss rate, aiming to obtain the highest packet loss rate supported with the actual delay. As shown in Table B.1, it can be observed that the limits of both packet loss and delay are reduced when both of them came into play at the same time. The different limit combinations that were obtained are:

- An RTT of **100 ms** with a packet loss rate of **2%**.
- An RTT of **120 ms** with a packet loss rate of **1%**.
- An RTT of **140 ms** with a packet loss rate of **0.5%**.
- An RTT of **160 ms** with a packet loss rate of **0.1%**.

Canada laboratory analysis

One of the purposes of this test was to evaluate whether the geographically split cluster topology could be implemented using the Canada laboratory virtual machines as some of the prototype MSC-S blades. With the results presented above, and having obtained an average **118.5 ms** RTT in a 24-hour test between Canada laboratory and the Stockholm laboratories (see Section 4.1.4), the limit that would apply would be a packet loss rate of **1%**. The average packet loss measured in a 24-hour test was **0.022%** (see Section 4.1.4), which is far enough from our limit. Therefore, a geographically split cluster with the prototype MSC-S blades located in the Canada laboratory virtual machines was theoretically possible.

4.2.3.2 Canada laboratory tests

As demonstrated above, a geographically split cluster with the presence of prototype MSC-S blades in the Canada laboratory was theoretically possible. Therefore, the virtual machines available in the Canada laboratory were used to test the geographically split cluster together with the virtual machines in the Stockholm laboratory B, deploying the prototype MSC-S blades in both locations. This section includes the laboratory setup, the tests, and the results obtained.

Laboratory setup

Every prototype MSC-S blade was configured with two interfaces following the Appendix A.3 manual. In addition, a virtual machine in every laboratory (Canada and Stockholm B) was used to bridge the local traffic (VLANs 171 to 173) to the VPN connection, thus providing cluster connecticity despite the geographical division of the cluster in two parts (see Figure 4.13 and Figure 4.14). The cluster activation was done as described in Appendix A.4. No network impairments were introduced in this test, as delay and packet loss were already present in the network.

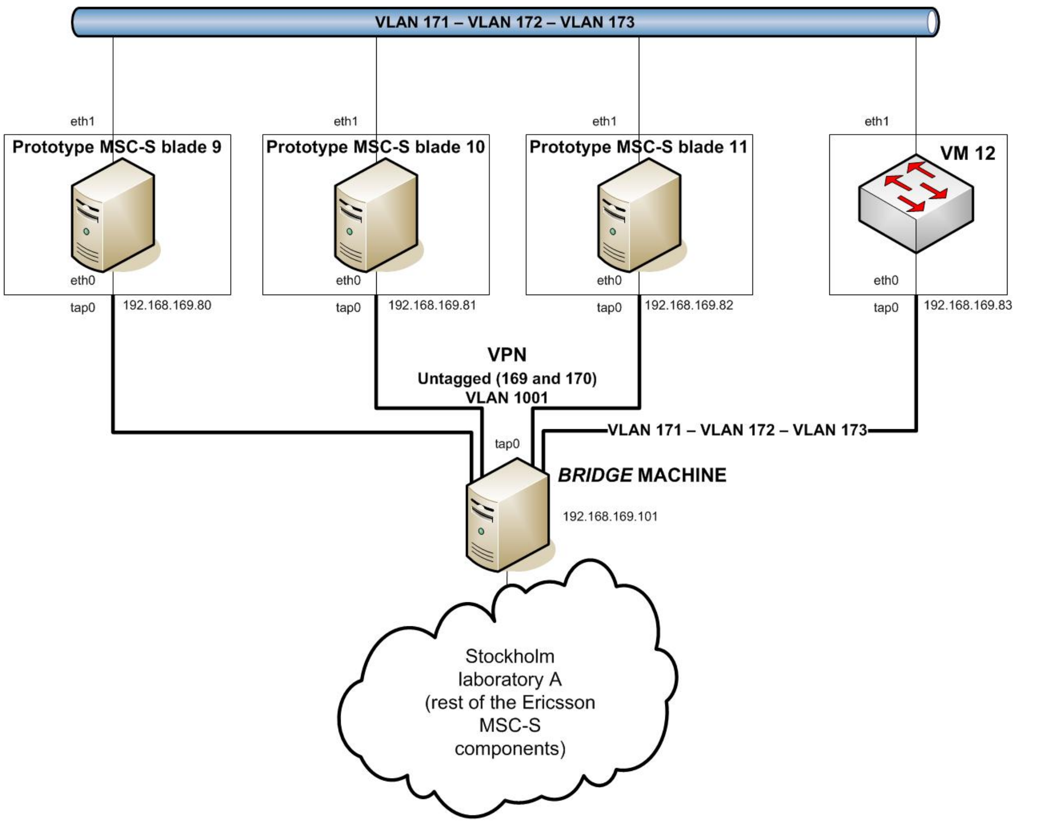


Figure 4.13. Canada laboratory network topology when testing the geographically split Ericsson MSC-S BC.

Tests

Several different variations of the geographically split cluster were tested when using the Canada laboratory virtual machines. For all of them, the activation process had the same prototyping issue explained in Section 4.2.5. In addition, the traffic run in the network was composed of 50 GSM calls per second, and 50 UMTS calls per second. The tested topologies were the following:

- Three MSC-S blades in Stockholm laboratory B, and one in Canada laboratory.
- Two MSC-S blades in Stockholm laboratory B, and two in Canada laboratory.

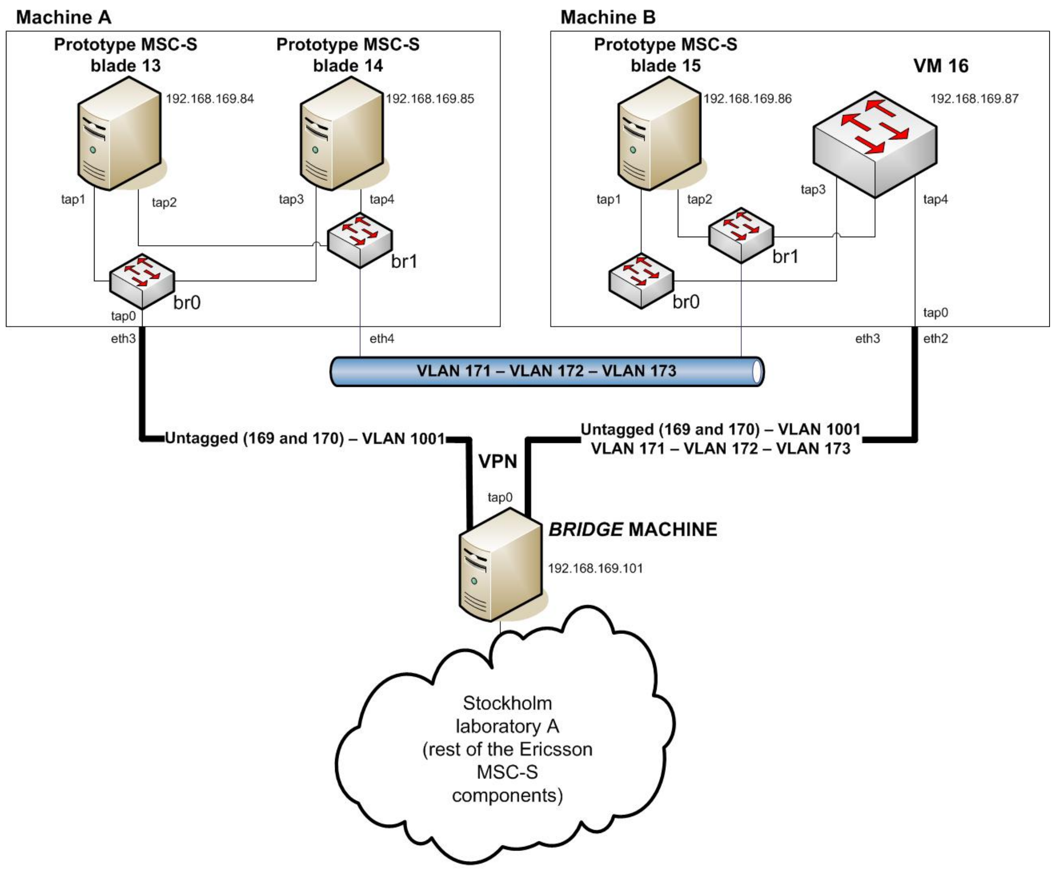


Figure 4.14. Stockholm laboratory B network topology when testing the geographically split Ericsson MSC-S BC with the Canada laboratory.

- One MSC-S blade in Stockholm laboratory B, and three in Canada laboratory.

In every case the performance of the cluster while serving traffic was correct, and no calls were dropped nor any incidence regarding cluster stability ocurred. Therefore, this test verified that an Ericsson MSC-S BC can be deployed with the prototype MSC-S blades placed in different, remote locations.

4.2.3.3 Discussion

The results obtained with the geographical split cluster are good, as the Ericsson MSC-S BC performance is correct even with large delays and packet loss rates. The encountered limits are all related to the SCTP connections that carry the signalling traffic between the SPX and the MSC-S blades. Therefore, this demonstrates that the cluster protocol's behavior is good in presence of this sort of network impairments even when the cluster is located in different remote locations.

In addition, these limits are high enough to allow the implementation of a geographically split cluster in several remote locations separated from each other. As stated in Section 4.2.2.3, SCTP parameters could be tuned to increase these limits, although this would entail longer times of call setup and tear down, thus reducing the QoS.

Finally, several geographically split cluster topologies were tested using the Canada laboratory virtual machines, thus verifying that the performance is indeed good when spreading the Ericsson MSC-S blade cluster over several remote locations. However, a new traffic generator would need to be developed in order to demonstrate a good performance with a higher traffic rate.

4.2.4. Cluster scalability test

As stated in Section 2.4.3, one of the main advantages of the Ericsson MSC-S BC is its high scalability. This scalabilty allows the addition and removal of MSC-S blades to/from an existing cluster with almost no cost to the performance of the system. However, this functionality needed to be tested for the Cloud solution, where network impairments are present in the network.

In this section, first, the processes followed in order to add or remove a MSC-S blade to/from an existing cluster are briefly described. Second, the tests conducted to demonstrate the correct operation of these processes in the Cloud solution are described and their results presented. Finally, the results obtained are discussed, and conclusions are drawn from them.

4.2.4.1 Theoretical introduction

To start with, it is important to note that the processes followed when adding or removing a blade to/from an exisiting Ericsson MSC-S BC are different from each other. Therefore, these processes are described separarely below.

MSC-S blade addition

Several steps conform the process of adding a MSC-S blade to an existing cluster:

- First, the new blade must be configured as shown in Appendix A.3.
- Then, the new blade needs to be added to the exisiting cluster following the manual in Appendix A.5. When the last command is issued, a process known as *cloning* would take place prior to adding the new blade to the cluster. In

the *cloning* process, the new blade *clones* the software dump from the leader of the existing cluster[3], so as to prevent inconsistencies when having blades executing different software dump releases. This data transfer is done through the VLAN 171.

- Finally, once the *cloning* process is terminated, the new MSC-S blade would become "active" in the cluster. Then, subscribers would be registered in the new MSC-S blade, and it would be ready to serve traffic as part of the Ericsson MSC-S BC.

MSC-S blade removal

The process by which an "active" MSC-S blade is removed from an existing Ericsson MSC-S BC is described in Appendix A.6. Once the first command (which isolates the blade from the traffic that the cluster is serving) takes effect, a subscriber registration process takes place to redistribute subscribers in the remaining MSC-S blades.

4.2.4.2 Laboratory setup

The laboratory setup chosen for the scalability tests is the same used in Section 4.2.3.2 (see Figures 4.13 and 4.14 for the Canada laboratory and Stockholm laboratory B setups, respectively). In addition, different combinations of external and geographically split clusters were set up in order to perform the scalability tests. The prototype MSC-S blades utilized are indicated for every test.

For every test case, the cluster was serving a low traffic rate (50 GSM calls per second and 50 UMTS calls per second), so as to observe the impact the blade addition or removal has in the traffic.

4.2.4.3 Blade addition test

First of all, the addition of a MSC-S blade to an existing Ericsson MSC-S BC was tested in the hybrid cluster, external cluster, and geographically split cluster topologies when no network impairments stood between the *cloned* blade and the cluster leader. The next step we took consisted of testing the addition of a new blade to an existing cluster when network impairments stood between the *cloned* blade and the cluster leader.

[3]The leader of the cluster is designated when the Ericsson MSC-S cluster is created. Specifically, the blade to which the MSC-S application is specified first would be the cluster leader.

Blade addition test without network impairments

The purpose of this test was to check whether the implementation of the Ericsson MSC-S BC with prototype MSC-S blades out of the rack negatively affected the scalability functionality in the tested topologies. The topology descriptions, tests, and results are compiled in Table 4.8.

Table 4.8. Compilation of tests where a MSC-S blade was added to an existing Ericsson MSC-S BC with no network impairments between the *cloned* blade and the leader.

Topology description	Test	Result
Active cluster with racked MSC-S blades 1, 2 and 3 in the Stockholm laboratory A.	Racked MSC-S blade 4 is added to the cluster.	The *cloning* process takes ∼20 seconds. Call failures: 2347 GSM calls and 2643 UMTS calls. The blade joins the cluster in "active" state.
Active cluster with racked MSC-S blades 1, 2 and 3 in the Stockholm laboratory A.	Prototype MSC-S blade 13 is added to the cluster.	The *cloning* process takes ∼1 minute and 8 seconds. The blade does not always join the cluster in "active" state (see Section 4.2.5). When it joins, the call failures are: 3043 GSM calls and 3176 UMTS calls.
Active external cluster with the prototype MSC-S blades 9, 10 and 11 in the Canada laboratory.	Prototype MSC-S blade 12 is added to the cluster.	The *cloning* process takes ∼30 seconds. The blade does not always join the cluster in "active" state (see Section 4.2.5). When it joins, the call failures are: 2517 GSM calls and 2843 UMTS calls.
Active geographically split cluster with MSC-S the prototype MSC-S blades 9 and 10 in the Canada laboratory, and 13 in the Stockholm laboratory B.	Prototype MSC-S 11 is added to the cluster.	The *cloning* process takes ∼30 seconds. The blade does not always join the cluster in "active" state (see Section 4.2.5). When it joins, the call failures are: 1910 GSM calls and UMTS 1937 calls.

Blade addition test with network impairments

Given that the protocol used in the software dump transference is reliable and, hence, packet loss would not corrupt the tranferred data, only delay was added between the *cloned* blade and the cluster leader. Therefore, the purpose of this test was to evaluate whether the presence of delay between the *cloned* blade and the cluster leader caused any trouble to the *cloning* process. The topology used for this test consisted of an active cluster with the racked MSC-S blades 1, 2 and 3, and the external prototype MSC-S blade 16 (as in Figure 4.10).

The first times the test was conducted, a timer that limited the *cloning* time to 3 minutes was detected. The timer was removed as this action did not affect the performance of the cluster but for allowing longer *cloning* times. This issue is further explained in Section 4.2.5.

Then, the test was conducted several times with several different delays. With every delay, after the *cloning* process was complete, the prototype MSC-S blade activation ended with different results: "recovery" , "passive" , and "active". After analyzing this result, we discovered that it was caused by a variable outcome of the software dump consistency check performed after the *cloning* process was finished, and not to the network configuration nor to the additional network delay. Therefore, as it was stated in Section 1.6, this issue was out of the scope of this project and is suggested as future work.

4.2.4.4 Blade removal test

The removal of a MSC-S blade from an existing Ericsson MSC-S BC was tested in the racked cluster, hybrid cluster, external cluster, and geographically split cluster topologies. As this action did not entail any data transfer process across the network, it did not matter whether there was any delay present between blades or between the cluster and the rest of the system. The topology descriptions, tests, and results are compiled in Table **??**.

Table 4.9. Compilation of tests where a MSC-S blade was removed from an existing Ericsson MSC-S BC.

Topology description	Test	Result
Active cluster with racked MSC-S blades 1, 2, 3 and 4 in the Stockholm laboratory A.	Racked MSC-S blade 1 is removed from the cluster.	Call failures: 2675 GSM calls and 2664 UMTS calls. The blade leaves the cluster correctly.
Active hybrid cluster with racked MSC-S blades 1, 2 and 3 in the Stockholm laboratory A, and prototype MSC-S blade 13 in the Stockholm laboratory B (without added delay).	Prototype MSC-S blade 13 is removed from the cluster.	Call failures: 3549 GSM calls and 3601 UMTS calls. The blade leaves the cluster correctly.
Active hybrid cluster with racked MSC-S blades 1, 2 and 3 in the Stockholm laboratory A, and prototype MSC-S blade 13 in the Stockholm laboratory B (without added delay).	Racked MSC-S blade 1 is removed from the cluster.	Call failures: 1855 GSM calls and 1806 UMTS calls. The blade leaves the cluster correctly.
Active external cluster with prototype MSC-S blades 9, 10, 11 and 12 in the Canada laboratory.	Prototype MSC-S blade 12 is removed from the cluster.	Call failures: 2086 GSM calls and 2090 UMTS calls. The blade leaves the cluster correctly.
Active geographically split cluster with prototype MSC-S blades 9, 10 and 11 in the Canada laboratory, and 13 in the Stockholm laboratory B.	Prototype MSC-S blade 11 is removed from the cluster.	Call failures: 1838 GSM calls and UMTS 1884 calls. The blade leaves the cluster correctly.
Active geographically split cluster with prototype MSC-S blades 9, 10 and 11 in the Canada laboratory, and 13 in the Stockholm laboratory B.	Prototype blade 13 is removed from the cluster.	Call failures: 1036 GSM calls and UMTS 1628 calls. The blade leaves the cluster correctly.

4.2.4.5 Cluster migration test

Considering the results obtained in Sections 4.2.4.3 and 4.2.4.4, a cluster migration of the Ericsson MSC-S BC was tested. As the blade activation process when network impairments were present between the *cloned* and the cluster leader was unstable, the cluster migration was only tested between the Stockholm laboratories A and B (in both directions) without additional network impairments. The migration proceeding consisted of first adding a MSC-S blade in the new location of the cluster, and then removing a MSC-S blade from the old location of the cluster. This process was iterated until the whole cluster was placed in the new location.

First, the cluster migration test from the Stockholm laboratory B (external cluster with prototype MSC-S blades 13, 14 and 15. For the network topology see Figure 4.14) to the Stockholm laboratory A (traditional racked blades 1, 2 and 3) was conducted. The steps taken throughout the cluster migration, and the results of every step are compiled in Table 4.11.

Then, the cluster migration test from the Stockholm laboratory A (racked MSC-S blades 1, 2 and 3) to the Stockholm laboratory B (external cluster with prototype MSC-S blades 13, 14, and 15. For the network topology see Figure 4.14) was conducted. It is important to note that, when adding the prototype MSC-S blades to the cluster, the issue commented in Section 4.2.5 sometimes appeared. Therefore, the blade addition process was repeated until the added prototype MSC-S blade joined the cluster in "active" state. The steps taken throughout the cluster migration, and the results of every step are compiled in Table 4.12.

Table 4.10. Description of every step taken throughout the Ericsson MSC-S BC migration from the Stockholm laboratory B to the Stockholm laboratory A.

Step	Result
Racked MSC-S blade 1 is added to the cluster.	The *cloning* process takes ∼2 minutes and 15 seconds. Call failures: 1381 GSM calls and 1442 UMTS calls. The blade joins the cluster in "active" state.
Prototype MSC-S blade 13 is removed from the cluster.	Call failures: 2844 GSM calls and 2922 UMTS calls. The blade leaves the cluster correctly.

Table 4.11. Description of every step taken throughout the Ericsson MSC-S BC migration from the Stockholm laboratory B to the Stockholm laboratory A. (cont.)

Step	Result
Racked MSC-S blade 2 is added to the cluster.	The *cloning* process takes ∼1 minute and 49 seconds. Call failures: 1471 GSM calls and 1607 UMTS calls. The blade joins the cluster in "active" state.
Prototype MSC-S blade 14 is removed from the cluster.	Call failures: 3438 GSM calls and 3670 UMTS calls. The blade leaves the cluster correctly.
Racked MSC-S blade 3 is added to the cluster.	The *cloning* process takes ∼21 seconds. Call failures: 1855 GSM calls and 2052 UMTS calls. The blade joins the cluster in "active" state.
Prototype MSC-S blade 15 is removed from the cluster.	Call failures: 4276 GSM calls and 4289 UMTS calls. The blade leaves the cluster correctly.

Table 4.12. Description of every step taken throughout the Ericsson MSC-S BC migration from the Stockholm laboratory A to the Stockholm laboratory B.

Step	Result
Prototype MSC-S blade 13 is added to the cluster.	The *cloning* process takes ∼1 minute and 15 seconds. When the blade joins the cluster in "active" state the call failures are: 4542 GSM calls and 5086 UMTS calls.
Blade 1 is removed from the cluster.	Call failures: 1915 GSM calls and 1843 UMTS calls. The blade leaves the cluster correctly.
Blade 14 is added to the cluster.	The *cloning* process takes ∼1 minute and 13 seconds. When the blade joins the cluster in "active" state the call failures are: 3158 GSM calls and 3253 UMTS calls.
Blade 2 is removed from the cluster.	Call failures: 1720 GSM calls and 1787 UMTS calls. The blade leaves the cluster correctly.

Table 4.13. Description of every step taken throughout the Ericsson MSC-S BC migration from the Stockholm laboratory A to the Stockholm laboratory B. (cont.)

Step	Result
Blade 15 is added to the cluster.	The *cloning* process takes ~1 minute and 15 seconds. When the blade joins the cluster in "active" state the call failures are: 2851 GSM calls and 3003 UMTS calls.
Blade 3 is removed from the cluster.	Call failures: 1505 GSM calls and 1570 UMTS calls. The blade leaves the cluster correctly.

4.2.4.6 Discussion

Given the results obtained in the tests, it can be observed that the cluster scalability functionality is not negatively affected by the introduction of network impairments in the system. However, some issues were encountered when adding prototype MSC-S blades to an active cluster that could not be solved, as they were out of the scope of this project (see limitations in Section 1.6). These incidents are not related to the network configuration used to move the Ericsson MSC-S blades to the Cloud, nor to the network impairments introduced in the system to simulate that part of a Cloud environment. As a matter of fact, these minor inconsistencies are due to prototyping matters and, therefore, can be debugged later on in order to make the cluster scalability functionality fully compatible with Cloud environments.

The removal of MSC-S blades from an active Ericsson MSC-S BC was successful in every case that was tested. Therefore, neither the presence of network impairments nor the network configuration affected this process.

Considering the prototyping issues mentioned above, cluster migration tests were successfully conducted from the Stockholm laboratory B to the Stockholm laboratory A and backwards. Hence, it was demonstrated that a cluster migration while the Ericsson MSC-S BC was serving live traffic was possible.

Finally, as it can be seen in Tables **??**, 4.8, 4.11 and 4.12, some calls fail when a MSC-S blade is either added or removed to/from the cluster. It is important to note that just when a new MSC-S blade is successfully activated after the *cloning* process, or when an "active" MSC-S blade is isolated from the traffic prior to be removed from a cluster, a subscriber registration/redistribution starts in the cluster, and it is not until this process finishes that calls stop failing. However, the service uptime percentage would be over the required values considering the high amount of calls that an Ericsson MSC-S BC serves over time.

Additionally, in the tests, the off-the-rack MSC-S blades were configured to store more subscriber information than the racked blades; hence, when an off-the-rack MSC-S blade was added or removed to/from the cluster, there were more call failures than when the addition or removal to/from the cluster was performed with a racked blade, as the subscriber registration/redistribution process was longer in the former case.

4.2.5. Difficulties

The complications that arose throughout the tests conducted in Sections 4.2.1, 4.2.2 and 4.2.3 are described below divided depending on whether they were associated to the network aspects of the tested Ericsson MSC-S BC implementations, or to prototyping issues (not covered in this thesis project. See Section 1.6). In addition, when a solution was provided to the problem, it is presented.

4.2.5.1 Network issues

The following issues were encountered when testing the Ericsson MSC-S BC using the designed network configuration and adding network impairments to the system (delay and packet loss).

RAN traffic generator: As mentioned in Section 4.2.2.1, an important issue was found out regarding the RAN traffic generator used in the tests. The high delays caused an important increment of the call establishment and call release times, thus increasing the number of in-progress calls. When this number of in-progress calls increased too much, the traffic generator was unable to keep the state of all of them and dropped an important amount of calls, thus the tests' results became flawed. Unfortunately, the traffic generator source code was not available due to privacy constraints, and an enhanced RAN traffic generator deployment was out of the scope of this project (see Section 1.6). Therefore, the rest of the tests were performed using a low call rate so that this issue did not affect the measurements.

Ethernet interface queue length: When the first traffic tests were performed, unexpected call failures appeared with low delay values. After some time of troubleshooting we noticed that the ethernet interfaces' queues were overloaded, and therefore dropping packets. The default interface queue size was 100 packets, and the value was increased to **5000** so that there was no option for these buffers to overflow.

Size Alteration Events: While the testing was performed the command described in Appendix A.8.6 was run in order to detect possible size alteration events[4] (SAEs). Several were detected, and the affected memory blocks sizes

[4]A Size Alteration Event pops up whenever a static memory block is filled up, and therefore its size needs to be increased.

were increased to support high delays and packet loss. The blocks that were modified, and their new sizes are presented in Appendix A.11.1.

SCTP buffer size: Since the network impairments cause an important increment of SCTP retransmissions, thus overflowing the SCTP buffers, it was necessary to increase the SCTP buffer size to support acceptable delay and packet loss values. The chosen value was **10 000 000 bytes**, as it was considered to be enough for our purposes. However, the SCTP buffer size could be set to any other value as long as there is enough memory available.

4.2.5.2 Prototyping issues

The following prototyping issues were encountered when conducting the tests compiled in Section 4.2. As stated in Section 1.6 the prototyping issues were out of the scope of this project and no solutions were provided for them.

Software dump consistency check failure: A prototyping issue was detected when testing the scalability functionality of the Ericsson MSC-S BC (see Section 4.2.4). When the *cloning* process that takes place in the addition of a prototype MSC-S blade to an active cluster was delayed more than the **3 minutes** established in the prototype, the software consistency check performed afterwards sometimes failed. This unexpected behavior was due to the fact that this MSC-S application prototype was not developed with long *cloning* times in mind.

MSC-S blade passive: The MSC-S application supervision system that decides whether a MSC-S blade newly added to the cluster is to be activated and contribute to serving traffic did not have a consistent behavior when handling prototype MSC-S blades. Whenever a new prototype MSC-S blade was added to an active cluster after a successful *cloning* process, its state was either "passive" or "active" with no defined pattern to be observed. This inconsistency also appeared when a cluster was created with prototype MSC-S blades, where one or more of the prototype MSC-S blades forming the cluster sometimes remained in "passive" state after the activation process.

4.3. Fault recovery

This section aims to focus on the fault situations that might appear when the Ericsson MSC-S BC is operating. The cluster protocols include recovery schemes that will recover the cluster when such situations occur. Hence, faults were introduced intentionally in the system in order to demonstrate the correct operation of the cluster recovery schemes when the MSC-S prototype blades are located in a Cloud environment. The recovery schemes were tested for two types of faults: single blade link fault, and partition fault.

The recovery scheme is different depending on whether the fault is temporary (short-time fault) or permanent (long-time fault). Therefore the two fault types stated above were tested for both temporary and permanent faults.

In this section, first, a theoretical introduction to the fault recovery mechanisms is provided. Second, the single blade link fault tests and the partition fault tests are described for both temporary and permanent faults and their results are presented. Then, the difficulties encountered during the test conduction are stated. Finally, the results obtained in the tests are discussed, and conclusions are drawn from them.

4.3.1. Theoretical introduction

As stated above, the fault recovery mechanisms that take on when either a single blade fault or a partition fault occur are different. In addition, the recovery mechanisms also depend on whether the fault is considered temporary or permanent. There is a time limit that differentiates a temporary fault from a permanent fault. Every fault recovery mechanism is explained separately below.

4.3.1.1 Single blade link fault

A single blade link fault occurs whenever a single MSC-S blade fails or loses connectivity with the rest of the system, thus being completely isolated from the rest of the MSC-S cluster. The fault recovery mechanism depends on whether the fault is temporary or permanent. These two cases are explained below:

Temporary fault: When a temporary single blade link fault occurs, the affected blade automatically restarts and switches to "recovery" state. Then, as soon as the connectivity is recovered, the faulty MSC-S blade returns to the cluster in "active" state and performs as it did before the fault happened.

Permanent fault: When a permanent single blade link fault occurs, the affected blade automatically restarts and switches to "recovery" state. Then, when the connectivity is recovered, the faulty MSC-S blade is automatically reinserted in the cluster using the *cloning* process explained in Section 4.2.4.1.

4.3.1.2 Partition fault

A partition fault occurs whenever one or more link failures divide the MSC-S cluster in several separated cluster partitions with no connectivity between each other. This sort of fault receives also the name of "split-brain" fault. The fault recovery mechanism depends on whether the fault is temporary or permanent. These two cases are explained below:

Temporary fault: When a temporary partition fault occurs, the cluster partition with the higher number MSC-S blades remains "active". If there is a tie in the number of MSC-S blades, the cluster partition with the cluster leader remains "active". The rest of the MSC-S blades switch to the "recovery" state. Then, when the connectivity is recovered, the cluster partitions in "recovery" state are automatically reinserted in the cluster in "active" state.

Permanent fault: When a permanent partition fault occurs, the cluster partition with the higher number MSC-S blades remains "active". If there is a tie in the number of MSC-S blades, the cluster partition with the cluster leader remains "active". The rest of the MSC-S blades switch to the "recovery" state. Then, when the connectivity is recovered, the cluster partitions in "recovery" state are automatically reinserted in the cluster using the *cloning* process explained in Section 4.2.4.1.

4.3.2. Fault recovery tests

This section includes the tests carried out for the fault recovery mechanisms of an implementation of the Ericsson MSC-S BC using prototype MSC-S blades located in a simulated Cloud environment. First, the single blade link fault test is described for both temporary and permanent faults. Then, the partition fault test is explained for both temporary and permanent faults.

All these tests were conducted with no mobile signalling traffic running in the network, as the fault recovery procedures are independent from the traffic and, additionally, the purpose of these tests was to demonstrate the correct operation of the fault recovery schemes themselves. Furthermore, every fault test was first conducted using the Stockholm laboratories A and B, and introducing delays with the Netem tool (see Appendix A.10). Since the protocols used in these fault recovery schemes are reliable, they were tested against network delay as the only impairment. Then, the same tests were conducted using the Canada laboratory. The hybrid cluster topology was selected for all the fault recovery tests as it permited to test both racked and prototype MSC-S blades in fault situations, thus allowing a comparison between results.

4.3.2.1 Laboratory setup

Every prototype MSC-S blade was configured with two interfaces following the Appendix A.3. Therefore, the Stockholm laboratory A blades were set up as illustrated in Figure 4.14, while the Canada laboratory blades were set up as depicted in Figure 4.13. These laboratory configurations enabled the possibility of building different hybrid cluster configurations to test both the single blade link fault and the partition fault recovery schemes. The cluster activation was done as described in Appendix A.4.

The additional network delay was introduced in the *Bridge* machine between the Stockholm laboratory A, and the Stockholm B and Canada laboratories, as illustrated in Figure 4.10. In addition, the link faults needed to reproduce the two tested fault types (single blade link and partition faults) were simulated disconnecting machine Ethernet interfaces using a simple script that brought down the interfaces, left them down for some specified time, and brought them up again.

4.3.2.2 Single link blade fault test

The main purpose of these tests was to demonstrate the correct operation of the single blade link fault recovery schemes when the system is exposed to the network impairments characteristic of a Cloud environment. The single blade link fault situation was tested in two different cluster configuration cases for both temporary and permanent faults.

Temporary fault

First, the temporary single blade link fault recovery schemes were tested in a hybrid cluster topology built using the Stockholm laboratory A and B. The blades that formed the cluster were the prototype MSC-S blades 13, 14 and 15 in the Stockholm laboratory B, and the racked MSC-S blade 1 in the Stockholm laboratory A.

This scenario was tested with a RTT value of 160 ms between the Stockholm laboratories A and B (close to the limit calculated in Section 4.2.2), while a single blade link fault of 0 to 250 seconds was reproduced in one of the blades. The link downtime value of **250 seconds** was found out to be the limit that differentiated a temporary single blade link fault from a permanent single blade link fault in the system. Two different tests were conducted with this topology:

- The single blade link fault was introduced in the prototype MSC-S blade number 15 (located in the Stockholm laboratory B). In this case, the behavior of the recovery scheme was always correct from the network perspective. Independently of the delay introduced and the link downtime value, the prototype MSC-S blade 15 restarted as expected and, once the restart was finished and the connectivity was recovered, it normally came back to the cluster in "active" state. However, due to prototyping aspects, there were some cases where the prototype MSC-S blade came back to the cluster in "passive" state. This issue is clarified in Section 4.3.3.
- The single blade link fault was introduced in the MSC-S blade number 1 (racked blade located in Stockholm laboratory A). For every link downtime value, the MSC-S blade 1 restarted and, once the restart was finished and the connectivity recovered, rejoined the cluster in "active" state. Therefore, the

performance of the fault recovery scheme that takes over in this situation is correct.

Then, the temporary single blade link fault recovery schemes were tested in a hybrid cluster topology built using the Stockholm laboratory A and the Canada laboratory. The blades that formed the cluster were the prototype MSC-S blades 9, 10 and 11 in the Canada laboratory, and the MSC-S blade 1 in the Stockholm laboratory A. Since the Canada laboratory network impairments calculated in Section 4.1.4.1 were enough for our purposes, no additional network impairments were introduced for these tests. These scenarios were tested reproducing single blade link faults of 0 to 250 seconds in a blade. Two different tests were carried out with this topology:

- The single blade link fault was introduced in the prototype MSC-S blade number 11 (located in the Canada laboratory). In this case, the behavior of the recovery scheme was always correct from the network perspective. Independently of the link downtime value, the prototype MSC-S blade 11 restarted as expected and, once the restart was finished and the connectivity recovered, the blade normally came back to the cluster "active" state. However, due to prototyping aspects, there were sometimes that the prototype MSC-S blade came back to the cluster in "passive" state. This issue is the same that occurred in the previous Stockhholm laboratory B test and is clarified in Section 4.3.3.
- The single blade link fault was introduced in the MSC-S blade number 1 (racked blade located in Stockholm laboratory A). For every link downtime value the MSC-S blade 1 restarted and, once the restart was finished and the connectivity recovered, rejoined the cluster in "active" state. Therefore, the performance of the fault recovery scheme that takes over in this situation is correct.

Permanent fault

First, the permanent single blade link fault recovery schemes were tested in the hybird cluster topology configured using the Stockholm laboratory A and B. The blades that formed the cluster were the prototype MSC-S blades 13, 14 and 15 in the Stockholm laboratory B, and the racked MSC-S blade 1 in the Stockholm laboratory A.

This scenario was tested with a RTT value of 160 ms between the Stockholm laboratories A and B (close to the limit calculated in Section 4.2.2), while a single blade link fault with link downtimes higher than 250 seconds (up to 300 seconds) was reproduced in one of the blades. Two different tests were conducted with this topology:

- The single blade link fault was introduced in the prototype MSC-S blade number 15 (located in the Stockholm laboratory B). Independently of the delay introduced and the link downtime value, the prototype MSC-S blade 15 restarted as expected and, once the restart was finished and 250 seconds passed, switched to "recovery" state. Finally, when the link fault was removed and the connectivity recovered between the rest of the cluster and the faulty blade, the prototype MSC-S blades 15 automatically started a *cloning* process to rejoin the cluster (see Section 4.3.1.2), finally coming back in "active" state. Therefore, the performance of the fault recovery scheme that takes over in this situation is correct.

- The single blade link fault was introduced in the racked MSC-S blade number 1 (located in Stockholm laboratory A). Independently of the delay introduced and the link downtime value, the prototype MSC-S blade 1 restarted as expected and, once the restart was finished and 250 seconds passed, switched to "recovery" state. Finally, when the link fault was removed and the connectivity recovered between the rest of the cluster and the faulty blade, the prototype MSC-S blades 1 automatically started a *cloning* process to rejoin the cluster (see Section 4.3.1.2), finally coming back in "active" state. Therefore, the performance of the fault recovery scheme that takes over in this situation is correct.

Then, the permanent single blade link fault recovery schemes were tested in a hybrid cluster topology set up using the Stockholm laboratory A and the Canada laboratory. The blades that formed the cluster were the prototype MSC-S blades 9, 10 and 11 in the Canada laboratory, and the MSC-S blade 1 in the Stockholm laboratory A. Since the Canada laboratory network impairments calculated in Section 4.1.4.1 were enough for our purposes, no additional network impairments were introduced in the system for these tests. These scenarios were tested reproducing single blade link faults with link downtimes higher than 250 seconds (up to 300 seconds). Two different tests were carried out with this topology:

- The single blade link fault was introduced in the prototype MSC-S blade number 11 (located in the Canada laboratory). For every link downtime value, the prototype MSC-S blade 11 restarted as expected and, once the restart was finished and 250 seconds passed, switched to "recovery" state. Finally, when the link fault was removed and the connectivity recovered between the rest of the cluster and the faulty blade, the prototype MSC-S blades 15 automatically started a *cloning* process to rejoin the cluster (see Section 4.3.1.2), finally coming back in "active" state. Therefore, the performance of the fault recovery scheme that takes over in this situation is correct.

- The single blade link fault was introduced in the racked MSC-S blade number 1 (located in Stockholm laboratory A). For every link downtime value, the prototype MSC-S blade 1 restarted as expected and, once the restart was finished and 250 seconds passed, switched to "recovery" state. Finally, when the link fault was removed and the connectivity recovered between the rest of the cluster and the faulty blade, the prototype MSC-S blades 1 automatically started a *cloning* process to rejoin the cluster (see Section 4.3.1.2), finally coming back in "active" state. Therefore, the performance of the fault recovery scheme that takes over in this situation is correct.

4.3.2.3 Partition fault test

These tests were meant to prove the correct operation of the partition fault recovery schemes when network impairments were present in the system. The partition fault situation was tested in two different cluster configuration cases for both temporary and permanent faults.

Temporary fault

First, the temporary partition fault was tested in a hybrid cluster topology built using the Stockholm laboratory A and B. Link faults were introduced in the link that connected the *Bridge* machine to the Stockholm A laboratory in order to reproduce a "split-brain" situation between the racked MSC-S blades located in the Stockholm laboratory A and the prototype MSC-S blades located in the Stockholm laboratory B.

This scenario was tested with a RTT value of 160 ms between the Stockholm laboratories A and B (close to the limit calculated in Section 4.2.2), and with link downtimes from 0 seconds to 90 seconds (The link downtime value of **90 seconds** was discovered to be the limit that differentiated a temporary partition fault from a permanent partition fault in the system). Two different tests were carried out with a slight modification in the topology, so that the MSC-S cluster group that remained "active" after the partition fault was located once in each laboratory (see Section 4.3.1.2):

- The Ericsson MSC-S BC was formed by the racked MSC-S blades 1 and 2 in the Stockholm laboratory A, and the prototype MSC-S blades 13, 14 and 15 in the Stockholm laboratory B. For every link downtime value, when the link fault was introduced in the *Bridge* machine, the MSC-S blades 1 and 2 restarted. Then, once the restart was finished and the connectivity was recovered, both MSC-S blades 1 and 2 came back to the cluster in "active" state. Therefore, the performance of the fault recovery scheme that takes over in this situation is correct.
- The Ericsson MSC-S BC was formed by the racked MSC-S blades 1, 2 and 3 in the Stockholm laboratory A, and the prototype MSC-S blades 13 and 14

in the Stockholm laboratory B. For every link downtime value, when the link fault was introduced in the *Bridge* machine, the prototype MSC-S blades 13 and 14 restarted. Then, once the restart was finished and the connectivity was recovered, both prototype MSC-S blades 13 and 14 came back to the cluster in "active" state. Therefore, the performance of the fault recovery scheme that takes over in this situation is correct.

Then, the temporary partition fault was tested in a hybrid cluster topology built using the Stockholm laboratory A and the Canada laboratory. Since the Canada laboratory network impairments calculated in Section 4.1.4.1 were enough for our purposes, no additional network impairments were introduced between system elements. Two different tests were carried out with a slight modification in the topology, so that the MSC-S cluster group that remained "active" after the partition fault was located once in each laboratory (see Section 4.3.1.2):

- The Ericsson MSC-S BC was formed by the racked MSC-S blades 1 and 2 in the Stockholm laboratory A, and the prototype MSC-S blades 9, 10 and 11 in the Canada laboratory. For every link downtime value, when the link fault was introduced in the *Bridge* machine, the MSC-S blades 1 and 2 restarted. Then, once the restart was finished and the connectivity was recovered, both MSC-S blades 1 and 2 came back to the cluster in "active" state. Therefore, the performance of the fault recovery scheme that takes over in this situation is correct.
- The Ericsson MSC-S BC was formed by the racked MSC-S blades 1, 2 and 3 in the Stockholm laboratory A, and the prototype MSC-S blades 9 and 10 in the Canada laboratory. For every link downtime value, when the link fault was introduced in the *Bridge* machine, the prototype MSC-S blades 13 and 14 restarted. Then, once the restart was finished and the connectivity was recovered, both prototype MSC-S blades 13 and 14 came back to the cluster in "active" state. Therefore, the performance of the fault recovery scheme that takes over in this situation is correct.

Permanent fault

First, the permanent partition fault was tested in a hybrid cluster topology built using the Stockholm laboratory A and B. Different network delays were introduced in the *Bridge* machine while a link fault was introduced in the link that connected the *Bridge* machine to the Stockholm A laboratory in order to reproduce a "split-brain" situation between the MSC-S blades located in the Stockholm laboratory A and the prototype MSC-S blades located in the Stockholm laboratory B.

This scenario was tested with a RTT value of 160 ms between the Stockholm laboratories A and B (close to the limit calculated in Section 4.2.2), and with link downtimes higher than 100 seconds (up to 300 seconds). Two different tests were carried out with a slight modification in the topology, so that the "active" MSC-S cluster group after the partition fault was located once in each laboratory (see Section 4.3.1.2):

- The Ericsson MSC-S BC was formed by the racked MSC-S blades 1 and 2 in the Stockholm laboratory A, and the prototype MSC-S blades 13, 14 and 15 in the Stockholm laboratory B. For every link downtime value, when the link fault was introduced in the *Bridge* machine, the prototype MSC-S blades 13, 14 and 15 remained in "active" state, while the MSC-S blades 1 and 2 restarted. Then, once the restart was finished, since the link fault time exceeded 90 seconds and the fault was considered permanent, the MSC-S blades 1 and 2 switched to "recovery" state. Eventually, when the link fault was removed and the connectivity recovered between the two cluster groups, both MSC-S blades 1 and 2 automatically started a *cloning* process to rejoin the cluster (see Section 4.3.1.2), finally coming back in "active" state. Therefore, the performance of the fault recovery scheme that takes over in this situation is correct.

- The Ericsson MSC-S BC was formed by the racked MSC-S blades 1, 2 and 3 in the Stockholm laboratory A, and the prototype MSC-S blades 13 and 14 in the Stockholm laboratory B. For every link downtime value, when the link fault was introduced in the *Bridge* machine, the MSC-S blades 1, 2 and 3 remained in "active" state, while the prototype MSC-S blades 13 and 14 restarted. Then, once the restart was finished, since the link fault time exceeded 90 seconds and the fault was considered permanent, the prototype MSC-S blades 13 and 14 switched to "recovery" state. Eventually, when the link fault was removed and the connectivity recovered between the two cluster groups, both prototype MSC-S blades 13 and 14 automatically started a *cloning* process to rejoin the cluster (see Section 4.3.1.2), finally coming back in "active" state. Therefore, the performance of the fault recovery scheme that takes over in this situation is correct.

Then, the permanent partition fault was tested in a split cluster topology built using the Stockholm laboratory A and the Canada laboratory. Since the Canada laboratory network impairments calculated in Section 4.1.4.1 were enough for our purposes, no additional network impairments were introduced. Two different tests were carried out with a slight modification in the topology, so that the "active" MSC-S cluster group after the partition fault was located once in each laboratory (see Section 4.3.1.2):

- The Ericsson MSC-S BC was formed by the racked MSC-S blades 1 and 2 in the Stockholm laboratory A, and the prototype MSC-S blades 9, 10 and 11 in the Canada laboratory. For every link downtime value, when the link fault was introduced in the *Bridge* machine, the prototype MSC-S blades 9, 10 and 11 remained in "active" state, while the MSC-S blades 1 and 2 restarted. Then, once the restart was finished, since the link fault time exceeded 90 seconds and the fault was considered permanent, the MSC-S blades 1 and 2 switched to "recovery" state. Eventually, when the link fault was removed and the connectivity recovered between the two cluster groups, both MSC-S blades 1 and 2 automatically started a *cloning* process to rejoin the cluster (see Section 4.3.1.2), finally coming back in "active" state. Therefore, the performance of the fault recovery scheme that takes over in this situation is correct.

- The Ericsson MSC-S BC was formed by the racked MSC-S blades 1, 2 and 3 in the Stockholm laboratory A, and the prototype MSC-S blades 9 and 10 in the Canada laboratory. For every link downtime value, when the link fault was introduced in the *Bridge* machine, the MSC-S blades 1, 2 and 3 remained in "active" state, while the prototype MSC-S blades 9 and 10 restarted. Then, once the restart was finished, since the link fault time exceeded 90 seconds and the fault was considered permanent, the prototype MSC-S blades 9 and 10 switched to "recovery" state. Eventually, when the link fault was removed and the connectivity recovered between the two cluster groups, both MSC-S blades 9 and 10 automatically started a *cloning* process to rejoin the cluster (see Section 4.3.1.2), finally coming back in "active" state. Therefore, the performance of the fault recovery scheme that takes over in this situation is correct.

4.3.3. Prototyping issue

A prototyping issue arose throughout the tests conducted in Section 4.3.2. Such an issue was encountered when the temporary single link fault was simulated over an external prototype MSC-S blade.

In the tests conducted in Section 4.3.2.2, when an external prototype MSC-S blade that belonged to an active cluter was isolated from the rest of the system due to a single link fault, the fault recovery scheme did not fully accomplish its purpose of recovering the blade. As explained in Section 4.3.1, the affected prototype MSC-S blade must first restart, which is correctly done in the tests. Then, when the link fault is removed and the connectivity is recovered, the affected blade would rejoin the cluster in "active" state. However, this step was not always fulfilled in the tests and, sometimes, the blade rejoined in "passive" state instead of "active" state. The reason why the affected prototype MSC-S blade sometimes rejoined in "passive" state was associated to a prototyping issue. Therefore, a solution to this incident was out of the scope of this project (see Section 1.6).

4.3.4. Discussion

Considering the results obtained in the tests of Section 4.3.2, we can conclude that not the network impairments characteristic of a Cloud environment, nor the network configuration used to adapt the Ericsson MSC-S BC to a Cloud environment affect the Ericsson MSC-S BC fault recovery mechanisms. The only hindrance that prevents the temporary fault recovery mechanisms from succeeding from a functional point of view is a prototyping issue that has nothing to do with the networking aspects of the implementation.

From a network perspective, the cluster is able to recover from every type of fault that was tested: single blade link fault and partition faults for both temporary and permanent faults. Therefore, the tested Ericsson MSC-S BC fault recovery functions are potentially able to put up with the network restrictions that a Cloud implementation would introduce.

In addition, the results obtained in all the permanent fault test cases, for both single blade link fault and partition faults, indicate that the insertion of prototype MSC-S blades in an active cluster after a *cloning* process is indeed possible. In these cases, the reinserted prototype MSC-S blades join the active cluster always in "active" state, thus avoiding the prototyping issue that was present in the tests conducted in Section 4.2 and that is briefly described in Section 4.2.5.

Chapter 5

Service providers' survey

As it was mentioned in Sections 1.3 and 1.4, one of the main goals of this master's thesis project was an evaluation of current Cloud computing capabilities in order to determine whether the Ericsson MSC-S BC Cloud solution would be feasible.

A survey of currently available Cloud providers was conducted. In order to gather all the necessary information, every one of these Cloud providers was contacted by e-mail and phone, and the information available via their websites was extended.

5.1. Technical requirements

The chosen Cloud provider needed to fulfill several technical requirements. These requirements were fixed by the software used and they can be splitted into two different groups: hardware requirements and network requirements.

5.1.1. Hardware requirements

The selected Cloud provider must meet, at least, the following hardware requirements:

- Availability of high-performance Linux blades with Intel architecture.
- 16 GB of on-board memory per blade.
- 2 GB of disk availability per blade.
- Two processors of 2 GHz, based on Intel x86 architecture.
- Linux kernel version 2.6.36 or later.
- Gigabit Ethernet technology.

5.1.2. Network requirements

The blades that are to be moved to the Cloud have some requirements regarding the network. These blades must have the following minimum network characteristics to be able to executing the prototype MSC software:

- 50 Mbps of bandwidth
- Support for multicast (both Ethernet and UDP varieties)
- Virtual machines with VLAN support at the software level
- Port selection options for virtualized blades

5.2. Privacy and Confidentiality demands

Due to the fact that proprietary software is used to implement the prototype MSC-S BC, there were several privacy and confidentiality demands that the selected provider should fulfill.

The prototype MSC-S BC was implemented by Ericsson and the software is proprietary. To avoid the possibility of access to this software by intruders, strong privacy and confidentiality policies were demanded. Any Cloud service provider that does not have a SLA which ensures the privacy and confidentiality of the software uploaded to their servers was automatically discarded.

5.3. Providers

Nowadays there are many companies that provide Cloud services. Due to the fact that Cloud storage and Cloud computing have become central to the future of the Information Technology industry, many companies have decided to invest money in developing such services. As a result, a survey including all the companies that provide these services would be huge, so a selection of the most important ones was made to realize the survey.

Initially several companies were suggested by our supervisors. These companies were well-known and important in this industry, but they were not the only ones that we considered. These companies were the following: *Amazon*, *IBM*, *Google App Engine*, *Telia Sonera*, *GoGrid*, *HP*, *Oracle/Sun*, and *Rackspace.*

After studying the products offered by these companies, several alternatives were found by searching the web and these companies were also considered. These companies were the following: *Savvis*, *Contegix*, *Fujitsu*, *Telstra*, *British Telecom*, *Deutsche Telecom*, *AT&T*, and *Cisco.*

5.4. Provider selection

To make the provider selection for our testing, a detailed survey has carried out. All the characteristics considered are listed in Appendix C.

Initially sixteen providers (those listed in Section 5.3) were considered. After studying the services offered by these companies, eight of them were discarded because they were not appropriate for this project. The eight companies remaining after this initial selection were: *GoGrid*, *Rackspace*, *Savvis*, *IBM*, *Contegix*, *Amazon*, *Fujitsu*, and *AT&T*. Details of the results of this first survey are given in Appendix C.1.

After this first selection, the sales representatives of these eight companies were contacted and asked for additional information. This additional information was needed to be sure that the provider that was to be selected could meet the requirements for our testing. From these eight companies, six were selected because the others did not fulfill the system requirements. All the information received from the representatives of these six companies can be found in Appendix C.2.

Then, after this exhaustive survey among the different considered providers, a selection was made. There were two companies that completely fulfill the project's requirements: *GoGrid* and *Rackspace*. Eventually a decision was made to select **Rackspace**.

5.5. Selected provider bench-marking

Once the Cloud service provider is selected, some measurements need to be done. These measurements are required to test the provider and to check if the service offered by this provider is suitable for running a Ericsson prototype MSC-S blade on it.

In this section, the scripts and tools that are used for evaluate the provider's characteristics are described. Then, to finalize, the performance and results obtained are exposed.

5.5.1. Testing scripts

There are five different kinds of tests that were used to evaluate the characteristics offered by the provider. To test the provider to decide if their offering was suitable for hosting the prototype MSC application, two servers are hired and these tests are carried out. The two servers were continuously hired for the tested period.

Initially, the servers both had 2048 MB of RAM memory and the test focused on communication. The communication between the servers within the provider's network was tested as well as the communication between one server in the provider and the *Isolated machine*. These tests are done, respectively, to simulate communication between blades in the same cluster and same location, and to simulate the communication between the blades and the *Bridge machine* in Stockholm.

Finally, to test the capacity of the biggest server that the provider offered[1], one of the servers was replaced by a server that had 15.08 GB of RAM memory. This server was used to test the read memory latency as well as the CPU capacity in terms of processing. These test are done to ensure that the servers offered by the provider were capable of running the prototype MSC application without any problem regarding communication or processing capacity.

5.5.1.1 UDP test

This test was done to test the UDP communication in the system. The UDP test exploits the capabilities of one tool called *iperf*. This tool is a piece of free software available for any Linux machine and very useful for testing communications.

Iperf

Iperf is an alternative for measuring maximum TCP and UDP performance in terms of bandwidth. It is a tool that allows the tuning of different parameters and UDP characteristics, and it reports different measures, as bandwidth, delay jitter and datagram loss.

The use of *iperf* is very simple. The main functionalities used in UDP test are going to be described below, but extended functionality and further explanations can be found in [74].

First of all, the application has to be installed in both the server and the client machines. Once it is installed, the following command needs to be run in the server machine:

```
iperf -s -u
```

Where the flags `-s` and `-u` indicate, respectively, that this machine is the server, and that a UDP measurement is being done.

[1]When the provider was tested, the biggest server offered by Rackspace was a server with 15.08 GB of RAM memory. However, they are now able to offer a server with 30 GB of RAM memory, more suitable with the blade's memory requirement.

Once it is done, the next command needs to be run in the client machine:

```
iperf -c address -u -b bandwidth
```

Where the fields **address** and **bandwidth** has to be filled, respectively, with the IP address of the server which it is wanted to make the measure with, and with the amount of bandwidth that want to be tested.

Script

The functionalities described in the previous headland are the functionalities used in the scrip created for this test. This script was used in the client machine and needed to have the program running in the server machine. The script was capable of doing a specified number of measures with a desired interval time between measures, and it was used during this thesis with the purpose of having the test gathering information about the network capacity during a period of 24 hours.

The script is written by the authors of this thesis. The script is shown below:

```bash
#!/bin/bash

# Use: ./UDP_script ADDRESS BANDWIDTH MEASURES SLEEP_TIME

address=$1
bw=$2
measures=$3
sleeptime=$4

rm /root/Tests/output_UDP
rm /root/Tests/udp_BW.txt
rm /root/Tests/udp_jitter.txt
rm /root/Tests/udp_lost.txt

for ((i=1; i <= $measures ; i++))
do
   iperf -c $address -u -b $bw > /root/Tests/output_UDP
   awk -f /root/Tests/awk_udp /root/Tests/output_UDP
   sleep $sleeptime
done
```

Listing 5.1: UDP script

As it can be seen, this script uses another script called *awk_udp*, which is only used as a parser for the *iperf* output to gather the results in another file called "*udp_lost.txt*". The code of the *awk_udp* script is written by the authors of this thesis and it is shown in Appendix D.1.

5.5.1.2 TCP test

This test was done to test the TCP communication in the system. The TCP test exploits the capabilities of one tool called *iperf*. This tool is a piece of free software available for any Linux machine and very useful for testing communications.

Iperf

As it is said in the previous subsection, *iperf* is an alternative for measuring maximum TCP and UDP performance in terms of bandwidth.

The use of *iperf* is very simple. The main functionalities used in TCP test are going to be described below, but extended functionality and further explanations can be found in [74].

First of all, the application has to be installed in both the server and the client machines. Once it is installed, the following command needs to be run in the server machine:

```
iperf -s
```

Where the flag -s indicates that this machine is the server.

Once it is done, the next command needs to be run in the client machine:

```
iperf -c address -t timing
```

Where the fields **address** and **timing** has to be filled, respectively, with the IP address of the server which it is wanted to make the measure with, and with the test duration time in seconds.

Script

The functionalities described in the previous headland are the functionalities used in the scrip created for this test. This script was used in the client machine and needed to have the program running in the server machine. The script was capable of doing a specified number of measures with a specific duration time for each measure, and with a desired interval time between measures. The script was used during this thesis with the purpose of having the test recovering information about the network capacity during a period of 24 hours.

The script is written by the authors of this thesis. The script is shown below:

```bash
#!/bin/bash

# Use: ./TCP_script ADDRESS TIMING MEASURES SLEEP_TIME

address=$1
timing=$2
measures=$3
sleeptime=$4

rm /root/Tests/output_BW
rm /root/Tests/bandwidth.txt

for ((i=1; i <= $measures ; i++))
do
   iperf -c $address -t  $timing > /root/Tests/output_BW
   awk -f /root/Tests/awk_BW /root/Tests/output_BW
   sleep $sleeptime
done
```

Listing 5.2: TCP script

As it can be seen, this script uses another script called *awk_BW*, which is only used as a parser for the *iperf* output to gather the results in another file called "*bandwidth.txt*". The code of the *awk_BW* script is written by the authors of this thesis and it is shown in Appendix D.2.

5.5.1.3 PING test

This test uses a common tool in communication's world, called *ping*. *Ping* command is used to know the RTT beetween two nodes. It is a command present in any machine with any operating system, whose most common way of use is the following:

```
ping XX.XX.XX.XX
```

Where **XX.XX.XX.XX** represents the IP address that wants to be reached.

Script

For this script, a variation of this command was used. This variation is called *ping flooding* and consists in overwhelming the receiver with *ping* packets in order to test the RTT in a loaded network. To use the *ping* command in the "flooding" variant, it is only needed to add the flag `-f` to the common use form. In the script, two more options were used: the test time (flag `-w`) and the sent packets' size (flag `-s`), which allowed to fix a specific amount of time for the flooding and a specific size for the packets that were sent to the receiver. Note that, in order to avoid that the ARP resolution time affects the measures, a simple *ping* was done prior to performing any PING test.

The script is written by the authors of this thesis. The script is shown below:

```bash
#!/bin/bash

#Use: ping_script ADDRESS TEST_TIME SIZE MEASURES SLEEP_TIME

address=$1
t_t=$2
size=$3
measures=$4
sleeptime=$5

rm /root/Tests/out_ping
rm /root/Tests/MAX_delay.txt
rm /root/Tests/MIN_delay.txt
rm /root/Tests/RTT.txt
rm /root/Tests/packet_loss.txt
rm /root/Tests/jitter.txt

for ((i=1; i <= $measures ; i++))
do
   ping -f $address -w $t_t -s $size > /root/Tests/out_ping
   awk -f /root/Tests/awk_average_RTT /root/Tests/out_ping
   awk -f /root/Tests/awk_loss /root/Tests/out_ping
   awk -f /root/Tests/awk_MAX_RTT /root/Tests/out_ping
   awk -f /root/Tests/awk_MIN_RTT /root/Tests/out_ping
   awk -f /root/Tests/awk_jitter /root/Tests/out_ping
   sleep $sleeptime
done
```

Listing 5.3: Ping script

As it can be seen, this script uses another scripts called *awk_average_RTT*, *awk_loss*, *awk_MAX_RTT*, *awk_MIN_RTT*, and *awk_jitter*; which were only used as parser for the *PING* output to gather the results in another files called, respectively, "*RTT.txt*", "*packet_loss.txt*", "*MAX_delay.txt*", "*MIN_delay.txt*", and "*jitter.txt*". All these scripts are written by the authors of this thesis and their code is shown, in order, in Appendices D.3, D.4, D.5, D.6, and D.7.

5.5.1.4 Read memory latency test

This test is done to check the velocity in the access to memory to read from it. In a blade, the application that runs onto it has to access to a huge amount of information in the memory, so the access velocity is a characteristic that is important for the good performance of the blade.

To perform this test, a set of tools and scripts called *lmbench* (available in Internet as free software) is necessary. The tool that is used within this set is the one called *lat_mem_rd*, which consists of a measurement of the memory read latency for varying memory sizes. With this tool, the entire memory hierarchy is measured, including onboard cache latency and size, external cache latency and size, main memory latency, and TLB miss latency.

The way of running this command is the following:

```
./lat_mem_rd  -P 1 192M 512
```

Where *-P 1*, *192M* and *512* are, respectively, the number of processes invoked to run the benchmark (1), the array size in megabytes (that is known to be much larger than the cache in order to ensure an accurate measurement) and the stride size in bytes (the skipped amount of memory before the next access). These values are fixed considering the recommendation of the supervisors of this Thesis project.

The output of this test is best examined in a graph where typically four plateaus appear. The plateaus that appear correspond to the onboard cache (if present), external cache (if present), main memory latency, and TLB miss latency.

5.5.1.5 CPU test

This test is done to check the CPU performance in the provider's machines. We need to reach a minimum CPU capacity if we want to run the Ericsson prototype MSC-S blades in the virtual machines offered by the provider.

To perform this test, a tool called *Dhrystone benchmark* (available in Internet as free software) is used. This tool provides a measure of integer performance (it does not use floating point instructions) – the performance we need to test because the MSC-S blade application only uses integer instructions. As it is said in [75], *Dhrystone benchmark* became the key standard benchmark from 1984, due to the growth of Unix systems.

In the beginning, this benchmark gave the performance ratings in terms of *Dhrystones per second*. Later, another value (VAX MIPS) was added by dividing *Dhrystones per second* by *1757* (the DEC VAX 11/780 result).

Nowadays, due to the PC evolution, we need another sort of measure to compare the CPU capacity in a computer. The way we have used to obtain a valid level of CPU performance is the following table, in which the only values represented are the values concerning the 64-bits machines running Ubuntu.

Table 5.1. *Dhrystone benchmark* results from Ubuntu GCC [75].

CPU	**Operating System**	**MHz**	**VAX MIPS**
Atom N455	64 bits Ubuntu	1 666	**2 704**
Athlon 64	64 bits Ubuntu	2 211	**6 873**
Core 2 Mob	64 bits Ubuntu	1 830	**8 241**
Phenom II	64 bits Ubuntu	3 000	**11 982**
Core 2 Duo	64 bits Ubuntu	2 400	**12 265**
Core i7 930	64 bits Ubuntu	****	**16 435**

5.5.2. Performance and results

The test programs presented above have been analyzed and we have several results. These results are described in the following subsections, along with figures that show the measured performance.

5.5.2.1 Inside the provider's network

This subsection brings together all the results obtained from the measurements done inside the provider's network. These results were obtained by using the resources available within the provider's network to communicate between the two virtual servers that were used during this bench-marking. These tests within the provider's network were designed to simulate the communication environment between two different blades in the same physical location.

As was stated in Section 5.1.2, one of the most important requirements concerning the network environment is the available bandwidth, hence the first tests measured the TCP and UDP bandwidth between these two machines. The requirement established at the beginning of this bench-marking for the network bandwidth was at least 50 Mbps. Figures 5.1 and 5.3 show the results obtained during a 24-hour test. As can be seen, the requirement is completely fulfilled by the provider's network, so we can be assured that the inter-blade communication is not going to be a limitation in the migration of the application to the Cloud.

Figure 5.1 shows the bandwidth measured during the performance of a TCP test. This test took place over a period of 24 hours, with 10-second measures, and an interval of five minutes between samples, obtaining a total of 288 samples. As it can be seen, the TCP bandwidth ranges between 115 and 120 Mbps, and the average TCP throughput measured is **116.55 Mbps**, which is more than sufficient to support the inter-blade communication requirements.

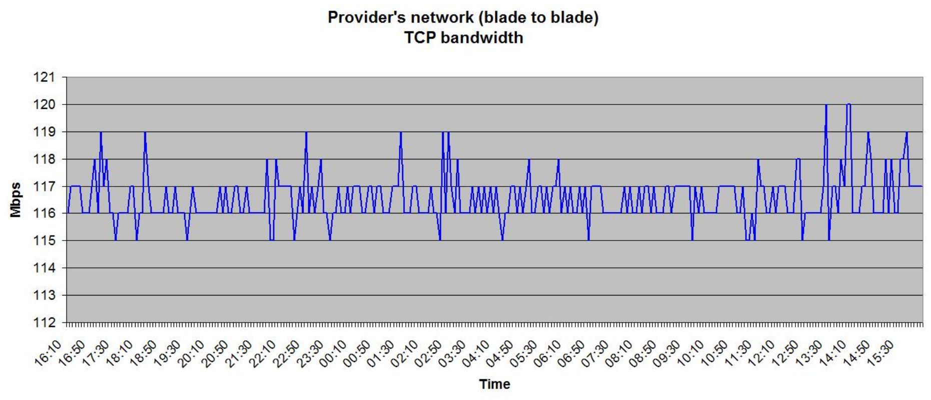


Figure 5.1. TCP bandwidth between blades inside the provider's network over a period of 24 hours.

On the other hand, Figure 5.2 shows an histogram with the frequency of the results obtained during the performance of the TCP test explained above. Together with this histogram, a probability density function of the TCP throughput is plotted. As it can be observed, the huge mayority of the measures are between 115 and 118 Mbps.

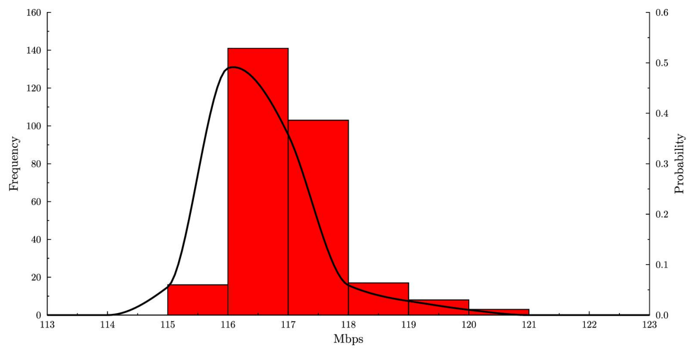


Figure 5.2. TCP throughput between blades inside the provider's network histogram and density function.

Figure 5.3 shows the bandwidth measured of the UDP throughput. This test was also conducted over a 24-hour period, with 10-second measures, and an interval of five minutes between samples, obtaining a total of 288 samples. As it can be seen, the UDP bandwidth ranges between 99 and 101 Mbps, and the average UDP throughput measured is **100.38 Mbps**, which is more than sufficient to support the inter-blade communication requirements.

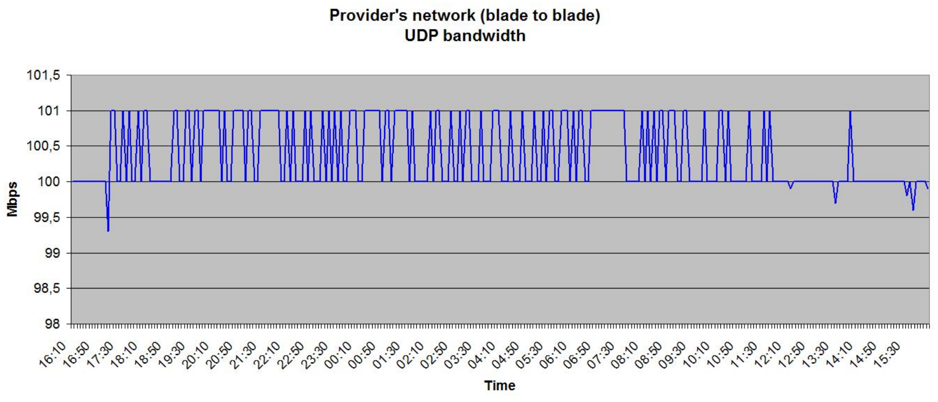


Figure 5.3. UDP bandwidth between blades inside the provider's network over a period of 24 hours.

Secondly, Figure 5.4 shows an histogram with the frequency of the results obtained during the performance of the UDP test explained above. Together with this histogram, a probability density function of the UDP throughput is plotted. As it can be observed, the huge mayority of the measures overcome 100 Mbps.

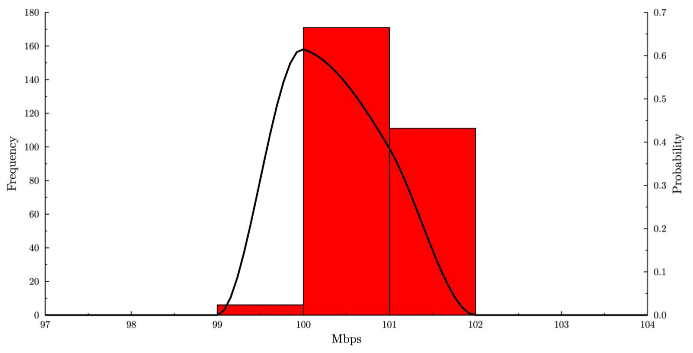


Figure 5.4. UDP throughput between blades inside the provider's network histogram and density function.

During the performance of the UDP test, two additional variables were measured. These variables are the observed jitter and the packet loss, as these are also important if the application is to be migrated to the Cloud.

Jitter is the deviation in the time between arrivals of packets that are sent at a fixed rate, caused by network congestion, timing drift, or route changes. In order to obtain a more representative sample of values, two 24-hour tests were performed and the results are shown in Figure 5.5 (each colour represents a 24-hour test). The jitter results observed in this test was lower than 0.2 ms and this value is considered a really good result.

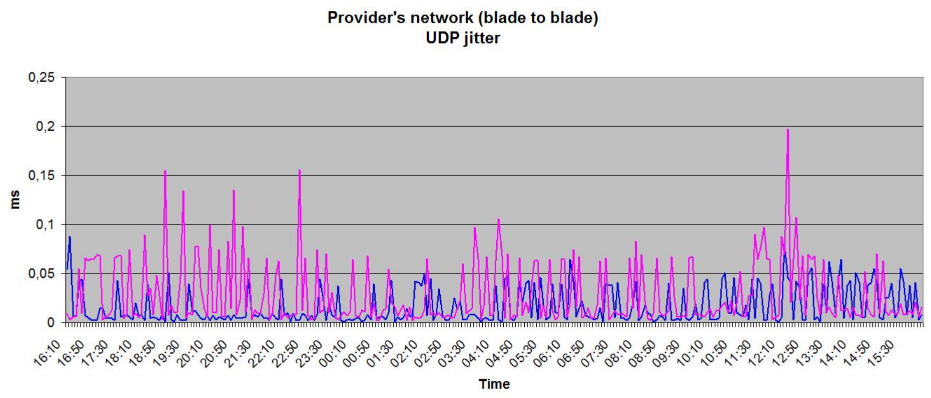


Figure 5.5. Jitter between blades inside the provider's network

Figure 5.6 shows a histogram in which it can be seen that the mayority of the measured jitters are in the range between 0 and 0.07 miliseconds. Taking this value into account we can assure that this amount of jitter is negligible during the prototype MSC-S blade performance.

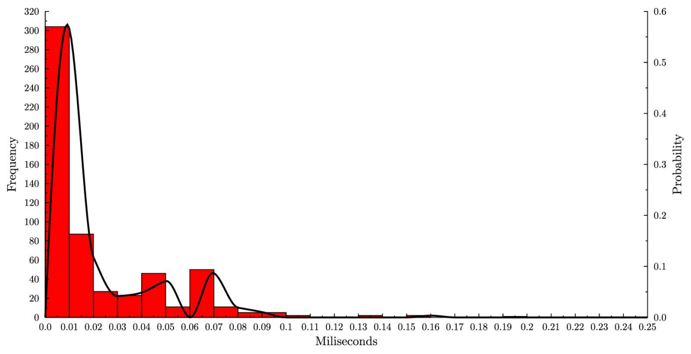


Figure 5.6. Jitter between blades inside the provider's network histogram and density function.

The packet loss represents the percentage of packets that are lost. Two 24-hour tests were carried out in order to obtain a more representative sample of values. As it can be observed in Figure 5.7, the packet loss value obtained from this test is normally lower than 0.6% with a maximum peak that reaches 1.2%. These values are considered good, because the application that should be run in the blades has mechanisms to retransmit packets that have been lost and it can tollerate this percentage of losses.

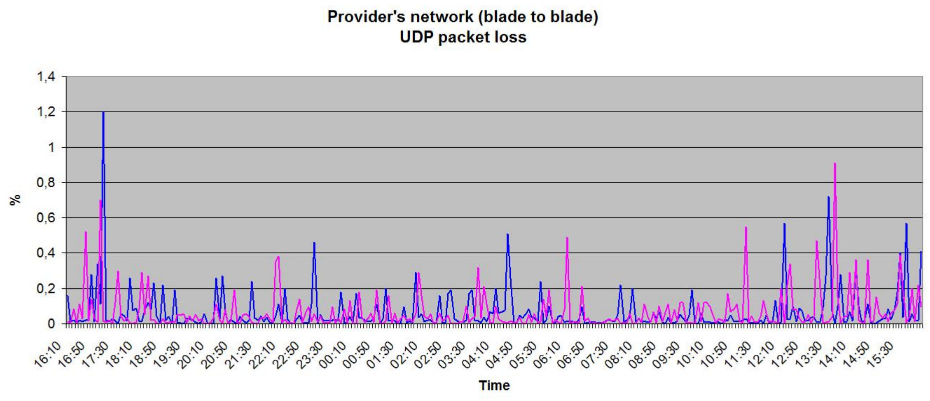


Figure 5.7. Packet loss between blades inside the provider's network

Figure 5.8 shows a histogram in which it can be seen that the percentage of packet loss is normally under 0.2%, a really good result. According to this value, we can assure that this amount of losses can be managed by the retransmit mechanisms available in the prototype MSC-S blades.

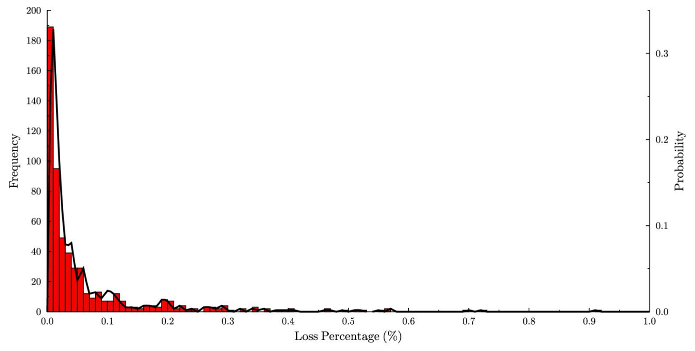


Figure 5.8. Packet loss between blades inside the provider's network histogram and density function.

Another important variable that has to be taken into account when the provider is tested is the round-trip time (RTT). The round-trip time is the needed for a packet to be sent to a destination plus the length of time needed by the sender to receive the acknowledgement coming from the receiver.

Figures 5.9, and 5.10 show, respectively, the average and maximum value for the RTT between the machines within the provider's network. The average RTT obtained is between 0.3 and 0.5 miliseconds, which is a extremely good value for communication between blades. Nevertheless the value that can limit the prototype MSC-S blade well-performance is the maximum RTT, in which case (an average of **22.96 ms** with a maximum peak of **64.88 ms**) it is perfectly affordable by the MSC-S blades, as can be seen in Section 4.2.2.1.

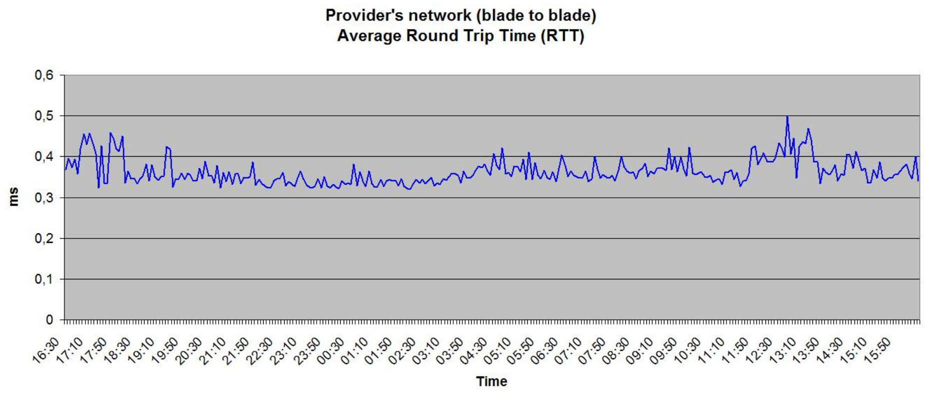


Figure 5.9. Average RTT between blades inside the provider's network

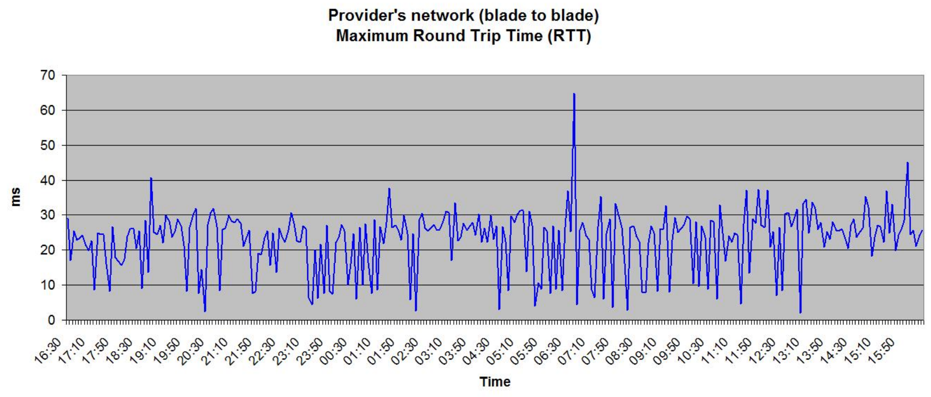


Figure 5.10. Maximum RTT between blades inside the provider's network

5.5.2.2 Between the provider and the *Isolated machine* in Stockholm

This subsection brings together all the results obtained from the measurements done between the provider's network and the *Isolated machine* located in Stockholm. The topology used to obtain these results had: the *Isolated machine* acting as the VPN server while the servers in the provider's network act as the VPN clients. These tests simulate the communication environment between two different blades in the same physical location (the servers in the provider) and the *Bridge machine* in Stockholm (the *Isolated machine*).

As was stated in Section 5.1.2, one of the most important requirements concerning the network environment is the bandwidth. The requirement established in the beginning of this bench-marking for the network bandwidth was at least 50 Mbps. Figures 5.11 and 5.12 show the results obtained during a 24-hour test, with 10-second measures, and an interval of five minutes between samples, obtaining a total of 288 samples. As it can be seen, the requirement was not met when UDP is used nor when TCP is used.

After careful checks and tests to discover the cause of this abnormal behaviour, the cause was finally found. Detailed measurements and several calls to the department in charge of maintaining Ericsson's internal network were needed in order to find out that there is a limitation in the upload throughput of Ericsson network. This limitation was established by the company in order to assure the network and it was not possible for the authors of this thesis to change this limitation.

As it can be seen in Figures 5.11 and 5.12, the 24-hour test shows a common behaviour concerning the obtained bandwidth with both TCP and UDP tests.

During the first three hours the bandwith was limited to 10 Mbps via TCP and to 6 Mbps via UDP while, during the rest of the test, the rate was limited to 14 Mbps in the case of TCP and to 12 Mbps in the case of UDP. Due to this limitation in throughput it is not possible to ensure that a blade can be run in the provider. Nevertheless, technical staff from Ericsson assured us that a modification in this limit could be made to set the limit at a higher value. Due to bureaucratic problems about confidentiality, security in the provider, and about permits to move the prototype application to the provider, and the long period of time required to deal with the procedures to ask for a change in the bandwidth, and due also to the limited duration of this project, the authors of this thesis decided to leave the deployment of a prototype MSC-S blade in the Cloud as future work.

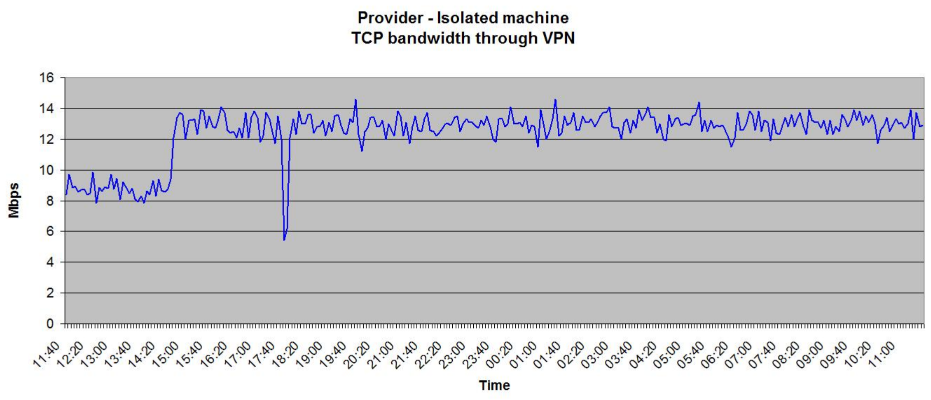


Figure 5.11. TCP bandwidth between the provider and the *Isolated machine* in Stockholm.

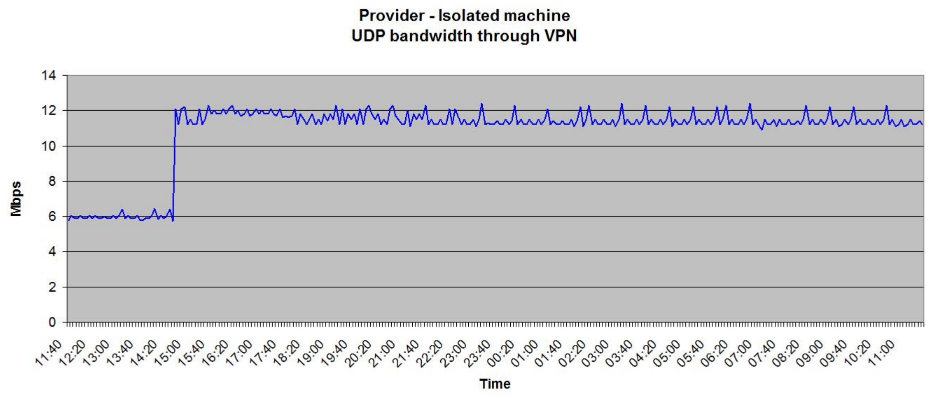


Figure 5.12. UDP bandwidth between the provider and the *Isolated machine* in Stockholm.

In Figures 5.13 and 5.14 two histograms can be seen. These histograms are made from the data obtained from TCP and UDP tests. In Figure 5.13, looking at the density function, can be observed that the bandwidth is more likely to be in the range between 12 and 14 Mbps, while in Figure 5.14 can be observed that, in this case, the bandwidth is more likely to be between 11 and 13 Mbps. Even so this amount of bandwidth is not enough for the right blade performance, as explained above.

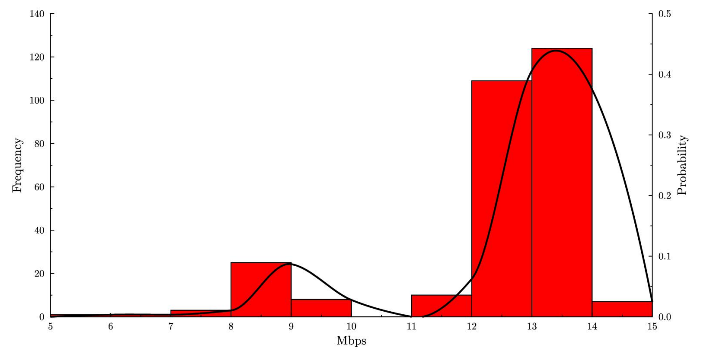


Figure 5.13. TCP bandwidth between the provider and the *Isolated machine* histogram and density function.

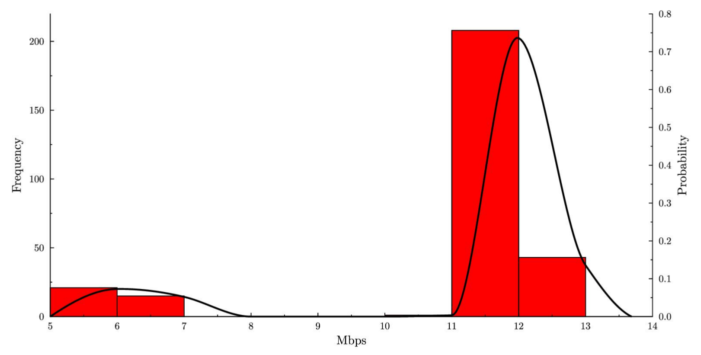


Figure 5.14. UDP bandwidth between the provider and the *Isolated machine* histogram and density function.

Due to the fact that the bandwidth from Ericsson to the provider is limited, the measurements of jitter and packet loss are not representative of what these parameters would be without the artificially limited throughput. That is why plots representing these values are not presented in this section. The tests are made to reach a given data rate so when this rate is not reached, the values regarding jitter and packet loss are not valid.

The RTT continues to be an important variable that has to be taken into account. In this case, the RTT represents the time needed for a request and a response between the provider and the *Isolated machine* (simulating, respectively, the blades and the *Bridge machine*). This value is primarily determined by the distance between the nodes, that is the distance between United Kingdom and Stockholm, and by the delay that comes from the routers along the path between the machines.

Figures 5.15, and 5.16 show, respectively, the average and maximum value for the RTT between the machines in the provider's network and the *Isolated machine*. The average RTT obtained is between 30.4 and 33.2 miliseconds, which is acceptable for the application that is running in the blades. Nevertheless the value that can limit the MSC-S blade well-performance is the maximum RTT, in which case (an average of **48.29 ms** with a maximum peak of **111.67 ms**) it is perfectly affordable by the MSC-S blades, as can be seen in Section 4.2.2.1.

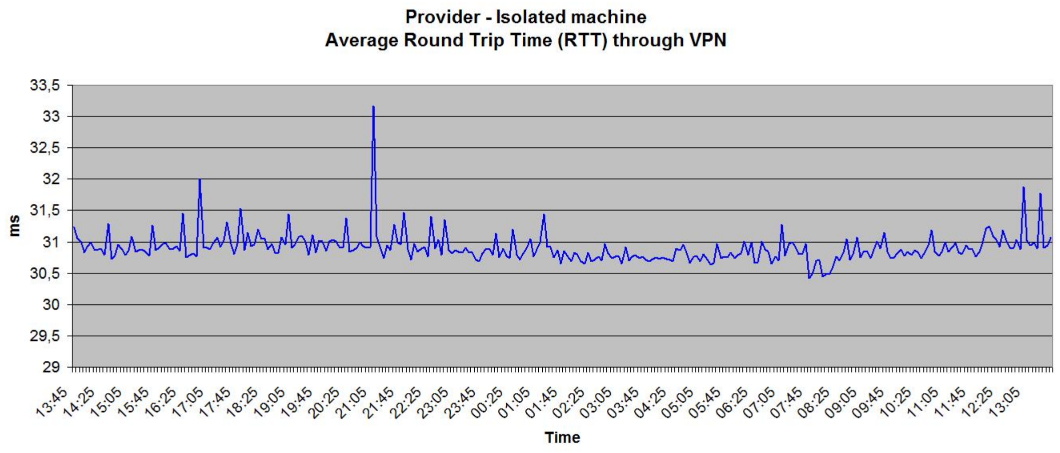


Figure 5.15. Average RTT between the provider and the *Isolated machine* in Stockholm.

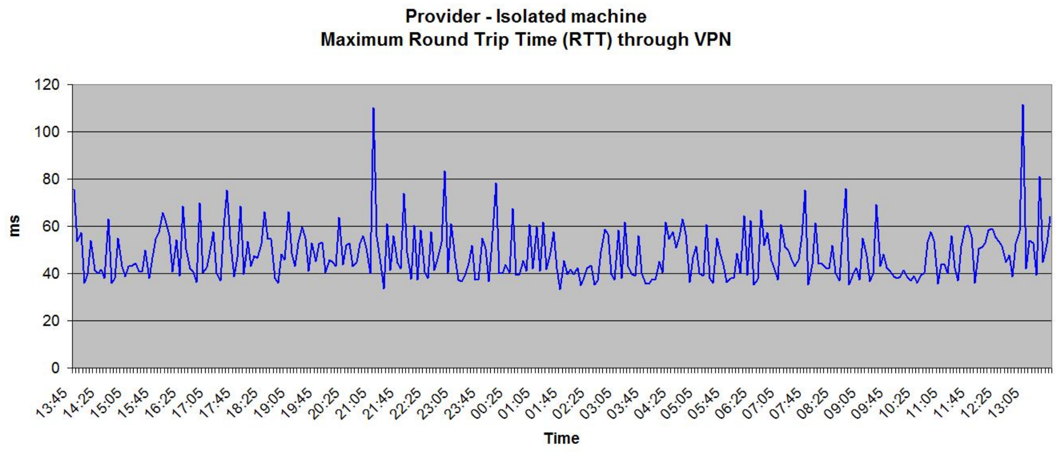


Figure 5.16. Maximum RTT between the provider and the *Isolated machine* in Stockholm.

5.5.2.3 Memory test in the provider

Another important test that must be done in the provider is a test that measures the memory read rate. It is important to know if the servers that the provider is offering are speedy enough to run the application (which is dominated by memory reading). Blades have to access a huge amount of information stored in the memory of the host machine, so a test called *lat_mem_rd* included in the *lmbench* set of tests was used. The characteristics of this test are described in Section 5.5.1.

In order to obtain a representative value of the memory read latency, the test is performed once every ten minutes over a period of 24 hours, obtaining 144 samples of each reading size, with 119 different values of reading sizes. After calculating the average value from all these samples, these values were plotted in Figure 5.17 using a logarithmic scale.

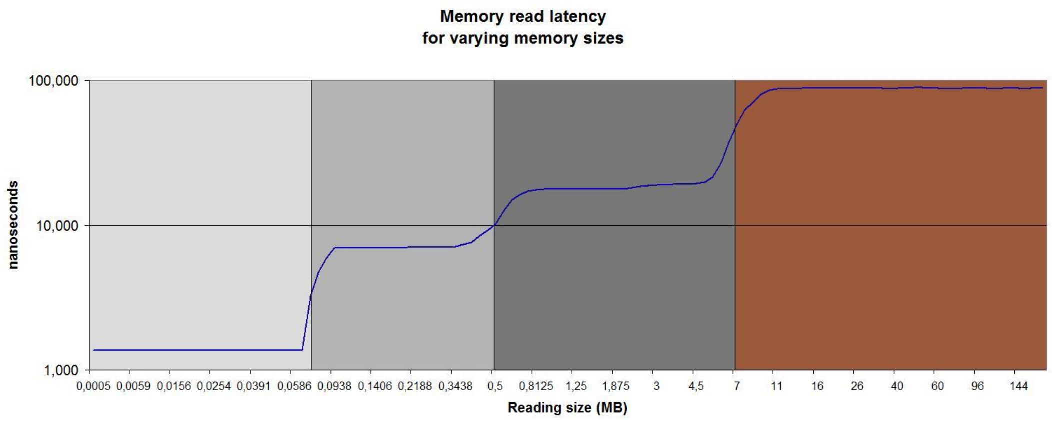


Figure 5.17. Memory read latency for varying memory sizes (from 0 to 192 MB)

The conclusions that can be obtained from Figure 5.17 and from the information available in [76][77] (where it is explained how to interpret the output) are the following:

- There are four plateaus that correspond to:
  - *a*) Onboard cache
  - *b*) External cache
  - *c*) Main memory
  - *d*) Translation Lookaside Buffer (TLB) miss
- The average Onboard Cache read latency is 1.366 ns
- The average External Cache read latency is 7.061 ns
- The average Main Memory read latency is 18.281 ns
- The average TLB miss latency is 88.060 ns

In conclusion, the results obtained from these tests indicate that the performance of the blades in the servers offered by the Cloud provider are comparable to the blades currently used in the traditional local rack solution.

5.5.2.4 CPU test in the provider

Since the communication between servers within the provider, the communication between the provider and the *Isolated machine*, and the memory access were determined to be adequate, the next and final step is a CPU test. This test is designed to characterize the performance of the CPU in the VM.

This CPU test is carried out using the program *Dhrystone Benchmark*, which is described in Section 5.5.1. This test has been run ten times in order to verify that the performance is uniform, producing the results shown in table 5.2.

Table 5.2. Obtained results running *Dhrystone Benchmark*

Test	Microseconds for one run through Dhrystone	Dhrystones per second	VAX MIPS rating
#1	0.07	15 103 406	8 596.13
#2	0.07	15 256 517	8 683.28
#3	0.07	15 291 209	8 703.02
#4	0.07	15 265 981	8 688.66
#5	0.07	15 185 110	8 642.64
#6	0.07	15 339 153	8 730.31
#7	0.07	15 341 271	8 731.51
#8	0.07	15 303 415	8 709.97
#9	0.07	15 358 888	8 741.54
#10	0.07	15 323 611	8 721.46
Average	**0.07**	**15 276 856.1**	**8 694.852**

Taking into account table 5.1, the results and comments taken from [75], and looking at the average VAX MIPS rating value in table 5.2, the performed test gives the virtual machine the same CPU capacity of a computer in the range between a *Core 2 Mob* and a *Phenom II*. According to our employers, this CPU capacity is enough for running a prototype MSC-S blade on these VMs.

The performed test gives also valuable information about the physical machine on which the virtual machine is running. The information obtained from the provider about the physical host machine is the following:

- CPU AuthenticAMD, Features Code 178BFBFF, Model Code 00100F42
- Quad-Core AMD Opteron™Processor 2374 HE
- Measured - Minimum 2200 MHz, Maximum 2200 MHz

This test uses also some Linux functions to access to the */proc/cpuinfo* folder, detecting the configuration of the current machine that is running, that is the virtual machine. The information obtained is the following:

- CPUs 4, Configured CPUs 4
- RAM Size 15.08 GB, Page Size 4096 Bytes
- Linux, Ubuntu-2, 2.6.32-31-server

Eventually, considering all the information gathered about the CPU performance in this section, we conclude that the virtual machine offered by the provider is perfectly suitable for running a Ericsson prototype MSC-S blade in terms of CPU performance.

Chapter 6

Conclusions and future work

In this chapter, a discussion of all the results obtained in this master's thesis project is presented. Then, the future work suggested by the authors of this master thesis project in order to continue the investigation is compiled.

6.1. Conclusions

In this master's thesis project the utilization of Cloud technology in order to implement telecommunication services has been proposed. In addition, a Cloud implementation of a specific MSC-S application, the Ericsson MSC-S BC, has been tested in a laboratory environment in order to evaluate the drawbacks that such a Cloud deployment could have from a network perspective. Then, a cloud service provider survey has been conducted, and a cloud service provider has been selected, based on specific application requirements, to be bench-marked. This section presents a discussion of all the laboratory tests results of the Ericsson MSC-S BC cloud solution. Then, based on the Cloud providers' survey and the subsequent bench-marking process of the selected Cloud provider, an evaluation of the Ericsson MSC-S BC cloud solution's feasability in a real Cloud environment is provided.

6.1.1. Laboratory tests

A network environment was designed and deployed by the authors of this thesis project in order to provide a Cloud solution for the Ericsson MSC-S BC. This Cloud solution consisted of an implemention of the prototype Ericsson MSC-S blades in a Cloud environment, while the rest of the Ericsson MSC-S BC system was located on-premises. The Cloud solution was tested in a Cloud environment with additional network impairments simulated in the laboratory. These tests are compiled in Chapter 4.

The results of the experiments have demonstrated that, from a network perspective, and after several modifications made to the default configuration parameters of the system (see Section 4.2.5), the primary Ericsson MSC-S BC functionalities **are** compatible with the Cloud solution proposed in this master's thesis project. The first functionality tested was the MSC-S application, that is run in a distributed manner over the MSC-S blades that form the Ericsson MSC-S BC. The test results prove that the MSC-S application serves the mobile signalling traffic correctly in every Cloud solution topology tested with external prototype MSC-S blades located in the lab-simulated Cloud environment (hybrid, external, and geographically distributed cluster topologies). However, this could only be tested in a low call rate scenario due to an inconsistency found in the RAN traffic generator during the tests.

The Ericsson MSC-S BC scalability function tests were also successful from a networking point of view. Only a prototyping issue alien to the network configuration or network impairments utilized in the Cloud solution arose throughout the tests. This issue is directly associated with the software implementation of the prototype MSC-S blade and prevents a fully functional operation of the cluster scalability feature for these specific blades. However, solving this challenge was out of the scope of this thesis work.

Finally, the fault recovery mechanisms have been verfied for the Ericsson MSC-S Cloud solution implementation. As with the scalability function, a prototyping issue hinders a fully functional operation of the fault recovery mechanisms triggered for the temporary faults (for both single blade link and partition faults). However, it has been demonstrated that not the network configuration utilized nor the network impairments introduced in the system in order to simulate the Cloud environment affect the fault recovery mechanisms' operation.

6.1.2. Cloud Service provider

As was shown in Chapter 5, after a thorough survey, *Rackspace* was the Cloud provider selected to be bench-marked. This decision was taken based on specific computing performance and network functionalities criteria imposed by the application to which the tests were going to be targeted: the Ericsson MSC-S BC. *Rackspace* met all those criteria, and was accepted by our industrial supervisors.

The bench-marking process conducted in the *Rackspace* contracted machines was successful in every area. Measurements of the computing performance and memory read latency, which were the two network unrelated conditions imposed by the application, showed that a prototype MSC-S blade can be executed running on the Cloud virtual machines. The results of the network bench-marking process indicate that *Rackspace*'s network supports all the functions that are needed by the Ericsson MSC-S BC system: VLAN and multicast.

Additionally, the bandwidth within the Cloud provider network was shown to be sufficient to deploy the prototype MSC-S blades in the *Rackspace* Cloud environment. However, the bandwidth from the Cloud virtual machines to the Internet could not be tested correctly due to Ericsson's network throughput limitation. Nevertheless, *Rackspace* representatives assured us that the outbound Internet bandwidth is limited to 200 Mbps and that the inbound Internet bandwidth has no limitations. Therefore, based upon these claims, we believe that the Cloud provider exceeds the initial requirements in terms of bandwidth from the Cloud virtual machines to the Internet. Despite this assumption, in order to be capable of making any definitive valuable judgement regarding communication matters in the Cloud provider, extensive benchmarking needed to be carried out.

Although it was not possible to run the Ericsson prototype MSC-S blades in *Rackspace's* Cloud machines due to confidentiality and privacy issues (see Section 1.7), a great overview of the performance of the offered servers has been obtained through our measurements and analysis. All these results, in the absence of an Internet bandwidth bench-marking, demonstrate that from a network persperctive the prototype MSC-S blades that compose any of the topologies mentioned in Section 3.1 could be implemented in *Rackspace* Cloud machines, or in any other Cloud computing provider's with the same characteristics as *Rackspace*.

6.2. Future work

This section analyses the limitations of this project as well as the issues encountered during the investigation, and formulates future work suggestions that would extend the work carried out in this master's thesis project.

As it has been mentioned several times in this report, the main purpose of this project has been the design, implementation, and evaluate a Cloud solution for the Ericsson MSC-S BC application, stressing the network aspects of the proposed solution. In this process, some issues directly related to the prototype Ericsson MSC-S blades affected the correct operation of some functions of the Ericsson MSC-S BC. The prototype MSC-S application needs to be debugged in order to remove those inconsistencies and provide a completely functional application that is compatible with standard hardware platforms and virtualized environments.

The introduction of high delays between the RAN traffic generator and the external prototype MSC-S blades, together with a high traffic load, triggered an increase of in-progress calls that the RAN traffic generator was unable to manage. Therefore, the RAN traffic generator needs to be modified to successfully generate higher loads.

The traffic scenario used for the Cloud solution laboratory tests was limited due to the RAN traffic generator's drawback mentioned above. Consequently, further testing needs to be conducted in order to prove the correct operation of the Cloud solution in a highly loaded traffic scenario.

As stated in Section 1.7, the Ericsson MSC-S BC Cloud solution tested in this thesis project could not be tested in a real Cloud environment due to privacy and confidentiality issues. Once allowed by Ericsson, the Ericsson MSC-S BC Cloud solution should be tested in an appropriate real Cloud environment such as that offered by *Rackspace*.

This thesis project proposes the deployment of telecommunication services in the Cloud. Nonetheless, the legal aspects of this sort of solution need to be analyzed, considering the telecommunication services' QoS standards that are required by the relevant authorities. In addition, an analysis of Cloud security would need to be conducted given the sensitive information that is handled by telecommunication services. Future work will also need to consider the aspects of lawful interception, especially those involved when doing so in the Cloud.

Finally, this master's thesis project has focused on a Cloud solution for a specific telecommunication services' application: the Ericsson MSC-S BC. However, we suggest an extension of this work so that other telecommunication services' applications could take advantage of the benefits the Cloud provides.

Bibliography

[1] Kurt Tutschku. Keynote: Mobile Service Evolution und Research Trends in Future Internet. In *Computer Software and Applications Conference Workshops (COMPSACW), 2011 IEEE 35th Annual*, pages 46 –47, July 2011.

[2] C.M. Machuca. Expenditures Study for Network Operators. In *Transparent Optical Networks, 2006 International Conference on*, volume 1, pages 18 –22, June 2006.

[3] J.D. Chimeh. Mobile services: Trends and evolution. In *Advanced Communication Technology, 2009. ICACT 2009. 11th International Conference on*, volume 02, pages 946 –948, February 2009.

[4] Marius Corici, Dragos Vingarzan, Thomas Magedanz, and Thomas Magedanz. 3GPP Evolved Packet Core - the Mass Wireless Broadband all-IP architecture. *Telecommunications: The Infrastructure for the 21st Century (WTC), 2010*, pages 1 –6, September 2010.

[5] Ericsson AB. Voice and Video calling over LTE, February 2012. Available from: http://www.ericsson.com/res/docs/whitepapers/WP-Voice-Video-Calling-LTE.pdf. [Accessed 20 March 2012].

[6] Srini, Rao. Mobile Broadband Evolution - LTE and EPC, April 2010. Available from: http://ieee-boston.org/presentations_society/lte_epc_ieee_comsoc_rao_april_8_2010.pdf. [Accessed 25 November 2011].

[7] Ericsson AB. Introduction to IMS, March 2007. Available from: http://www.facweb.iitkgp.ernet.in/~pallab/mob_com/Ericsson_Intro_to_IMS.pdf. [Accessed 22 November 2011].

[8] Korinthios Georgios. Mobile Services Network Technology Evolution and the role of IMS. Available from: http://www.ict-fireworks.eu/fileadmin/events/FIREweek/2nd-WS-Converged-Networks/03-Georgios_Korinthios.pdf. [Accessed 23 November 2011].

[9] G. Camarillo and M.A. García-Martín. *The 3G IP Multimedia Subsystem (IMS): Merging the Internet and the Cellular Worlds.* John Wiley & Sons, 2007.

[10] Cisco and/or its affilliates. Cisco Visual Networking Index: Global Mobile Data Traffic Forecast Update, 2010-2015, February 2011. Available from: http://www.cisco.com/en/US/solutions/collateral/ns341/ns525/ns537/ns705/ns827/white_paper_c11-520862.pdf. [Accessed 29 October 2011].

[11] Peter Mell and Timothy Grance. The NIST Definition of Cloud Computing, September 2011. Available from: http://csrc.nist.gov/publications/nistpubs/800-145/SP800-145.pdf. [Accessed 21 March 2012].

[12] S. Dhar. From outsourcing to Cloud computing: Evolution of IT services. In *Technology Management Conference (ITMC), 2011 IEEE International*, pages 434 –438, June 2011.

[13] G. Caryer, T. Rings, J. Gallop, S. Schulz, J. Grabowski, I. Stokes-Rees, and T. Kovacikova. Grid/cloud computing interoperability, standardization and the Next Generation Network (NGN). In *Intelligence in Next Generation Networks, 2009. ICIN 2009. 13th International Conference on*, pages 1 –6, October 2009.

[14] Adetola Oredope and Antonio Liotta. Plugging 3G Mobile Networks into the Internet: A Prototype-Based Evaluation. In *Computing, 2006. CIC '06. 15th International Conference on*, pages 406 –411, November 2006.

[15] T. Rings, J. Grabowski, and S. Schulz. On the Standardization of a Testing Framework for Application Deployment on Grid and Cloud Infrastructures. In *Advances in System Testing and Validation Lifecycle (VALID), 2010 Second International Conference on*, pages 99 –107, August 2010.

[16] D. Meyer and G. Zobrist. TCP/IP versus OSI. *Potentials, IEEE*, 9(1):16 –19, February 1990.

[17] D.M. Bortoluzzi, E.S. Cunha, and E.S. Specialski. An extended model for TCP/IP architecture. In *Communications Systems, 2004. ICCS 2004. The Ninth International Conference on*, pages 351 –355, September 2004.

[18] James F. Kurose and Keith W. Ross. *Computer Networking: A Top-Down Approach*. Addison-Wesley Publishing Company, USA, 5th edition, 2009.

[19] S. Zaman and F. Karray. TCP/IP Model and Intrusion Detection Systems. In *Advanced Information Networking and Applications Workshops, 2009. WAINA '09. International Conference on*, pages 90 –96, May 2009.

[20] IEEE 802.3 Standards, 2010. Available from: http://standards.ieee.org/about/get/802/802.3.html. [Accessed 25 October 2011].

[21] D. Plummer. Ethernet Address Resolution Protocol: Or Converting Network Protocol Addresses to 48.bit Ethernet Address for Transmission on Ethernet Hardware. RFC 826 (Standard), 1982. Updated by RFCs 5227, 5494. Available from: http://www.ietf.org/rfc/rfc826.txt. [Accessed 25 October 2011].

[22] IEEE. 802.1Q - Virtual LANs, 2006. Available from: http://www.ieee802.org/1/pages/802.1Q.html. [Accessed 21 March 2012].

[23] SysKonnect GmbH. White Paper - Virtual Networks, 2001. Available from: http://staff.pccu.edu.tw/~lchou/reference/network/vlan.pdf. [Accessed 25 October 2011].

[24] Micrel Inc. Virtual LAN. Applications and Technology. A White Paper, 2004. Available from: http://www.fourdtech.com/downloads/virtual_lan.pdf. [Accessed 25 October 2011].

[25] D. McPherson and B. Dykes. VLAN Aggregation for Efficient IP Address Allocation. RFC 3069 (Informational), 2001. Available from: http://www.ietf.org/rfc/rfc3069.txt. [Accessed 25 October 2011].

[26] J. Postel. Internet Protocol. RFC 791 (Standard), 1981. Updated by RFC 1349. Available from: http://www.ietf.org/rfc/rfc791.txt. [Accessed 25 October 2011].

[27] Sami Iren, Paul D. Amer, and Phillip T. Conrad. The transport layer: tutorial and survey. *ACM Comput. Surv.*, 31:360–404, December 1999.

[28] Miao Xue, Fei Song, Feng Qiu, Ming Wan, Sidong Zhang, and Hongke Zhang. Redesigning Transport Layer Architecture for Future Internet. In *Wireless Communications, Networking and Mobile Computing (WiCOM), 2011 7th International Conference on*, pages 1 –5, September 2011.

[29] J. Postel. Transmission Control Protocol. RFC 793 (Standard), 1981. Updated by RFCs 1122, 3168, 6093. Available from: http://www.ietf.org/rfc/rfc793.txt. [Accessed 25 October 2011].

[30] J. Postel. User Datagram Protocol. RFC 768 (Standard), 1980. Available from: http://www.ietf.org/rfc/rfc768.txt. [Accessed 25 October 2011].

[31] R. Stewart. Stream Control Transmission Protocol. RFC 4960 (Proposed Standard), 2007. Updated by RFCs 6096, 6335. Available from: http://www.ietf.org/rfc/rfc4960.txt. [Accessed 25 October 2011].

[32] R. Stewart and C. Metz. SCTP: new transport protocol for TCP/IP. *Internet Computing, IEEE*, 5(6):64 –69, nov/dec 2001.

[33] P. Natarajan, F. Baker, P.D. Amer, and J.T. Leighton. SCTP: What, Why, and How. *Internet Computing, IEEE*, 13(5):81 –85, sept.-oct. 2009.

[34] E.P. Rathgeb, C. Hohendorf, and M. Nordhoff. On the Robustness of SCTP against DoS Attacks. In *Convergence and Hybrid Information Technology, 2008. ICCIT '08. Third International Conference on*, volume 2, pages 1144 –1149, nov. 2008.

[35] K.-J. Grinnemo, T. Andersson, and A. Brunstrom. Performance Benefits of Avoiding Head-of-Line Blocking in SCTP. In *Autonomic and Autonomous Systems and International Conference on Networking and Services, 2005. ICAS-ICNS 2005. Joint International Conference on*, page 44, oct. 2005.

[36] S.E. Deering. Host extensions for IP multicasting. RFC 1112 (Standard), 1989. Updated by RFC 2236. Available from: `http://www.ietf.org/rfc/rfc1112.txt`. [Accessed 25 October 2011].

[37] B. Fenner, H. He, B. Haberman, and H. Sandick. Internet Group Management Protocol (IGMP) / Multicast Listener Discovery (MLD)-Based Multicast Forwarding ("IGMP/MLD Proxying"). RFC 4605 (Proposed Standard), 2006. Available from: `http://www.ietf.org/rfc/rfc4605.txt`. [Accessed 25 October 2011].

[38] B. Fenner, M. Handley, H. Holbrook, and I. Kouvelas. Protocol Independent Multicast - Sparse Mode (PIM-SM): Protocol Specification (Revised). RFC 4601 (Proposed Standard), 2006. Updated by RFCs 5059, 5796, 6226. Available from: `http://www.ietf.org/rfc/rfc4601.txt`. [Accessed 25 October 2011].

[39] W. Simpson. IP in IP Tunneling. RFC 1853 (Informational), 1995. Available from: `http://www.ietf.org/rfc/rfc1853.txt`. [Accessed 25 October 2011].

[40] R. Housley and S. Hollenbeck. EtherIP: Tunneling Ethernet Frames in IP Datagrams. RFC 3378 (Informational), 2002. Available from: `http://www.ietf.org/rfc/rfc3378.txt`. [Accessed 25 October 2011].

[41] I. Bojanova and A. Samba. Analysis of Cloud Computing Delivery Architecture Models. In *Advanced Information Networking and Applications (WAINA), 2011 IEEE Workshops of International Conference on*, pages 453 –458, March 2011.

[42] Chunye Gong, Jie Liu, Qiang Zhang, Haitao Chen, and Zhenghu Gong. The Characteristics of Cloud Computing. In *Parallel Processing Workshops (ICPPW), 2010 39th International Conference on*, pages 275 –279, September 2010.

[43] Victor Delgado. Exploring the limits of cloud computing, Master Thesis, KTH Royal Institute of Technology, Stockholm, Sweden, TRITA-ICT-EX-2010:277. November 2010. Available from: http://web.it.kth.se/~maguire/DEGREE-PROJECT-REPORTS/101118-Victor_Delgado-with-cover.pdf. [Accessed 29 November 2011].

[44] Red Hat. Red Hat Cloud Foundations: Cloud 101, 2010. Available from: http://www.redhat.com/f/pdf/cloud/101_whitepaper.pdf. [Accessed 27 October 2011].

[45] The Rackspace Cloud - Login. Available from: https://lon.manage.rackspacecloud.com/pages/Login.jsp;jsessionid=F473FF163BA62F37E84B9BF03B5F42A1.manage-n01. [Accessed 24 November 2011].

[46] Eucalyptus Cloud Computing Software, 2011. Available from: http://www.eucalyptus.com/. [Accessed 27 October 2011].

[47] OpenNebula: The Open Source Toolkit for Cloud Computing, 2011. Available from: http://opennebula.org/. [Accessed 27 October 2011].

[48] B.R. Kandukuri, V.R. Paturi, and A. Rakshit. Cloud security issues. In *Services Computing, 2009. SCC '09. IEEE International Conference on*, pages 517 –520, sept. 2009.

[49] M. Alhamad, T. Dillon, and E. Chang. Sla-based trust model for cloud computing. In *Network-Based Information Systems (NBiS), 2010 13th International Conference on*, pages 321 –324, sept. 2010.

[50] Minqi Zhou, Rong Zhang, Wei Xie, Weining Qian, and Aoying Zhou. Security and privacy in cloud computing: A survey. In *Semantics Knowledge and Grid (SKG), 2010 Sixth International Conference on*, pages 105 –112, nov. 2010.

[51] F. Sabahi. Cloud computing security threats and responses. In *Communication Software and Networks (ICCSN), 2011 IEEE 3rd International Conference on*, pages 245 –249, may 2011.

[52] W.A. Jansen. Cloud hooks: Security and privacy issues in cloud computing. In *System Sciences (HICSS), 2011 44th Hawaii International Conference on*, pages 1 –10, jan. 2011.

[53] S. Pearson and A. Benameur. Privacy, security and trust issues arising from cloud computing. In *Cloud Computing Technology and Science (CloudCom), 2010 IEEE Second International Conference on*, pages 693 –702, 30 2010-dec. 3 2010.

[54] Cisco Systems, Inc. Cloud: Powered by the network, 2010. Available from: http://www.cisco.com/en/US/solutions/collateral/ns341/ns991/white_paper_c11-609220.pdf. [Accessed 28 October 2011].

[55] Kate Craig-Wood. IaaS vs. PaaS vs. SaaS Definition, May 2010. Available from: http://www.katescomment.com/iaas-paas-saas-definition/. [Accessed 28 October 2011].

[56] ETSI World Class Standards, 2011. Available from: http://www.etsi.org/WebSite/homepage.aspx. [Accessed 23 November 2011].

[57] Nokia Networks Oy. GSM Architecture, January 2002. Available from: http://www.roggeweck.net/uploads/media/Student_-_GSM_Architecture.pdf. [Accessed 19 November 2011].

[58] 3GPP A Global Initiative, 2011. Available from: http://www.3gpp.org/. [Accessed 23 November 2011].

[59] J.M. Ballot. The IP Road to Mobile Network Evolution, 2007. Available from: http://www.alcatel-lucent.com/enrich/v1i12007/pdf/The_IP_Road_en.pdf. [Accessed 18 November 2011].

[60] Cisco Systems, Inc. GPRS White Paper, 2000. Available from: http://www.cisco.com/warp/public/cc/so/neso/gprs/gprs_wp.pdf. [Accessed 22 November 2011].

[61] Nokia Telecommunications Oy. Enhanced Data Rates for GSM Evolution EDGE, 1999. Available from: http://www.privateline.com/Cellbasics/Nokiaedge_wp.pdf. [Accessed 23 November 2011].

[62] Ericsson Radio Systems AB. Basic Concepts of WCDMA Radio Access Network, 2001. Available from: http://www.cs.ucsb.edu/~almeroth/classes/W03.595N/papers/wcdma-concepts.pdf. [Accessed 18 November 2011].

[63] Ericsson AB. Basic Concepts of HSPA, February 2007. Available from: http://netlab.ulusofona.pt/cr/HSPA-Concepts.pdf. [Accessed 22 November 2011].

[64] Ericsson AB and Juniper Networks. Beyond Best Effort: Telecom-Quality IP Infrastructure for Mobile Softswitching, 2006. Available from: http://s-tools1.juniper.net/solutions/literature/solutionbriefs/beyond_best_effort.pdf. [Accessed 23 November 2011].

[65] Motorola, Inc. Long Term Evolution (LTE): A Technical Overview, 2007. Available from: http://www.motorola.com/web/Business/Solutions/Industry%20Solutions/Service%20Providers/Wireless%20Operators/LTE/_Document/Static%20Files/6834_MotDoc_New.pdf. [Accessed 24 November 2011].

[66] Petri Maekiniemi and Jan Scheurich. Ericsson MSC Server Blade Cluster, March 2008. Available from: http://www.ericsson.com/res/thecompany/docs/publications/ericsson_review/2008/Blade.pdf. [Accessed 25 October 2011].

[67] VMware, Inc. VMware Virtualization Software for Desktops, Servers & Virtual Machines, 2012. Available from: http://www.vmware.com/. [Accessed 16 January 2012].

[68] Main Page - KVM, 2012. Available from: http://www.linux-kvm.org/page/Main_Page. [Accessed 28 January 2012].

[69] Fabrice Bellard. QEMU, December 2011. Available from: http://wiki.qemu.org/Main_Page. [Accessed 3 January 2012].

[70] EtherIP for Linux. Available from: http://www.8bytes.org/wiki/EtherIP_for_Linux. [Accessed 11 January 2012].

[71] OpenVPN Technologies, Inc. OpenVPN - Open source VPN, 2012. Available from: http://openvpn.net/. [Accessed 11 January 2012].

[72] Ubuntu. OpenVPN. Available from: https://help.ubuntu.com/10.04/serverguide/C/openvpn.html. [Accessed 11 January 2012].

[73] OpenVPN Technologies, Inc. Ethernet Bridging, 2012. Available from: http://openvpn.net/index.php/open-source/documentation/miscellaneous/76-ethernet-bridging.html#linuxscript. [Accessed 11 January 2012].

[74] IPERF - The Easy Tutorial, December 2010. Available from: http://openmaniak.com/iperf.php. [Accessed 6 February 2012].

[75] Roy Longbottom. Dhrystone Benchmark Results - Roy Longbottom's PC benchmark Collection, March 2011. Available from: http://www.roylongbottom.org.uk/dhrystone%20results.htm#anchorLinux. [Accessed 3 February 2012].

[76] Joshua Ruggiero. Measuring Cache and Memory Latency and CPU to Memory Bandwidth, December 2008. Available from: ftp://download.intel.com/design/intarch/PAPERS/321074.pdf. [Accessed 9 February 2012].

[77] LAT_MEM_RD(8) - LMBENCH man page. Available from: http://www.bitmover.com/lmbench/lat_mem_rd.8.html. [Accessed 2 February 2012].

[78] Luke Galea. KQEMU - KDE GUI For QEMU, 2005. Available from: http://kqemu.sourceforge.net/. [Accessed 3 February 2012].

[79] Linux Foundation. Netem | The Linux Foundation, November 2009. Available from: http://www.linuxfoundation.org/collaborate/workgroups/networking/netem. [Accessed 29 January 2012].

[80] Cloud Servers, Virtual Servers: GoGrid Cloud Hosting. Available from: http://www.gogrid.com/cloud-hosting/cloud-servers.php. [Accessed 2 November 2011].

[81] Cloud & Hybrid Hosting Pricing: GoGrid Cloud Hosting. Available from: http://www.gogrid.com/cloud-hosting/cloud-hosting-pricing.php. [Accessed 2 November 2011].

[82] Dedicated Servers, Physical Servers: GoGrid Cloud Hosting. Available from: http://www.gogrid.com/cloud-hosting/dedicated-servers.php. [Accessed 2 November 2011].

[83] Service Level Agreement (SLA): GoGrid Cloud Hosting, June 2010. Available from: http://www.gogrid.com/legal/sla.php. [Accessed 28 October 2011].

[84] Terms of service: GoGrid Cloud Hosting, June 2011. Available from: http://www.gogrid.com/legal/terms-service.php. [Accessed 28 October 2011].

[85] Acceptable Use Policy (AUP): GoGrid Cloud Hosting, October 2008. Available from: http://www.gogrid.com/legal/aup.php. [Accessed 28 October 2011].

[86] Privacy Policy: GoGrid Cloud Hosting, October 2008. Available from: http://www.gogrid.com/legal/sla.php. [Accessed 28 October 2011].

[87] EU Safe Harbor Policy: GoGrid Cloud Hosting. Last updated: November 1st, 2004. Available from: http://www.gogrid.com/legal/EU-safe-harbor.php. [Accessed 28 October 2011].

[88] Cloud Server Costs, Cloud Hosting Prices | Rackspace Cloud. Available from: http://www.rackspace.co.uk/cloud-hosting/cloud-products/cloud-servers/prices/. [Accessed 2 November 2011].

[89] Rackspace - Cloud Files SLA, October 2010. Available from: http://www.rackspace.co.uk/legal/cloud-sla-files/. [Accessed 28 October 2011].

[90] Rackspace - Managed Hosting Terms, April 2011. Available from: http://www.rackspace.co.uk/legal/managed-hosting-terms/. [Accessed 28 October 2011].

[91] General Terms | Rackspace Hosting, August 2011. Available from: http://www.rackspace.co.uk/legal/general-terms-rackspace-hosting/. [Accessed 28 October 2011].

[92] Rackspace - Privacy Policy, August 2011. Available from: http://www.rackspace.co.uk/legal/privacy-policy/. [Accessed 28 October 2011].

[93] Rackspace - Code of Business Conduct and Ethics. Available from: http://media.corporate-ir.net/media_files/irol/22/221673/cg/CodeofConduct.pdf. [Accessed 28 October 2011].

[94] Enterprise Cloud Computing - Colocation Managed Hosting | Savvis, Inc. Available from: http://www.savvis.com. [Accessed 2 November 2011].

[95] IBM Cloud - Server Configuration - United Kingdom. Available from: http://www-935.ibm.com/services/uk/igs/cloud-development/configurations.html. [Accessed 2 November 2011].

[96] IBM Cloud - Price Sheet - United Kingdom. Last update: August 11, 2011. Available from: http://www-935.ibm.com/services/uk/igs/cloud-development/pricesheet.html. [Accessed 1 November 2011].

[97] SLAs - Contegix | Cloud Hosting, Managed Hosting & Colocation Services, July 2004. Available from: http://www.contegix.com/support/service-level-agreement/. [Accessed 11 January 2012].

[98] Acceptable Use Policy - Contegix | Cloud Hosting, Managed Hosting & Colocation Services, September 2004. Available from: http://www.contegix.com/support/acceptable-use-policy/. [Accessed 11 January 2012].

[99] Privacy Policy - Contegix | Cloud Hosting, Managed Hosting & Colocation Services. Available from: http://www.contegix.com/support/privacy-policy/. [Accessed 11 January 2012].

[100] Support That Never Sleeps - Contegix | Cloud Hosting, Managed Hosting & Colocation Services. Available from: http://www.contegix.com/support/support-that-never-sleeps/. [Accessed 11 January 2012].

[101] Amazon EC2 Dedicated Instances. Available from: http://aws.amazon.com/dedicated-instances/. [Accessed 2 November 2011].

[102] Amazon EC2 Instances Types. Available from: http://aws.amazon.com/ec2/instances-types/. [Accessed 2 November 2011].

[103] Amazon EC2 SLA, October 2008. Available from: http://aws.amazon.com/ec2-sla/. [Accessed 1 November 2011].

[104] AWS Customer Agreement. Last update: August 23, 2011. Available from: http://aws.amazon.com/agreement/. [Accessed 1 November 2011].

[105] AWS Terms of Use. Last update: February 8, 2011. Available from: http://aws.amazon.com/terms/. [Accessed 1 November 2011].

[106] AWS Privacy Policy. Last update: October 1, 2008. Available from: http://aws.amazon.com/privacy/. [Accessed 1 November 2011].

[107] Amazon Web Services: Overview of Security Processes, May 2011. Available from: http://d36cz9buwru1tt.cloudfront.net/pdf/AWS_Security_Whitepaper.pdf. [Accessed 1 November 2011].

Appendix A

How to configure the machines used for testing

This appendix aims to provide the information necessary to replicate the environment used in this project. First, the VPN setup is described, and the VPN configuration files used in this project are given. Second, the Stockholm laboratory B virtual machines' configuration is detailed. Third, the steps to configure an external MSC-S blade are presented and described. Then, basic Ericsson MSC-S BC handling information and commands are provided. In addition, information is provided on how to handle the tools used in the tests in Chapter 4. Finally, the modifications performed in the Ericsson MSC-S BC software are enumerated and explained.

A.1. VPN setup

In order to be able to connect the external blades to the lab where the rest of the MSC-S blade cluster system is located a level 2 bridged VPN is needed. The VPN server is a machine located in such lab with two interfaces, one connected to the lab, where private IP addresses are used, and another connected to the Ericsson intranet, where the external blades will be located.

A.1.1. Installation

OpenVPN is the open source version of VPN in Linux. To install it, the following command must be typed in a Linux console:

```
sudo apt-get install openvpn}\vspace{2mm}
```

A.1.2. Configuration

Once the program is installed in the server, it must be configured. In order to configure the program, the tutorial in [72] was used by the authors of this thesis.

Server certificates

First of all, the server certificates must be created and placed in determined directories. The following commands create the directory, move the necessary example files to it, and assign permission access to the user:

```
sudo mkdir /etc/openvpn/easy-rsa/
sudo cp -r /usr/share/doc/openvpn/examples/easy-rsa/2.0/*
                /etc/openvpn/easy-rsa/
sudo chown -R $USER /etc/openvpn/easy-rsa/
```

The commands provided below create the keys and move them to its directory:

```
cd /etc/openvpn/easy-rsa/
source vars
./clean-all
./build-dh
./pkitool --initca
./pkitool --server server
cd keys
openvpn --genkey --secret ta.key
sudo cp server.crt server.key ca.crt dh1024.pem ta.key
/etc/openvpn/
```

Client certificates

Certificates are also needed for each client that is going to connect to the VPN. These certificates must be created in the VPN server and sent securely to the clients. Each certificate is created running the following commands:

```
cd /etc/openvpn/easy-rsa/
source vars
./pkitool hostname
```

And the following files must be sent to each client and be located in the same directories:

```
/etc/openvpn/ca.crt
/etc/openvpn/easy-rsa/keys/hostname.crt
/etc/openvpn/easy-rsa/keys/hostname.key
/etc/openvpn/ta.key
```

Where hostname is the client's machine name.

Server configuration file

Additionally, the configuration file must be modified. This file must be located in the **/etc/openvpn** directory. An example configuration file is provided by the program in **/usr/share/doc/openvpn/examples/sample-config-files/server.conf.gz**. This is how the configuration file should look like after the modification:

```
# It is the server
mode server
tls-server

# Server IP address in the intranet's interface
local 134.138.89.187

# OpenVPN port
port 1194

# UDP is used
proto udp

# A tap interface is needed to build a level 2 bridged VPN
 dev tap0
--script-security 2

# These are the scripts that bring up or down the bridge
# when VPN is started
# or stoped
up "/etc/openvpn/bridge-start.sh"
down "/etc/openvpn/bridge-stop.sh"

# Try to preserve some state across restarts
persist-key
persist-tun
```

```
# These are the locations of the encryption keys
ca /etc/openvpn/ca.cr
cert /etc/openvpn/server.crt
key /etc/openvpn/server.key
dh /etc/openvpn/dh1024.pem

# The VPN server will have the following IP address and
# will allow the subsequent network mask. The last two
# IP addresses are the first and the last that the server
# will provide to the clients
server-bridge 192.168.169.101 255.255.0.0 192.168.169.200
                192.168.169.255

# This option allows the VPN clients to communicate between
# each other through
 # the VPN server
client-to-client

# Keep-alive time value
keepalive 10 120

# TLS authentication key location
tls-auth /etc/openvpn/ta.key 0

# Compression is enabled in the link
comp-lzo

# Maximum number of clients
max-clients 10
user nobody
group nogroup

# Log and verbosity parameters
status openvpn-status.log
verb 3
```

Listing A.1: OpenVPN Server configuration file

The most important parameters and options are commented. The rest are options by default and its meaning can be found in the original configuration file that is provided by OpenVPN.

The **bridge-start.sh** and **bridge-down.sh** scripts used in this project are:

```bash
#!/bin/bash
#################################
# bridge-start.sh
#################################

# Define Bridge Interface
br="br0"

# Define list of TAP interfaces to be bridged,
# for example tap="tap0 tap1 tap2"
tap="tap0"

# Define physical ethernet interface to be bridged
# with TAP interface(s) above
eth="eth1"
eth_ip="192.168.169.101"
eth_netmask="255.255.255.0"
eth_broadcast="192.168.169.255"

for t in $tap; do
        openvpn --mktun --dev $t
done

brctl addbr $br
brctl addif $br $eth
for t in $tap; do
        brctl addif $br $t
done
for t in $tap; do
        ifconfig $t 0.0.0.0 promisc up
done
ifconfig $eth 0.0.0.0 promisc up
ifconfig $br $eth_ip netmask $eth_netmask broadcast
$eth_broadcast

# The txqueuelen is increased before starting to keep up
# with the high traffic load
ifconfig $tap txqueuelen 5000
ifconfig eth1 txqueuelen 5000
ifconfig eth0 txqueuelen 5000
```

Listing A.2: *bridge-start.sh*

```
#!/bin/bash
####################################
# bridge-stop.sh
####################################

# Define Bridge Interface
br="br0"

# Define list of TAP interfaces to be bridged together
tap="tap0"
ifconfig $br down
brctl delbr $br
for t in $tap; do
        openvpn --rmtun --dev $t
done
```

Listing A.3: *bridge-stop.sh*

These scripts create a bridge between the lab interface and the VPN interface, thus providing LAN access to the lab to every VPN client. The scripts must be granted permission to execute with the following commands:

```
sudo chmod 755 /etc/openvpn/bridge-stop.sh
sudo chmod 755 /etc/openvpn/bridge-start.sh
```

Client configuration file

Each client must have a configuration file to connect to the VPN server. The configuration file used for this project clients is:

```
# It is a client
client

# As in the server, a tap interface is used
dev tap

# UDP is used
proto udp

# The hostname/IP and port of the VPN server
remote 134.138.89.187 1194
```

```
# Keep trying indefinitely to resolve the host name
# of the VPN server
resolv-retry infinite

# The client is not bound to any port
nobind

# Try to preserve some state across restarts
persist-key
persist-tun

# SSL/TLS parameters
ca /etc/openvpn/ca.crt
cert /etc/openvpn/hostname.crt
key /etc/openvpn/hostname.key
tls-auth ta.key 1

# A cryptographic cipher is selected
cipher BF-CBC

# As in the server, compression is enabled in the link
comp-lzo

# Verbosity paremeter
verb 5
```

Listing A.4: OpenVPN Client configuration file

A.1.3. OpenVPN start, restart, and stop

In order to start, restart or stop the OpenVPN server or clients, the following commands must be used respectively:

```
sudo /etc/init.d/openvpn start
sudo /etc/init.d/openvpn restart
sudo /etc/init.d/openvpn stop
```

A.2. Stockholm lab B configuration

As mentioned in Section 4.1.4, in order to dispose of 4 machines in the Stockholm lab B, two virtual machines were created over each physical machine using KVM. In addition, a specific network configuration was necessary in order to enable full connectivity of the resultant virtual machines with the rest of the test environment.

A.2.1. Virtual machines setup

As previously stated in Section 4.1.4, KVM and QEMU were the virtualization software programs chosen to implement the virtual machines in the Stockholm lab B. In addition, KQEMU [78], which is a graphical interface to run virtual machines, was used in order to ease the network configuration of the virtual machines. The reasons for these selections were the open source basis of the programs, as well as their simplicity when running them over a Linux platform. This section describes how the virtual machines were created, the complications that arose in the process, and how they were addressed. The process is explained for one virtual machine, although it just needs to be replicated for any additional virtual machine.

Programs' installation and host machine setup

Several programs must be installed in the host machine so that the virtual machines can be setup. Those programs are installed through the following command:

```
sudo apt-get install qemu-kvm libvirt-bin bridge-utils
kqemu-source kqemu-common
```

This provides the virtualization software as well as the utils necessary to create linux bridges and the virtual machine's graphical interface.

Image creation

First of all, the virtual machine image must be created from an OS International Standard Organization image (ISO image). In this master thesis project, Ubuntu 10.04 LTS 64-bit version was the selected OS. Once the selected OS ISO image is available in a directory, the following commands must be typed to create the virtual machine image:

```
qemu-img create -f qcow2 ubuntu_image.img 32G
qemu-system-x86\_64 -m 8000 -cdrom
        ubuntu-10.04.3-desktop-amd64.iso ubuntu_image.img
```

With the first command an image is created which disk size limit is set to 32 GB. The option **-f qcow2 ubuntu_image.img** sets the new image file name to **ubuntu_image.img**, with **qcow2** as the disk image format. This format contemplates an allocation strategy where the storage is only seized when actually needed.

The second command initiates the OS installation in the created image. The **-m** option indicates the amount of RAM memory used in the OS installation process, and **ubuntu-10.04.3-desktop-amd64.iso** is the name of the OS ISO image file.

Once the OS has been installed, as every change performed in the virtual machine will be stored in the image, it will be persistent. Therefore, once the virtual machine is properly configured, it will always keep all the installed programs and modifications.

Virtual machine network configuration

The virtual machine network interface must be configured so that it can be remotely accessed from any machine in the network. Our solution was to run the virtual machine with the KQEMU graphical interface in the host machine, and configure an interface statically. That was done in the **/etc/network/interface** network configuration file. An example of this configuration file is presented below:

```
auto lo
iface lo inet loopback
auto eth0
iface eth0 inet static
        address IP_address
        netmask 255.255.255.0
```

Listing A.5: Network configuration file

Where "**eth0**" is the interface's name, and "**IP_address**" is the IP address assigned to the virtual machine.

In addition, another modification needed to be done due to an inconsistency regarding the interfaces' names and MAC addresses. Therefore, the names "**eth0**" and "**eth1**" were chosen. Besides, for every virtual machine two MAC addresses needed to be generated. For that purpose we used the following command:

```
openssl rand -hex 6 | sed 's/\(..\)/\1:/g; s/.$//'
```

Once the MAC addresses were obtained, in order for the interfaces' names and MAC addresses to be static the **/etc/udev/rules.d/70-persitent-net** configuration file needed to be modified. To do so, it was necessary to hardcode both names and MAC addresses in the configuration file as in the example below:

```
# PCI device 0x1af4:0x1000 (virtio-pci)
SUBSYSTEM="net" ACTION=="add", DRIVERS=="?*",
ATTR{address}=="MAC_eth0", ATTR{type}=="1",
KERNEL=="eth0", NAME="eth0"
```

```
# PCI device 0x1af4:0x1000 (virtio-pci)
SUBSYSTEM=="net", ACTION=="add", DRIVERS=="?*",
ATTR{address}=="MAC_eth1", ATTR{dev_id}=="0x0",
ATTR{type}=="1", KERNEL=="eth1", NAME="eth1"
```

Listing A.6: *70-persistent-net* configuration file

Where "**eth0**" and "**eth1**" are the interfaces' names, and "**MAC_eth0**" and "**MAC_eth1**" are their MAC addresses.

Subsequently, two bridges must be created in the host machine so that the virtual machines' interfaces setup later on can be reached remotely. This is done through the following command:

```
brctl addbr br0 br1
```

When starting a virtual machine, a tap interface will be created in the host machine for every virtual machine's interface. Then, it will be added to one of the bridges set up in the host machine. Since two interfaces are created in every virtual machine, they will be added to different bridges in order to be able to set up the topology illustrated in Figure 4.1.

Finally, a route must be added in the host machine so that the virtual machine is reachable. The route will go through one of the bridges previously set up, as they are linked to the tap interfaces that connect the host to the virtual machine, and will point to the virtual machine's IP address statically configured. This route can be added through the following command:

```
route add -net net_prefix netmask net_mask
```

Where "**net_prefix**" is the network prefix, and "**net_mask**" is the network mask.

Running the virtual machine

Once the OS has been installed, and the network configured, the virtual machine can be initiated. The following command runs a virtual machine from the newly created image:

```
qemu-system-x86_64 -enable-kvm -hda ubuntu_image.img
        -boot c -m 24000 -k sv -net nic, macaddr=MAC_eth0,
        model=virtio,name=eth0, netdev=tap1 -netdev tap,
        id=tap1, script=/etc/qemu-ifup,
        downscript=/etc/qemu-ifdown -net nic,
        model=virtio, macaddr=MAC_eth1, name=eth1,
        netdev=tap2 -netdev tap, id=tap2,
        script=/etc/qemu-ifup1, downscript=/etc/qemu-ifdown1
        -smp 4 &
```

As it can be appreciated in the command, two interfaces are setup in the virtual machine, as they are enough for the thesis project purpose. Below, an explanation of the utilized options is provided:

- Since the OS that is necessary for our purposes is 64 bits-based the qemu command must be **qemu-system-x86_64**.
- The **-enable-kvm** command enables KVM full virtualization support.
- The **-hda ubuntu_image.img** command designates "`ubuntu_image.img`" as the image file from which to load the virtual machine.
- The **-boot c** command orders to boot the virtual machine from hard disk.
- The **-m 24000** command sets the RAM memory of the virtual machine to 24 GB.
- The **-k sv** command sets the keyboard type to swedish sort.
- The **-net nic,macaddr=MAC_eth0,model=virtio,name=eth0, netdev=tap1 -netdev tap,id=tap1,script=/etc/qemu-ifup, downscript=/etc/qemu-ifdown** command sets up an interface in the loaded virtual machine with the following characteristics:
  - MAC address "**MAC_eth0**".
  - Interface driver **virtio**, which was selected because it supports VLAN tagging.
  - Interface name **eth0**.
  - **tap** is selected as the inteface type to connect to the host machine, and that interface name is **tap0**.

  – **/etc/qemu-ifup** and **/etc/qemu-ifdown** are the scripts that bring the tap interface either up or down in the host machine whenever the virtual machine is loaded or shutdown, respectively. In addition, they add those interfaces to a bridge so that the virtual machine can be reachable from the host machine. Those scripts are provided below:

```
#qemu-ifup
#!/bin/sh
set -x
switch=br0
vtap=tap0}
tunctl -u root -t $1
ip link set $1 up
brctl addif $switch $1
ifconfig $1 txqueuelen 5000
exit 0
```

Listing A.7: *qemu-ifup* script

```
#qemu-ifdown
#!/bin/sh
set -x
tunctl -d $1
exit 0
```

Listing A.8: *qemu-ifdown* script

- The **-net nic,model=virtio,macaddr=MAC_eth1,name=eth1,netdev=tap2 -netdev tap,id=tap2,script=/etc/qemu-ifup1,downscript=/etc/qemu-ifdown1** command is equivalent to the last one, and creates the second virtual machine interface.

- Finally, the **-smp 4** command sets the number of processors of the virtual machine to 4.

Remote access to the virtual machine

Once all the previous steps have been followed to set up the virtual machine, it will be reachable through SSH by issuing the following command:

```
ssh root@IP_address
```

Where **IP__address** is the virtual machine IP address. Later on the machine's password would need to be introduced to successfully access it.

A.2.2. Network configuration

In order to obtain the network topology configuration drawn in Figure 4.1 three more steps needed to be taken in the two host machines that were utilized:

- First, the host machines needed to connect to the *Bridge* machine through VPN, so that they had communication with the rest of the network.
- Then, the VPN resulting tap interfaces needed to be added to the bridge that already contained the tap interfaces corresponding to the virtual machines' interfaces meant to connect to the rest of the network.
- Finally, the host machines' interfaces used to connect internally the four virtual machines in the Stockholm lab B needed to be added to the remaining bridge.

A.3. Blade configuration

The MSC-S blades are the main part of the system. As it is explained in Section 2.4, the MSC-S blades are the components in charge of controlling all circuit-switched call services, the user plane, and the MGWs. The MSC-S blade configuration is described in this section, which is divided in three subsections: network configuration, running the APZ Virtual Machine, and loading the software dump.

A.3.1. Network configuration

Once the VPN connection is created and the communication is estabished between the bridge machine and the machine that is going to be used as a MSC-S blade, the necessary software and configuration files must be transferred to the blade machine (via SCP or FTP).

After that, the network interfaces must be configured. As it is explained in Section 4.1.4, six virtual interfaces are used on each MSC-S blade, and their IP configuration uses the following rules due to constraints of the system:

- One interface is connected to VLAN 169, and its IP address is 192.168.169.**X**.
- One interface is connected to VLAN 170, and its IP address is 192.168.170.**X**.
- One interface is connected to VLAN 171, and its IP address is 192.168.171.**X**.

- One interface is connected to VLAN 172, and its IP address is 192.168.172.**X**.
- One interface is connected to VLAN 173, and its IP address is 192.168.173.**X**.
- One interface is connected to VLAN 1001, and its IP address is 192.168.1.**Y**.

Where **X** is equal to "$71 + N$" and **Y** is equal to "$103 + N$", being N the number assigned to the blade in the system.

The configuration would be slightly different depending on whether the blade is going to use one or two ethernet interfaces. In the case of one interface, all the virtual interfaces (VLANs interfaces) would need to be attached to the VPN interface so that all the system elements can be reached. However, in the case of two interfaces, the virtual interfaces associated to VLANs 169 and 1001 would need to be attached to the VPN interface to connect to the SPX and APG, while the virtual interfaces corresponding to VLANs 170, 171, and 172 could be attached to an additional interface to connect to the other MSC-S blades.

One interface

A blade that is going to use just one physical interface requires the following configuration script file to be run:

```
./network_bladeX_vpn
```

Where the **"X"** is the number assigned to the blade in the system.

This file creates the necessary VLANs to run the MSC-S blade, configures its virtual interfaces, and increases the TCP and SCTP buffer sizes. The file must be modified to set the interface that is going to be used to communicate with the bridge machine through the VPN connection, as well as to define the IP addresses assigned to the blade's virtual interfaces.

The configuration file is provided below, where **"X"** is the number of the VPN tap interface that is going to be used, and **"Y"** and **"Z"** are the blade IP addresses' last octets after applying the rules indicated in the beginning of this section:

```
#/etc/init.d/network restart
#openvpn /etc/openvpn/client.conf

modprobe sctp
swapoff -a -v
sysctl net.sctp.sack_timeout="40 60"
sysctl net.ipv4.neigh.default.locktime=0
sysctl net.sctp.rto_initial=200
sysctl net.sctp.rto_min=100
sysctl net.sctp.rto_max=400
sysctl net.sctp.hb_interval=100

sysctl -w net.core.rmem_default=524287
sysctl -w net.core.wmem_default=524287
sysctl -w net.core.rmem_max=524287
sysctl -w net.core.wmem_max=524287
sysctl -w net.core.optmem_max=524287
sysctl -w net.core.netdev_max_backlog=300000
sysctl -w net.ipv4.tcp_rmem="10000000 10000000 10000000"
sysctl -w net.ipv4.tcp_wmem="10000000 10000000 10000000"
sysctl -w net.ipv4.tcp_mem="10000000 10000000 10000000"
modprobe sctp
sysctl -w net.sctp.sctp_rmem="10000000 10000000 10000000"
sysctl -w net.sctp.sctp_wmem="10000000 10000000 10000000"
sysctl -w net.sctp.sctp_mem="10000000 10000000 10000000'"

echo 8 > /proc/irq/25/smp_affinity

echo "XPU settings"
sysctl fs.mqueue.msgsize_max=8500
mkdir -p /dev/mqueue
mount -t mqueue none /dev/mqueue
mkdir -p /dev/shm
mount -t tmpfs tmpfs /dev/shm
mkdir -p /xpu
mount -t tmpfs tmpfs /xpu -o size=100M
mkdir -p /cgroups/cpu
mkdir -p /cgroups/mem
mount -t cgroup cpuset /cgroups/cpu -o cpuset
mkdir -p /cgroups/cpu/XPUSet
mkdir -p /cgroups/mem/XPUSet
echo 2G > /cgroups/mem/XPUSet/memory.limit_in_bytes

ifconfig tapX 192.168.169.Y netmask 255.255.255.0 up
ifconfig tapX mtu 1500
ifconfig tapX:A 192.168.170.Y netmask 255.255.255.0 up
```

```
sysctl net.ipv4.neigh.tapX.locktime=0
vconfig add tapX 171
ifconfig tapX.171 192.168.171.Y netmask 255.255.255.0
vconfig add tapX 172
ifconfig tapX.172 192.168.172.Y netmask 255.255.255.0
vconfig add tapX 173
ifconfig tapX.173 192.168.173.Y netmask 255.255.255.0
vconfig add tapX 1001
ifconfig tapX.1001 192.168.1.Z netmask 255.255.255.0
```

Listing A.9: *network_bladeX_vpn* configuration file

Two interfaces

A blade that is going to use two physical interfaces requires the following configuration script file to be run:

```
./network_bladeX_dual
```

Where the **"X"** is the number assigned to the blade in the system.

This file creates the necessary VLANs to run the MSC-S blade, configures its virtual interfaces, and increases the TCP and SCTP buffer sizes. The file must be modified to set the interfaces that are going to be used to communicate with the bridge machine through the VPN connection, and with the rest of the blades internally, as well as to define the IP addresses assigned to the blade's virtual interfaces.

The configuration file is provided below, where **"X"** is the number of the VPN tap interface used for the VPN connection, **"Y"** is the number of the internal interface used to communicate with the rest of the blades, and **"W"** and **"Z"** are the blade IP addresses' last octets after applying the rules indicated in the beginning of section A.3.1:

```
#/etc/init.d/network restart
#openvpn /etc/openvpn/client.conf

modprobe sctp
swapoff -a -v
sysctl net.sctp.sack_timeout="40 60"
sysctl net.ipv4.neigh.default.locktime=0
sysctl net.ipv4.neigh.tapX.locktime=0
sysctl net.ipv4.neigh.ethY.locktime=0
sysctl net.sctp.rto_initial=200
sysctl net.sctp.rto_min=100
sysctl net.sctp.rto_max=400
sysctl net.sctp.hb_interval=100

sysctl -w net.core.rmem_default=524287
sysctl -w net.core.wmem_default=524287
sysctl -w net.core.rmem_max=524287
sysctl -w net.core.wmem_max=524287
sysctl -w net.core.optmem_max=524287
sysctl -w net.core.netdev_max_backlog=300000
sysctl -w net.ipv4.tcp_rmem="10000000 10000000 10000000"
sysctl -w net.ipv4.tcp_wmem="10000000 10000000 10000000"
sysctl -w net.ipv4.tcp_mem="10000000 10000000 10000000"
modprobe sctp
sysctl -w net.sctp.sctp_rmem="10000000 10000000 10000000"
sysctl -w net.sctp.sctp_wmem="10000000 10000000 10000000"
sysctl -w net.sctp.sctp_mem="10000000 10000000 10000000"

echo 8 > /proc/irq/25/smp_affinity

echo "XPU settings"
sysctl fs.mqueue.msgsize_max=8500
mkdir -p /dev/mqueue
mount -t mqueue none /dev/mqueue
mkdir -p /dev/shm
mount -t tmpfs tmpfs /dev/shm
mkdir -p /xpu
mount -t tmpfs tmpfs /xpu -o size=100M
mkdir -p /cgroups/cpu
mkdir -p /cgroups/mem
mount -t cgroup cpuset /cgroups/cpu -o cpuset
mkdir -p /cgroups/cpu/XPUSet
mkdir -p /cgroups/mem/XPUSet
echo 2G > /cgroups/mem/XPUSet/memory.limit_in_bytes
```

```
ifconfig tapX 192.168.169.W netmask 255.255.255.0 up
ifconfig tapX mtu 1500
ifconfig ethY mtu 1500
ifconfig tapX:A 192.168.170.W netmask 255.255.255.0 up
vconfig add ethY 171
ifconfig ethY.171 192.168.171.W netmask 255.255.255.0
vconfig add ethY 172
ifconfig ethY.172 192.168.172.W netmask 255.255.255.0
vconfig add ethY 173
ifconfig ethY.173 192.168.173.W netmask 255.255.255.0
vconfig add tapX 1001
ifconfig tapX.1001 192.168.1.Z netmask 255.255.255.0
```

Listing A.10: *network_bladeX_dual* configuration file

A.3.2. Running the APZ Virtual Machine

The APZ Virtual Machine (APZ VM) is a middleware program on top of which the MSC-S functionality is run in a standard PC. Before running the APZ VM, several system variables must be setup by running the following script file:

```
source config_bladeX_vpn
```

Where the **"X"** is the number assigned to the blade in the system.

The configuration file is provided below, where **"X"** is the number of the interface used to communicate with the *Bridge* machine, the **"Y"** is the number assigned to the blade in the system, and the **"Z"** are the blade IP addresses' last octets after applying the rules indicated in the beginning of Section A.3.1.

```
#Ericsson specific variables /etc/init.d/boot.d/S16network.sh
export DEV_ETHA=intl0
export DEV_ETHB=intl1
export ISOB_VLAN_ID=4042
export ISOS_VLAN_ID=4043
export TFTP_SERVER=169.254.190.18
export BLADE_SYS_NAME=bs_CPHW_6
export UNTAGGED_IP=169.254.131.114
export UNTAGGED_IP_MASK=255.255.192.0
export NTP_SERVERS=134.138.168.247
export SNMP_RECEIVER=169.254.64.2
export LAG_IF=ethX
export DEV_IPETHA=ethX
export DEV_IPETHB=ethX
export OWN_ISOB_IPADDRESS=169.254.3.113
export ISOB_TFTP_SERVER=169.254.62.18
export ISOS_IPADDRESS=169.254.64.7
export ISOS_NETMASK=255.255.224.0
#Ericsson specific variables /etc/init.d/boot.d/S24getCpd.sh
export IS_SLOT_ID=23
export IS_SUBRACK_ID=1
export IS_BLADE_SYSTEM_NAME=bs_CPHW_6
export IS_BLADE_IP=169.254.3.113
export IS_BSOM_ADDRESS=169.254.64.7
export IS_ISOB_ADDRESS=169.254.3.113
export IS_ISOS_ADDRESS=169.254.64.7
export IS_ISOB_NETMASK=255.255.192.0
export IS_ISOS_NETMASK=255.255.224.0
export IS_ISOB_BROADCAST_ADDRESS=169.254.63.255
export IS_ISOS_BROADCAST_ADDRESS=169.254.95.255
export IS_RLSP_MARKER_GENERATE_INTERVAL=10
export IS_RLSP_LINK_DOWN_THRESHOLD=3
export IS_RLSP_LINK_UP_THRESHOLD=3
export IS_BLADE_NAME=blade_1_23
export IS_KERNEL_PATH=
#Ericsson specific variables /etc/init.d/boot.d/S26APdhcp.sh
export CPUB_TYPE=BLADE
export VLAN_A_CFG='169 3'
export VLAN_B_CFG='170 3'
export VLAN_A=169
export VLAN_B=170
export SLOT_NO=23
export BOARD_LOCATION=0.0.1.23
export CPHW_KERNEL_NFS=/private/blade_1_23/sw/CPHW_K*
export BLADE_BOOT_TFTP=.boot
export OWN_VLAN_A_IPADDRESS=192.168.169.Z
export OWN_VLAN_A_NETMASK=255.255.255.0
export OWN_VLAN_B_IPADDRESS=192.168.170.Z
export OWN_VLAN_B_NETMASK=255.255.255.0
export CP_NUM=Y
export CP_NAME=BCY
export CPHW_LOG_ROOT=APZ/Logs/BCY/CPA/CPHW
export CONFIGDB_FILE=APZ/data/BCY/CPA/configdb.ini
export RUF_FILE_ROOT=APZ/data/BCY/CPA/fw_upgrade
#Ericsson specific variables /etc/init.d/boot.d/S32setIPHost.sh
export CP_SIDE_IS_A=1
```

```
#Ericsson specific variables /etc/init.d/boot.d/S64clearExempt.sh
export LAST_BOOT_WAS_SOFT=0
export SCN_PCAP_FILESIZE=15000000
export SCN_PCAP_POINT_SULA_RX=1
export SCN_PCAP_POINT_SULA_TX=1
export SCN_PCAP_POINT_KIP_RX=1
export OWN_CH_IP=192.168.172.Z
export OWN_DEF_IP=192.168.173.Z
export OWN_CP2P_IP=192.168.171.Z
export XPU_ROOT=/xpu
```

Listing A.11: *config_bladeX_vpn* configuration file

Finally, the APZ VM is run by typing the following command:

```
./APZ_VM.ubuntu -i 0 -ds 17000 -no X
```

Where the **"X"** is the number assigned to the blade in the system.

The option *-ds* provides the amount of RAM memory that is exclusively assigned to that blade. The minimum amount necessary to run the APZ VM is 16000 MB. However, 1 GB extra was added to that number as a prevention.

It is important to remark that this program runs in first plane if no "&" is used at the end of the line. This means that the console used to run the APZ VM is going to be blocked by the application, therefore easing the visualization of the printouts.

A.3.3. Software dump

After running the APZ VM, a software dump is required in the blade to continue with its setup process. In order to carry out this software dump an integrated console must be opened by running the following executable file:

```
./SystemCommandConsole localhost 9000
```

Subsequently, the following printout with a blinking prompt at the end should be obtained:

```
*************************
** SystemCommandConsole **
*************************

-----------------------------------------------------------
                 Connected to remote system
-----------------------------------------------------------
```

Now, the following command must be typed to make the software dump in the corresponding blade:

```
ptcpl file=relfswX/
```

Where the **"X"** is the number of the software release that it is going to be loaded in the blade.

This action should render a printout as the one presented below:

```
InitialLoad started
Reading BackupInfo from RELFSWX/BUINFO (564 bytes).
Reading LDD from RELFSWX/LDD1 (564347648 bytes).
Reading PS from RELFSWX/PS (137784904 bytes).
```

```
Reading RS from RELFSWX/RS (15517572 bytes).
Performing Consistency Checks on BUA
Loading from Backup Area
Dump compatibility check returned OK
executed.
****** A global initial start is performed ******
Phase 1
Phase 2
Phase 3
Phase 4
[...]
Phase 254
****** End of restart ******
```

This process must be repeated for every blade that is going to be part of the cluster.

A.4. Cluster setup

Once the blades are up and running, and the software dump has been done, the cluster must be brought up by running commands in the APG component. The following steps must be done to start a cluster:

- First of all, all the blades that are going to be part of the cluster must be restarted from the APG running the following commands[1]:

```
mml -cp allbc
syati:restart;
exit;
```

- Secondly, once the blades' restart is finished, the application that the cluster is going to run (MSC) must be specified for each blade that is going to be part of the cluster:

```
mml -cp bcX
cqmsi:app=msc;
exit;
```

  Where "**X**" is the number assigned to the blade in the system.

- Finally, the cluster must be started running the following command from the APG in one of the blades that is going to be part of the cluster:

```
mml -cp bcX
cqaci:cp=expression
exit;
```

  Where "**X**" is the number assigned to the blade in the system, and "**expression**" is a regular expression that indicates the blades that are going to be part of the cluster. This regular expression can be composed of the two following operators:

  - **&**: Where $X\&Y$ is equivalent to "X and Y".
  - **&&**: Where $X\&\&Y$ is equivalent to "from X to Y".

[1]When using *mml* commands a `-c` can be added to the instruction (for instance, `mml -cp allbc -c`) so that the commands run from there on will not need confirmation. If this option is not used, every *mml* command will need to be confirmed by typing "`;`" after the command.

Where "*X*" and "*Y*" are numbers assigned to blades in the system.

As an example, the expression 1&&4&6 means that the cluster would be formed by blades 1,2,3,4 and 6.

A.5. Add a blade to the cluster

When a blade is to be added to an active cluster the following steps must be followed:

- First, the blade must be configured as in Section A.3.
- Secondly, the blade must be restarted and, once the blade's restart is finished, the MSC application must be designated, as done with the blades that form the cluster, by executing in the APG the commands expressed below:

```
mml -cp bcX
syati:restart;
cqmsi:app=msc;
exit;
```

- Finally, the blade must be added to the cluster using the following commands:

```
mml -c
cqaci:cp=X;
exit;
```

Where "**X**" is the number assigned to the blade in the system.

A.6. Remove a blade from the cluster

In order to remove a blade from a cluster several steps must be taken:

- If the blade is in "active" state, it must be first isolated by running the following commands:

```
mml -c
cqtii:state=interm,delay=00-00,cp=X;
```

- In the next step the blade is sent to a non-operation state by executing:

```
cqmse:cp=X;
```

- Finally, the blade is removed completely removed from the cluster:

```
cqmsr:cp=X;
exit;
```

Where "**X**" is the number assigned to the blade in the system.

It is important to note that the minimum number of blades in a cluster is 2. Therefore, if a blade is to be removed having this minimum, the process to follow would be the one indicated in A.7.

A.7. Cluster tear down

When a cluster tear down is required the following commands must be issued for each blade that composes the cluster:

```
mml -cp bcX
ptcoi;
y;
```

```
ptcrl:pro=gpr,reg=cpclusterstate,data=0;
ptcrl:pro=gpr,reg=appltype,data=0;
ptcpl:file=release;
ptcoe;
exit;
```

Where "**X**" is the number assigned to the blade in the system, and "**release**" is the software release that is going to be loaded. When the blades are external, the software load will not be performed, and therefore it would need to be done manually, as described in A.3.3. However, they will be removed from the cluster anyway.

A.8. Cluster information and statistics

There are several commands very useful to recover statistics and other information from the cluster. The commands used in this master thesis project are described and explained in this section.

A.8.1. Cluster state

In order to inquire about the cluster state the following commands must be issued in the APG console:

```
mml -c
cqmsp:cp=all;
```

Once these commands are issued, the console must be unlocked by typing *Ctrl + D*. Then, the state of the cluster will be printed, where they inform about the state of every blade in the cluster, including themselves. An example of this printout is:

```
PAS                            AD-326  TIME 980101 0012  BC11
QUORUM MEMBERSHIP DATA
 CP  CPNAME  STATE     APP    TRAFFIC   APZ-SUBST  APT-SUBST
  1  BC1     ACTIVE    MSC    NORMAL    UNDEF      H'01
  2  BC2     ACTIVE    MSC    NORMAL    UNDEF      H'01
  3  BC3     ACTIVE    MSC    NORMAL    UNDEF      H'01
  4  BC4     ACTIVE    MSC    NORMAL    UNDEF      H'01
 11  BC11    PASSIVE   MSC    NONE      DEFAULT    UNDEF
 13  BC13    ACTIVE    MSC    NORMAL    UNDEF      H'01
 14  BC14    ACTIVE    MSC    NORMAL    UNDEF      H'01
END
```

In addition, the same command can be run for a single blade in order to know what is the state of the cluster from that blade view:

```
mml -cp bcX
cqmsp:cp=all;
```

Where "**X**" is the number assigned to the blade in the system. However, when the cluster is stable and there is no inconsistency, all the blades must show the same cluster state.

A.8.2. Blades' processor load

The load of every blade's processor in the cluster can be printed using the following command:

```
mml -cp allbc
plldp;
```

This command provides a printout where the processors' load appears in the second column to the right.

A.8.3. Cluster subscribers' information

The number of subscribers handled by every blade that is within the cluster can be seen typing the following command:

```
mml -cp allbc
mgsvp;
```

Once these commands are issued, the console must be unlocked by typing *Ctrl + D*. After that, the printout with the data will appear in the APG console.

A.8.4. Event reporting

In order to recover event report information from the cluster, the following commands are available to initiate, print and finish that report:

```
mml -c
erepi:enum=all;
```

The latter command initiates a report for every type of event that could happen in the cluster. If specific events are to be covered they would need to be typed instead of `all` using a regular expression with the same rules as in Section A.4. Once the report is initiated, it can be printed using:

```
mml -c
erepp:enum=all;
```

Finally, if the report needs to be stopped, it could be done issuing the following commands:

```
mml -c
erepe:enum=all;
```

A.8.5. Alarms

There are two different ways to check the cluster alarms:

- If the alarms are to be checked as they are triggered, the "alarm mode" of the APG console would need to be accessed by typing the following command:

```
mml -c -a
```

  In this case the console would need to be unlocked by issuing *Ctrl* + *D* so that the alarms are printed as they occur.

- The alarms can also be checked by typing an *mml* command that would yield the alarms' that took place up to that moment:

```
mml -c
allip;
```

A.8.6. Print size alteration events

The following command yields information of whether a Size Alteration Event (SAE) has ocurred, and what memory block has triggered the event, for all the blades in the cluster:

```
mml -cp allbc
dbtsp:tab=saactions;
```

A.9. Traffic generators

This section provides more information regarding the mobile networks' signalling traffic generators used in this thesis project, as well as a manual explaining how to use them. Traffic generators were used in this thesis project to simulate two different components:

HLR: All the functionalities of an HLR are replicated by a traffic generator: the subscribers database and the message exchange with the rest of the system.

RAN: The whole Radio Access Network is simulated by a traffic generator. In this project, two BSCs and two RNCs were replicated, thus generating the signalling traffic equivalent to both subscribers' location updates, and GSM and UMTS calls.

A.9.1. HLR traffic generator

In this section, the HLR traffic generator setup is described. This setup will leave the simulated HLR running so that it can interact with the rest of the elements in the system:

- First of all, a connection via SSH to the machine in which the HLR traffic generator is located is needed. Once the connection is setup, the following commands must be introduced in this machine in order to run the HLR traffic generator:

```
cd /home/tgen/tgen\_R1N
./tgen_R1N -c network_SCP_HLR_02_R1N -p 9000 -s --camel &
```

- Once the HLR traffic generator has been successfully initialized, the following command must be typed in order to access the console that controls it:

```
telnet localhost 9000
```

- If the HLR traffic generator's setup has been successful, the following printout would be obtained:

```
TGEN - Traffic Generator.
LXA 104 04, R1N, 2011-09-21.
Copyright (C) 2005-2009 Ericsson AB, Sweden.
```

```
Traffic Generator, R1N. Running Time 0H:00M:10S

TGEN>
```

In this prompt, several options are available:

HELP Print help message.

PRINT Print the different traffic cases with their statistics.

EXIT Terminate the Traffic Generator (TGen).

QUIT Quit from the HLR remote command session (Exit without closing the traffic generator).

A.9.2. RAN traffic generator

The RAN traffic generator is a piece of software that simulates the signalling originated to setup GSM and UMTS calls, and subscribers' location updates. This tool substitutes the RAN components of a mobile network in the tests implemented in this master thesis.

First of all, a connection via SSH to the machine in which RAN traffic generator is going to be run. Once the connection is setup, the directory where the application is located must be accessed. In our case, this would done with the following command:

```
cd /home/tgen/tgen_R1N/
```

At this point, there are two paths depending on the traffic case that is going to be run: periodic subscribers' location updates, or periodic GSM and UMTS calls.

Subscribers location updates

The following commands must be introduced in order to run the traffic generator with periodic location updates as the only traffic:

```
./tgen_R1N -c network_M1BSC_M2RNC_40ks_BS_LU -p 9000 -s &
```

Where `network_M1BSC_M2RNC_40ks_BS_LU` is the configuration file associated to this traffic case.

In order to access the console from which the program is controlled, the following command must be typed:

```
telnet localhost 9000
```

If the program setup has been successful, a printout like the following would be obtained:

```
TGEN - Traffic Generator.
LXA 104 04, R1N, 2011-09-21.
Copyright (C) 2005-2009 Ericsson AB, Sweden.

Traffic Generator, R1N. Running Time 0H:00M:11S

TGEN>
```

In order to start running periodic location updates, the following command must be typed in the prompt:

```
run UMTS_LU_PERIODIC cr=X
```

Where the **"X"** is the desired periodic location updates' rate in *location updates/s*.

There are several other commands that can be used from this console:

HELP Print help message.

PRINT Print the statistics associated with the traffic that is being run.

EXIT Terminate the Traffic Generator (TGen).

QUIT Quit from the remote command session (Exit without close the TGen).

GSM and UMTS calls

The following command needs to be introduced in order to run the traffic generator with periodic GSM and UMTS calls:

```
./tgen_R1N -c network_M1BSC_M2RNC_40ks_BS -p 9000 &
```

After typing the command above, if the subscribers' registration has been successful the following printout would be obtained:

```
Initializing SS7 signalling channels.

TGen Remote Control Command Server, port = 9000.

Simulation started on bc48tgen2 at Fri Jan 13 15:47:10 2012.

Network Element    SS7 CP    Type      SS7 link state
bsc6000              6000    BSC       READY
bsc6100              6100    BSC       READY
rnc6510              6510    RNC       READY
rnc6610              6610    RNC       READY

SS7 signalling channels are initialized.
Starting mobile registration phase.
Registered GSM mobiles 0019994, Registered UMTS mobiles
        0019992,  0000014 in progress
Registered GSM mobiles 0020000, Registered UMTS mobiles
        0019996.  0000004 in progress

End of GSM mobile registration phase (ready for traffic).

Registered GSM mobiles 0020000, Registered UMTS mobiles
        0020000.  0000000 in progress

End of UMTS mobile registration phase (ready for traffic).
```

However, if there is any connectivity issue either with the HLR or the Ericsson MSC-S BC, the subscribers' registration would fail and the program would abort.

In order to access to the console from which the RAN traffic generator can be controlled, the following command needs to be introduced:

```
telnet localhost 9000
```

If the program setup has been successful, a printout like the following would be obtained:

```
TGEN - Traffic Generator.
LXA 104 04, R1N, 2011-09-21.
Copyright (C) 2005-2009 Ericsson AB, Sweden.
```

```
Traffic Generator, R1N. Running Time 0H:09M:26S

TGEN>
```

In this prompt, several options can be used:

HELP Print help message.

PRINT Print the statistics associated with the traffic that is being run.

EXIT Terminate the Traffic Generator (TGen).

QUIT Quit from the remote command session (Exit without close the TGen).

In order to start running GSM and UMTS periodic calls, the following commands must be typed in the program console:

```
run GSM_MOB_TO_MOB cr=Y
run UMTS_MOB_TO_MOB cr=Y
```

Where the "**X**" and "**Y**" are the desired call rates in *calls/s* for GSM and UMTS, respectively.

Every call consists on the signalling process necessary in the system to set the call up and, when that is done, the process to terminate it. Therefore, with this traffic generator no simulated voice conversation takes place in the call process, as it is not a concern of the signalling part of the system.

Once the traffic is running, there are several options that can be used to either modify the generated traffic or obtain statistics from the program:

```
setcalls GSM_MOB_TO_MOB cr=X
```

This command is used to set the GSM call rate to the value fixed by "**X**". To use this command it is needed to have traffic running.

```
setcalls UMTS_MOB_TO_MOB cr=X
```

This command is used to set the UMTS call rate to the value fixed by "**X**". To use this command it is needed to have traffic running.

```
clearstats
```

This command is used to reset the statistics shown by the command *print*.

```
stop
```

This command is used to stop the running traffic. It stops all the traffic cases that are running in the moment.

A.10. Netem

This section describes the operation of the Netem tool, which was used in this thesis project in order to introduce network penalties, and, therefore, emulate the irregularities that surround a cloud environment. As the only functions of this tool that were used in the project are delay, and packet loss addition to a determined interface, this document only provides information about the handling of those features.

Additionally, the authors of this thesis used the contents of this tutorial [79] in order to master the tool operation. This tutorial also provides more information regarding other Netem functions.

A.10.1. Installation

As we were running a Linux distribution later than 2.6, Netem was already enabled in the kernel and nothing else needs to be installed. Besides, no further configuration is needed, as the tool is easily controlled through the `tc` command of the *iproute2* package of tools [79].

A.10.2. Delay

The introduction of delay in a specific interface with the Netem tool is trivially done through the following command:

```
tc qdisc add dev interface root netem delay time_units
```

Where "**interface**" is the interface where the packets are going to be delayed, and "**time_units**" is the delay time along with its unit.

In addition, if a delay is already introduced in an interface, its value can be changed with the following command:

```
tc qdisc change dev interface root netem delay time_units
```

Where "**interface**" is the interface where the delay is going to be changed, and "**time_units**" is the new delay time along with its unit.

Finally, if the delay needs to be removed, the next command must be typed:

```
tc qdisc del dev interface root netem delay time_units
```

Where "**interface**" is the interface from where the delay is going to be removed, and "**time_units**" is the actual delay time in the interface along with its unit.

A.10.3. Packet loss

The addition of packet loss in a specific interface with the Netem tool is done through the following command:

```
tc qdisc add dev interface root netem loss rate%
```

Where "**interface**" is the interface where the packets are going to be dropped, and "**rate**" is the packet loss rate in percentage.

In addition, if a packet loss rate is already added to an interface, its value can be changed with the following command:

```
tc qdisc change dev interface root netem loss rate%
```

Where "**interface**" is the interface where the packet loss rate is going to be changed, and "**rate**" is the new packet loss rate in percentage.

Finally, if the packet loss needs to be removed, the next command must be typed:

```
tc qdisc del dev interface root netem loss rate%
```

Where "**interface**" is the interface from where the packet loss rate is going to be removed, and "**rate**" is the actual packet loss rate in the interface.

A.11. Blade cluster modifications

This section details the changes that were performed over the default blade cluster configuration, so that its performance under the test environment conditions was satisfactory. The two modifications carried out were the SAE data files' resize and the cloning timer removal.

A.11.1. SAE data files resize

During the tests, the default memory blocks' sizes of all the blades needed to be modified due to the overflow of SAE data files, caused by the network penalties introduced to the traffic. The SAE data files that were resized, and the commands used for that modification are the following:

- Local SAE **500**, block **SHTTM**. Size from 1820 to 2500:

```
mml -cp allbc
saaii:sae=500,block=SHTTM,ni=2500;
```

- Local SAE **701**, block **C7TCP**. Size from 4000 to 5000:

```
mml -cp allbc
saaii:sae=701,block=C7TCP,ni=5000;
```

- Local SAE **527**, block **MCCMH**. Size from 500 to 1000:

```
mml -cp allbc
saaii:sae=527,block=MCCMH,ni=1000;
```

- Local SAE **700**, block **C7TCP**. Size from 5000 to 6000:

```
mml -cp allbc
saaii:sae=700,block=C7TCP,ni=6000;
```

- Local SAE **500**, block **GMAPVS**. Size from 1500 to 3000:

```
mml -cp allbc
saaii:sae=500,block=GMAPVS,ni=3000;
```

- Global SAE **948**. Size from 2000 to 3000:

```
mml -cp allbc
saaii:sae=948,ni=3000;
```

- Global SAE **944**. Size from 2000 to 2500:

```
mml -cp allbc
saaii:sae=944,ni=2500;
```

- Local SAE **500**, block **MAUTH**. Size from 4500 to 5000:

```
mml -cp allbc
saaii:sae=500,block=MAUTH,ni=5000;
```

- Local SAE **527**, block **MRMMH**. Size from 8000 to 9000:

```
mml -cp allbc
saaii:sae=527,block=MRRMH,ni=9000;
```

- Local SAE **604**, block **SHGTCU**. Size from 2000 to 2500:

```
mml -cp allbc
saaii:sae=604,block=SHGTCU,ni=2500;
```

- Local SAE **500**, block **GAP3**. Size from 1500 to 2000:

```
mml -cp allbc
saaii:sae=500,block=GAP3,ni=2000;
```

- Local SAE **605**, block **SHTRI**. Size from 1200 to 2000:

```
mml -cp allbc
saaii:sae=605,block=SHTRI,ni=2000;
```

- Global SAE **949**. Size from 3000 to 5000:

```
mml -cp allbc
saaii:sae=949,ni=5000;
```

- Local SAE **500**, block **MVIPH**. Size from 100 to 200:

```
mml -cp allbc
saaii:sae=500,block=MVIPH,ni=200;
```

A.11.2. Cloning timer removal

The cloning process was afected by the existence of network penalties in the test environment. A **3 minutes** timer along with the lower network throughput caused by the network delay and packet loss, prevented the cloning process from succeeding. Therefore, the timer was removed to perform the tests conducted in this thesis project. The timer was located in the code that yields the *APZ_VM.ubuntu* binary, thus the code needed to be modified and recompiled in order to produce the new binary file.

Appendix B

Ericsson MSC-S BC prototype laboratory measurements

This appendix includes all the Ericsson MSC-S BC measurements compiled in the tests described in Chapter 4. First, tables containing the data collected when testing the external cluster are provided for the three different test cases: delay, packet loss, combination of delay and packet loss. Second, data tables are provided for the same three cases when testing the split cluster.

B.1. External cluster measurements

Table B.1 compiles the data collected when testing the Ericsson MSC-S BC external cluster with delay between the MSC-S blades and the rest of its elements. Table B.2 compiles the data collected when testing the Ericsson MSC-S BC external cluster with packet loss between the MSC-S blades and the rest of its elements. Finally, table B.3 compiles the data collected when testing the Ericsson MSC-S BC external cluster with both delay and packet loss between the MSC-S blades and the rest of its elements. The purpose of this measurements was to obtain the limit over which the performance of the system dropped inacceptably. A call fail rate of **5%** was considered as the limit of acceptance, at the behest of our employers.

Table B.1. Measurements compiled when an external Ericsson MSC-S BC cluster was serving traffic with different delays between the blades and the rest of the system.

Delay (ms)	**RTT (ms)**	**Call type**	**Successful calls**	**Calls in progress**	**Failed calls**	**Fail rate (%)**
0	0	GSM	40 874	74	0	0
		UMTS	40 528	76	0	0
10	20	GSM	25 160	110	0	0
		UMTS	25 167	115	0	0
20	40	GSM	25 022	134	0	0
		UMTS	25 024	143	0	0
30	60	GSM	25 002	159	0	0
		UMTS	25 005	171	0	0
40	80	GSM	25 013	183	0	0
		UMTS	25 017	197	0	0
50	100	GSM	25 011	209	0	0
		UMTS	25 003	226	0	0
60	120	GSM	25 003	234	0	0
		UMTS	24 999	253	0	0
70	140	GSM	25 002	261	0	0
		UMTS	25 001	282	0	0
80	160	GSM	25 035	283	0	0
		UMTS	25 032	305	0	0
82.5	165	GSM	50 024	295	0	0
		UMTS	50 020	316	0	0
83	166	GSM	25 175	294	0	0
		UMTS	25 178	314	0	0
83.5	167	GSM	25 721	292	1	0.004
		UMTS	25 708	322	1	0.004
84	168	GSM	203 399	1 075	41 581	16.973
		UMTS	200 020	1 139	44 918	18.339

Table B.2. Measurements compiled when an external Ericsson MSC-S BC cluster was serving traffic with different packet loss rates between the blades and the rest of the system.

Packet loss rate (%)	**Call type**	**Successful calls**	**Calls in progress**	**Failed calls**	**Fail rate (%)**
10^{-5}	GSM	25 323	74	0	0
	UMTS	25 008	77	0	0
10^{-4}	GSM	40 132	75	0	0
	UMTS	40 131	78	0	0
10^{-3}	GSM	26 300	76	0	0
	UMTS	26 301	77	0	0
10^{-2}	GSM	25 506	75	0	0
	UMTS	25 212	77	0	0
10^{-1}	GSM	25 002	75	0	0
	UMTS	25 000	77	0	0
1	GSM	25 010	81	0	0
	UMTS	25 010	82	0	0
2.5	GSM	28 045	89	0	0
	UMTS	28 051	92	0	0
5	GSM	25 007	103	0	0
	UMTS	25 003	108	0	0
7.5	GSM	25 046	127	0	0
	UMTS	25 047	138	0	0
10	GSM	30 074	156	0	0
	UMTS	30 094	160	0	0
11	GSM	50 073	172	0	0
	UMTS	50 069	185	0	0
11.5	GSM	50 059	182	0	0
	UMTS	50 078	193	0	0
12	GSM	50 100	197	142	0.283
	UMTS	50 062	211	214	0.338
12.5	GSM	200 292	526	50 107	20.011
	UMTS	200 991	436	49 403	19.730

Table B.3. Measurements compiled when an external Ericsson MSC-S BC cluster was serving traffic with different combinations of delay and packet loss between the blades and the rest of the system.

Delay (ms)	**RTT** (ms)	**Packet loss rate (%)**	**Call type**	**Successful calls**	**Calls in progress**	**Failed calls**	**Fail rate (%)**
50	100	0.1	GSM	20 412	212	0	0
			UMTS	20 059	233	0	0
		1	GSM	25 409	266	0	0
			UMTS	25 387	289	0	0
		1.5	GSM	30 113	328	0	0
			UMTS	30 102	352	0	0
		2	GSM	50 076	483	134	0.267
			UMTS	50 044	521	136	0.271
		2.5	GSM	30 893	981	2 186	6.608
			UMTS	30 491	989	2 618	7.907
60	120	0.1	GSM	25 388	233	0	0
			UMTS	25 386	252	0	0
		1	GSM	35 652	381	86	0.241
			UMTS	35 640	429	78	0.218
		1.5	GSM	31 240	944	4 069	11.524
			UMTS	30 331	996	4 949	14.028
70	140	0.1	GSM	25 124	257	0	0
			UMTS	25 121	281	0	0
		0.5	GSM	30 421	473	128	0.419
			UMTS	30 028	528	177	0.586
		1	GSM	25 462	1 080	3 264	11.363
			UMTS	25 008	1 174	3 646	12.724
80	160	0.1	GSM	101 805	293	9	0.009
			UMTS	101 816	308	12	0.012
		0.5	GSM	31 796	981	4 124	11.481
			UMTS	30 088	1 032	5 521	15.505
		1	GSM	28 379	1 169	15 989	36.037
			UMTS	25 003	1 269	19 284	43.543

B.2. Split cluster measurements

Table B.4 compiles the data collected when testing the Ericsson MSC-S BC geographically split cluster with delay between the MSC-S blade groups. Table B.5 compiles the data collected when testing the Ericsson MSC-S BC geographically split cluster with packet loss between the MSC-S blade groups. Finally, table B.6 compiles the data collected when testing the Ericsson MSC-S BC split cluster with both delay and packet loss between the MSC-S blade groups. The purpose of this measurements was to obtain the limit over which the performance of the system dropped inacceptably. A call fail rate of **5%** was considered as the limit of acceptance, at the behest of our employers.

Table B.4. Measurements compiled when an geographically split Ericsson MSC-S BC cluster was serving traffic with different delays between blade groups.

Delay (ms)	**RTT (ms)**	**Call type**	**Successful calls**	**Calls in progress**	**Failed calls**	**Fail rate (%)**
60	120	GSM	15 265	142	0	0
		UMTS	15 016	147	0	0
65	130	GSM	15 042	148	0	0
		UMTS	15 036	152	0	0
70	140	GSM	20 019	152	0	0
		UMTS	25 011	153	0	0
75	150	GSM	40 009	166	0	0
		UMTS	40 004	168	0	0
77.5	155	GSM	52 443	167	0	0
		UMTS	52 460	168	0	0
80	160	GSM	56 922	168	0	0
		UMTS	56 909	169	0	0
82.5	165	GSM	50 156	175	0	0
		UMTS	50 161	177	0	0
83	166	GSM	25 003	234	0	0
		UMTS	24 999	253	0	0
83.5	167	GSM	50 155	179	0	0
		UMTS	50 137	184	0	0
84	168	GSM	250030 035	342	24249	8.841
		UMTS	250 586	375	23715	8.646

Table B.5. Measurements compiled when a geographically split Ericsson MSC-S BC cluster was serving traffic with different packet loss rates between the blades.

Packet loss rate (%)	**Call type**	**Successful calls**	**Calls in progress**	**Failed calls**	**Fail rate (%)**
10^{-1}	GSM	10 130	77	0	0
	UMTS	10 128	78	0	0
1	GSM	20 010	80	0	0
	UMTS	20 003	82	0	0
2.5	GSM	21 144	84	0	0
	UMTS	21 146	85	0	0
5	GSM	30 171	92	0	0
	UMTS	31 175	94	0	0
7.5	GSM	51 856	98	11	0.021
	UMTS	51 866	104	0	0
10	GSM	50 833	112	73	0.143
	UMTS	50 915	117	0	0
11	GSM	75 559	122	213	0.281
	UMTS	75 772	131	0	0
11.5	GSM	30 109	153	3 385	10.106
	UMTS	30 174	162	3 328	9.934

Table B.6. Measurements compiled when an Split Ericsson MSC-S BC cluster was serving traffic with different combinations of delay and packet loss between the blades and the rest of the system.

Delay (ms)	**RTT** (ms)	**Packet loss rate (%)**	**Call type**	**Successful calls**	**Calls in progress**	**Failed calls**	**Fail rate (%)**
50	100	0.1	GSM	15 047	136	0	0
			UMTS	15 053	141	0	0
		1	GSM	30 025	161	0	0
			UMTS	30 021	166	0	0
		1.5	GSM	32 983	179	0	0
			UMTS	32 984	182	0	0
		2	GSM	30 027	188	0	0
			UMTS	30 014	196	0	0
		2.5	GSM	70 209	580	5 589	7.374
			UMTS	70 698	595	5 026	6.637
60	120	0.1	GSM	15 478	155	0	0
			UMTS	15 464	158	0	0
		1	GSM	30 053	186	13	0.043
			UMTS	30 054	194	0	0
		1.5	GSM	136 898	1006	89 512	39.535
			UMTS	145 080	1242	81 061	35.845
70	140	0.1	GSM	30 896	161	0	0
			UMTS	30 906	169	3	0.010
		0.5	GSM	50 077	184	2	0.004
			UMTS	50 094	195	0	0
		1	GSM	195 636	934	30 191	13.369
			UMTS	205 013	1 196	20 620	9.139
80	160	0.1	GSM	30 056	176	0	0
			UMTS	30 054	185	8	0.027
		0.5	GSM	110 027	231	948	0.854
			UMTS	110 887	246	108	0.097
		1	GSM	8 894	1091	17 224	65.947
			UMTS	9 422	1 214	16 523	63.685

Appendix C

Cloud Service Providers' survey

A survey analyzing the features of some of the most important Cloud services providers has been carried out.

In this survey, several features are analyzed. These features are the ones that are required to have a suitable platform to run the MSC application. The main restrictions in terms of hardware are both RAM memory and Ethernet connectivity. Sixteen gigabytes of RAM memory are needed to run the application, while a connection with gigabit technology is required. In relation with hard disk there is also a need, but every provider fulfills this requirement. There are other features that are important but not essential, like the type of offered servers (virtualized or physical) and the number of CPUs or cores for considering the processing power.

The survey has been realized in two steps. In the first step, a first selection is made and the main characteristics of providers' offer are written down. In the second step, six Cloud service providers within the initial are chosen and a research about their services is carried out.

C.1. First Cloud Service Providers' selection

In a first moment, sixteen companies were considered to perform this survey. After some research, a selection of six companies has been made. The six companies that fulfill our requirements in the best way are: *GoGrid*, *Rackspace*, *Savvis*, *IBM*, *Contegix*, *Amazon*, *Fujitsu* and *AT&T*.

Tables C.1 and C.2 show the studied features in this first selection.Two first features, support for managed hosting and data centers' location, are considered for a possible future use.

Table C.1. Cloud Service Providers' features

Provider	**Support for managed hosting**	**Data centres' location**	**CPUs / cores**	**RAM memory**	**HDD memory**	**Ethernet connectivity**	**Type of servers**
GoGrid *Cloud server*	Yes	San Francisco California Virginia	16 cores	16 GB maximum	800 GB	Gigabit Ethernet technology	Virtualized server
GoGrid *Dedicated server*	Yes	San Francisco California Virginia	8 cores	24 GB maximum	5 x 147 GB SAS RAID-5	Gigabit Ethernet technology	Physical server (non-virtualized)
Rackspace	Yes	America Europe Asia	4 vCPUs	15.82 GB (30 GB soon)	620 GB RAID-10	Gigabit Ethernet technology	Virtualized server
Savvis *Virtual Private Data Center*	No	Europe Asia North America	1 to 8 vCPUs	16 GB (32 GB soon)	500 GB maximum	Gigabit Ethernet technology	Virtualized server
Savvis *Open Cloud Compute*	Yes	Europe Asia North America	4 cores	16 GB	32 GB	Gigabit Ethernet technology	Virtualized server
Savvis *Dedicated Cloud Compute*	Yes	Europe Asia North America	1 to 4 cores	144 GB maximum	2 x 144 GB SAS RAID-1	Gigabit Ethernet technology	Physical server

Table C.2. Cloud Service Providers' features (cont.)

Provider	**Support for managed hosting**	**Data centres' location**	**CPUs / cores**	**RAM memory**	**HDD memory**	**Ethernet connectivity**	**Type of servers**
IBM	Yes	*Information not provided*	1 to 16 CPUs	58 GB maximum	2048 GB maximum	Gigabit Ethernet technology	Virtualized server
Contegix	Yes	St. Louis (USA)	1/4 to 8 CPUs	32 GB maximum	640 GB maximum	from 100 to 2000 GB of bandwidth	Physical dedicated and virtualized servers
Amazon *Elastic Compute Cloud (EC2) (I)*	Yes	Europe Asia North America	4 virtual cores	34.2 GB maximum	850 GB maximum	Gigabit Ethernet technology	Virtualized server
Amazon *Elastic Compute Cloud (EC2) (II)*	Yes	Europe Asia North America	2 x Intel Xeon X5570 quad-core	23 GB	1690 GB	10-Gigabit Ethernet technology	Virtualized server
Fujitsu	Yes	Australia United Kingdom Ireland	from 2 to 4 sockets machine	256 GB maximum	*Information not provided*	*Information not provided*	Virtualized server
AT&T	Yes	*Information not provided*	from 1 to 4 vCPUs	16 GB maximum	1 GB maximum	One virtual LAN per costumer	Virtualized server

C.2. Final Cloud Service Providers' selection

The final selection of Cloud service providers has been made taking into account the information obtained in tables C.1 and C.2. With this information, we have checked which providers fulfill in a best way our needs and we have obtained six candidates: *GoGrid*, *Rackspace*, *Savvis*, *IBM*, *Contegix* and *Amazon*.

After that, we have contacted directly with technical staff from these companies in order to obtain more information about the different services they offer and an extension to the characteristics of these services. The following sections show us the gathered information for that services.

C.2.1. GoGrid

- They are very excited with the idea of working with Ericsson.
- They are willing to look at custom options for us.
- Managed from a web-based control panel or API.
- Their data centers are in San Francisco, California and Ashburn, Virginia. In six months they will have one in Amsterdam and in a year one in Asia.

Their products are:

– **Cloud Servers:**

Table C.3. GoGrid Cloud Servers options. Adapted from [80]

CPU (cores)	RAM (GB)	Storage Space (GB)
0.5	0.5	25
1	1	50
2	2	100
4	4	200
8	8	400
16	16	800

  – A server image would need to be chosen and then we could configure it however we want. Many different server images are available, both Windows and Linux types, although these images can also be customized.
  – The RAM memory can be dynamically allocated.

With the Monthly Pre-Paid Plans, you get an allotment of Server RAM Hours to build your cloud hosting infrastructure. They also offer flexible pay-as-you-go pricing which allows you to pay only for the resources consumed on a per hour basis.

Table C.4. GoGrid payment plans [81]

Plan	**Monthly Cost**	**Server RAM Hours**	**Effective hourly Cost**	**Overage Rate**	**Effective Server Cost (per 0.5 GB RAM)**
Professional Cloud	$ 199.00	2 500	$ 0.08	$ 0.09	$ 29.90 / month
Business Cloud	$ 999.00	14 500	$ 0.07	$ 0.08	$ 25.55 / month
Corporate Cloud	$ 3 999.00	67 000	$ 0.06	$ 0.07	$ 21.90 / month
Enterprise Cloud	$ 9 999.00	200 000	$ 0.05	$ 0.05	$ 18.25 / month
Pay as you go	N/A	N/A	$ 0.19	$ 0.19	$ 69.35 / month

- **Dedicated Servers:**
  - Standard Dedicated Server

Table C.5. Standard Dedicated Server features [82]

Cores	**RAM (GB)**	**Storage**	**Price**
4	8	2 x 320 GB SATA RAID-1	$ 300 / month $ 3000 / year **

  - Advanced Dedicated Server

Table C.6. Advanced Dedicated Server features [82]

Cores	**RAM (GB)**	**Storage**	**Price**
8	12	2 x 500 GB SATA RAID-1	$ 400 / month $ 4000 / year **

  - Ultra Dedicated Server

Table C.7. Ultra Dedicated Server features [82]

Cores	**RAM (GB)**	**Storage**	**Price**
8	24	5 x 147 GB SAS RAID-5	$ 600 / month $ 6000 / year **

** Upfront payment for the year.
One year of minimum commitment. No setup fees.

- Included free with every account:
  - F5 hardware load balancing.
  - 24/7 premium support.

  - Unlimited inbound data transfer.
  - Licensing fees for Windows Server 2003 & 2008, and Red Hat Enterprise Linux *.
  - 10 GB Cloud Storage/month.
  - Up to 16 IP addresses**.

  * Additional fees will apply for Microsoft SQL Server 2005 and 2008 usage.
  ** 8 free IP addresses are assigned for use in each GoGrid Datacenter.

- Outbound traffic billing:

  With the Monthly Pre-Paid Plans, you get an allotment of outbound data transfer. They also offer flexible pay-as-you-go pricing which allows you to pay only for the resources consumed on a per GB basis.

Table C.8. GoGrid Outbound traffic billing [81]

Plan	**Monthly Cost**	**Outbound Transfer (GB)**	**Effective Unit Cost**	**Overage**
Pay-as-you-go	N/A	N/A	$ 0.29	N/A
Tranfer 500 GB	$ 99	500	$ 0.20	$ 0.29
Tranfer 3.6 TB	$ 499	3 600	$ 0.14	$ 0.20
Tranfer 20 TB	$ 1 999	20 000	$ 0.10	$ 0.14
Tranfer 57 TB	$ 3 999	57 000	$ 0.07	$ 0.07

Technical requirements:

- Multicast is supported within the GoGrid network but not to the Internet.
- They provide up to 16 public IP addresses and load balancing.
- Bandwidth 100% guaranteed (virtual servers burst beyond 100 Mbps closer to 1 Gbps, GoGrid has over 25 Gbps of bandwidth to the Internet and it could be easily increased if needed).
- They provide 24/7 service support.

Service quality agreements and privacy policies:

- Service Level Agreement (SLA) [83]
- Terms of service [84]

– Acceptable use policy [85]
– Privacy policy [86]
– Safe harbor policy [87]

C.2.2. Rackspace

- They support multicast in both Ethernet and UDP varieties.
- They offer Gigabit Ethernet, however the Cloud Servers will have a set amount of throughput depending on size (see Table C.9).
- They will be soon offering a 30 GB top end Cloud Server (likely available in October 2010), which will include more storage and throughput too.
- All Linux servers have 4 vCPUs.
- They offer the possibility of Cloud Servers with a Managed Service Level.
- No minimum commitments or contracts.
- Accessible via online control panel and open API.
- Fully customizable with root access.
- They offer the option of a dedicate server.
- Possibility of provisioning a Cloud Server that is IPv6-ready.

Their products are:

Table C.9. Rackspace servers

Memory Size	**Storage Size**	**Network Throughput (public/private)**
1 GB	40 GB	30 Mbps / 60 Mbps
2 GB	80 GB	60 Mbps / 120 Mbps
4 GB	160 GB	100 Mbps / 200 Mbps
8 GB	320 GB	150 Mbps / 300 Mbps
15.5 GB	620 GB	200 Mbps / 400 Mbps

Pricing:

Table C.10. Rackspace server pricing based in a Linux Server. Adapted from [88]

RAM size	Disk size	Hourly	Estimated Monthly*
256 MB	10 GB	1p	£7.30
512 MB	20 GB	2p	£14.60
1 024 MB	40 GB	4p	£29.20
2 048 MB	80 GB	8p	£58.40
4 096 MB	160 GB	16p	£116.80
8 192 MB	320 GB	32p	£233.60
15 872 MB	620 GB	64p	£467.20

* Monthly estimates based on 730 hours of usage

Table C.11. Rackspace bandwidth pricing based in a Linux Server [88]

Bandwidth IN	No Charge
Bandwidth OUT	12p / GB

Technical features:

– Storage is RAID-10 protected for superior performance and durability. If a host failure should occur, they bring it back and the data is preserved.

– No oversubscription. Reserved RAM and storage.

– Separate public and private network interfaces. Private bandwidth is free and provides inter-Cloud Server communication as well as access to Cloud Files. If we use a public IP, there is a cost for outgoing bandwidth only.

– Snapshots based server images - scheduled or on-demand. Useful for creating "gold" images or backups.

– Web-based console access. It is possible to access the server if there is a boot or network problem.

– Bootable rescue mode. File system access to repair a troubled Cloud Server.

– Combine Dedicated Servers and Cloud Servers to match your compute needs with the hybrid hosting option.

– Each Cloud Server comes with the simplicity of a dedicated and persistent public IP address (no NAT) with a second, private address for free. There is also low latency bandwidth between Cloud Servers.

Service quality agreements and privacy policies:

– SLA [89]

– Managed Hosting Terms and SLA [90]

– General Terms and Conditions [91]

– Privacy policy [92]

– Code of business conducts [93]

C.2.3. Savvis

- Their data centers are in London, Singapore, Tokyo and North America (14 data centers).

Their products are:

– **Savvis Symphony Virtual Private Data Center (VPDC)**

  The Balanced profile is the only one that would suit our needs.

  – CPU and memory: from 1 to 8 vCPUs (3 GHz) and 16 GB of memory.

  – Location selectivity: regional level transparency.

  – Access to the virtual server and operating environment (storage, network and security).

  – Virtual servers segregation

    - Each one can include a managed operating system, allocated virtual CPU and memory resources, and the Savvis Intelligent Agent (SIA) for monitoring.
    - Each one is partitioned from other instances to segregate each Customer's data from those of other customers (to prevent cross communication).
    - Customer environments stored on the VPDC Storage Area Network (SAN) are also segregated using dedicated Data stores / Logical Unit Numbers (LUNs) for individual VPDCs and their associated instances, allowing only the virtual server to which the allocation is assigned to access that storage.

- Supported OSs
  - Windows 2008 enterprise
  - Red Hat enterprise Linux 5.x
- Storage
  - Subject to additional charges.
  - The standard configuration of any Linux VPDC instance is deployed with a minimum 25 GB boot drive and a minimum 25 GB data drive.
  - Storage increments 50 GB / Max 500 GB per drive.
  - Information Lifecycle Management (ILM) storage: SAN Attached 2-Tier ILM (Fiber Channel & SATA Drives).
  - Data retention: 3 x Daily SNAP copies (3 Day Retention) and Daily Full Backup to disk (14 Day Retention).
- Supported network and IP configuration (pre-defined)
  - Multicast is not guaranteed, and should be previously tested.
  - Network topology: 2-tier.
  - Two dedicated VLANs: One public and one private.
  - Bandwidth: 100 Mbps enterprise-grade QoS (95^{th} percentile).
  - One virtual Network Interface Card (vNIC) per virtual server.
  - NAT 1-to-1 IP (optional).
  - Savvis management IP (one per virtual servers).
- Security
  - Configurable perimeter firewall (SPI) and 3-tier software-based server tier firewall (SPI).
  - Firewall logging: up to 30 days upon request.
  - Supported port range: 0 - 65,535.
  - Configurable server-to-server isolation.
  - Configurable tier-to-tier isolation.
  - Rule set number allowance: unlimited per tier.

  - Savvis will update VPDC virtual servers with all recommended and approved security patches, service packs and hot-fixed in order to maintain the overall integrity and performance of the virtual servers.
  - Savvis uses third-party anti-virus software in conjunction with centralized management tools to maintain AV policy control and regular signature file updates.
  - Anti-virus technology provides reasonable protection against malware; including viruses, spyware and trojans, but such technology cannot ensure the prevention of such malware.
  - Should disruption or changes occur due to malware, Savvis will use commercially reasonable efforts to remedy the situation as soon as possible after being notified of the problem, but Savvis will not be responsible for any damages due to worms, phishing attacks, root kits, trojan horses or other such malware, including infection of end-user devices or lost or corrupted data/messages.

– Server load balancer services are supported.

– Monitoring and support

  - 24/7 reactive support and agent based server monitoring.
  - Savvis does not provide any monitoring or management of Customer-provided software.
  - Savvis will proactively monitor the infrastructure and notify the Customer of an issue.

– User privileges

  - Savvis provides "sudo" access, which permits specifically authorized users to execute certain privileged commands without explicitly being root on the system.
  - Savvis maintains full root or administrator access on VPDC virtual servers.
  - It is Savvis' security policy that root logins be limited to console access only and any such access is logged.
  - Customer may request higher level access and Savvis will grant such access subject to Customer acknowledging in writing that any Services to which Customer is granted root or any similar level of access shall be excluded from any otherwise applicable SLA and Savvis shall have no responsibility whatsoever to the extent the

applicable Service incurs any incident, outage or other service issues caused by any act or omission of Customer.

– **Savvis Dedicated Cloud Compute**

Dedicated server hardware is provided, it uses Utility Storage services for storage and leverages VMware to allow multiple OSs to operate in parallel.

– Hardware options

DL-series

Table C.12. Savvis DL-series servers [94]

HP Server	No. of CPUs	Processor type	RAM	Disk drives	Network interface
DL380 G6	2	2.93 GHz Quad-Core Intel Xeon X5570 or 2.53 GHz Quad-Core Intel Xeon E5440 or 2.26 GHz Quad-Core Intel Xeon E5520 CPUs (Nehalem)	From 4 to 144 GB	2x 144 GB SAS HDDs in RAID-1 configuration	Dual 4 Gbps Fiber Channel interfaces (additional NICs may be added)
DL385 G5p	2	2.30 GHz Quad-Core AMD Opteron 2376HE CPU (Shanghai)	From 4 to 128 GB		
DL580 G5	4	2.93 GHz Quad-Core Intel Xeon X7350 or 1.60 GHz Quad-Core Intel Xeon E7310 CPUs (Tigerton)	From 4 to 144 GB		
DL585 G5	4	2.30 GHz Quad-Core AMD Opteron 8356 CPU (Barcelona)	From 4 to 144 GB		

BL-series

Table C.13. Savvis BL-series servers [94]

HP Server	**No. of CPUs**	**Processor type**	**RAM**	**Disk drives**	**Network interface**
BL460c	2	2.93 GHz Quad-Core Intel Xeon X5570 or 2.53 GHz Quad-Core Intel Xeon E5440 or 2.26 GHz Quad-Core Intel Xeon E5520 CPUs (Nehalem)	From 4 to 144 GB	2x 144 GB SAS HDDs in RAID-1 configuration	Dual 4 Gbps Fiber Channel interfaces (additional NICs may be added)
BL465c	2	2.30 GHz Quad-Core AMD Opteron 2356 CPU (Shanghai)	From 4 to 64 GB		
BL680c	4	2.4 GHz Quad-Core Intel Xeon X7340 or 1.60 GHz Quad-Core Intel Xeon E7310 CPUs (Tigerton)	From 4 to 128 GB		
BL685c	4	2.30 GHz Quad-Core AMD Opteron 8356 CPU (Barcelona)	From 4 to 128 GB		

– Management
  - Savvis provide OS licensing, installation, management and monitoring of Savvis standardized OS builds.
  - Customer approves patches and upgrades that may impact their application.

– Number of instances
  - Each Savvis Dedicated Cloud Compute Node requires at least one Savvis Cloud Compute Instance or managed operating system.

– Storage
  - Storage for the Savvis Dedicated Cloud Compute Instances is available via the Utility Storage service, which must be purchased separately to provide storage for Savvi Dedicated Cloud Compute Instances.

  – Network

    • Multicast is supported.
    • We think all network requirements will be accomplished since it is one of their best services.

  – Failover

    • For customers that purchase more than one Savvis Dedicated Cloud Computer Node in a failover group, should a hardware failover occur with a Savvis Dedicated Cloud Compute Node, the Savvis Dedicated Cloud Compute Instances will automatically be restarted and distributed across the other available nodes within the failover group.
    • Failover groups can be configured with up to sixteen Savvis Dedicated Cloud Compute Nodes.
    • Once the failed node is restored and confirmed stable by Savvis Operations, instances may be redistributed across the nodes.

  – Dedicated Cloud Compute Dynamic Instance Balancing is supported.

– **Savvis Open Cloud Compute**

It is a managed server service that includes use of virtual machine instances in a multi-tenant computing environment.

Savvis provides the infrastructure including space and power, compute resources, virtualization operating system license, instance operating system license, hosting area network connectivity, use of Utility Storage, management and monitoring of the server instance and infrastructure, and use of Savvis Station Portal for management and automated instance provisioning.

  – Instance types

Table C.14. Savvis Open Cloud Compute Instances types [94]

Number of CPU cores	RAM	Processor type	Storage
½	2 GB	3.0 GHz Intel Xeon 5000 series processors	32 GB
1	4 GB		
1	6 GB		
2	8 GB		
2	12 GB		
4	16 GB		

- Group of servers and resource allocation
  - Each instance will reside within a group of servers that are networked together into a source group with VMware® High Availability and Dynamic Resource Balancing capabilities.
  - Each instance is allocated its own CPU and RAM resources and will not compete with other instances for access to these resources under normal operating conditions. A normal operating environment is defined as an environment in which all servers within a cluster of five servers are fully operational without a system or component failure.
  - Additional CPU and RAM resources are set aside in each server and each cluster to manage VMware overhead and provide additional resources in the event of a node failure.
- Instances segregation
  - Each instance is partitioned from other instances to segregate each customer's data from those of other customers. This partitioning prevents cross communication from one costumer's instance(s) to those of another customer.
  - Savvis further segregates customer environments stored in Savvis Utility Storage SAN by utilizing dedicated volumes/LUNs for individual instances, allowing only the instance to which the allocation is assigned to access that storage.
- Failover service as in dedicated cloud servers.
- Cloud storage service as in dedicated cloud servers.
- Network
  - Multicast is supported.
  - We think all network requirements will be accomplished since it is one of their best services.
- Supported OSs
  - Microsoft Windows Enterprise 2003 and 2008.
  - Red Hat Enterprise Linux.
  - Sun Solaris 10.
- Same privileges as in VPDC.

C.2.4. IBM

- Instance types

  IBM only offers virtual machines instances.

Table C.15. IBM Instances types [95]

Type	*Copper*	*Bronze*	*Silver*	*Gold*	*Platinum*	*Iridium**
Number of vCPUs with 1.25 GHz	2	2	4	8	16	16
Virtual memory (GB)	4	4	8	16	16	58
Instance storage (GB)	60	850	1 024	1 024	2 048	Unknown, but surely enough

*Iridium type requires a determined level of commitment.

All the pricing details are in [96].

– Supported OSs

- Red Hat Linux OS.
- Novell SUSE Linux OS.
- Windows Server.

– Access

- Full administrative access to the guest operating system (root in Linux) which includes manipulating network capabilities, and installing additional software.
- There are also ways to manipulate the infrastructure firewall for more restrictive settings.

– Network

- IBM Smart Cloud Enterprise employs multi-carrier resilient network connections to the Internet in all of our locations, utilizing redundant physical connections to a range of Wide Area Network providers.
- The available bandwidth is shared between all customer instances.
- Different IP configuration options:
  - *Dynamic public static IP*: A public static IP range. Each data center has its own IP range. IP addresses are assigned at provisioning time, and released back to the "global" pool when VM Instances are deleted.

  - *Reserved public static IP*: Users can reserve IP addresses from the data centers public IP range. This way you can ensure to have the same IP address for clients independent of VM Instance life cycle.
  - *Own IP address range*: Through the use of hardware VPN, you can choose at provision time to provision VMs as an extension to your own LAN – meaning that they will have the same IP address as your own data center, public or private. The instances will be, in this case, only accessible through the VPN tunnel.
  - Unfortunately Broadcast and Multicast are disabled in the network. However, it is possible to deploy an overlay network which will allow broadcast and multicast traffic between the nodes (VPN).

- They have an SLA commitment of 99.5% availability on individual instances.

C.2.5. Contegix

- Their data centers are located in the Bandwidth Exchange Building on Walnut Avenue in St. Louis (USA).

- Their facility is staffed with Tier 3 Support Engineers 24x7x365. They guarantee all support request will be responded to within thirty minutes and the average response is <u>four</u> minutes.

- Many of their engineers are Dell certified technicians. In addition, they maintain ample stock of spare parts for Dell servers including hard drives, memory, etc.

- Their network infrastructure is fully meshed, running redundant Juniper routers and Foundry BigIron core switches. They have five Tier 1 providers including Sprint, Level, MCI, XO and WilTel running BGP4.

- All of their connections to their providers in their managed network are "On net" meaning they connect directly to the Internet avoiding local loops and local connections.

- All power supplied to the data center is clean and constant coming from the redundant Uninterruptible Power Supply (UPS) (Alternating Current) or battery plants (Direct Current). The PowerWare UPS systems run in a redundant configuration to maximize reliability.

<u>Their product is:</u>

- **Managed Dedicated Server**

  - Dell PowerEdge Server

- 2 x Intel Quad-Core Xeon with 4 MB Cache and 1066 MHz FSB
- 24.0 GB DDR2 RAM
- 2 x 146 GB Serial Attached SCSI Har Drives (Hot Swap)
- PERC RAID Card for RAID 1/5/10 Capable
- Dual 10/100/1000 NICs
- RedHat Enterprise Linux ES
- Beyond Managed Hosting™Service with Hyperic HQ Monitoring
- Managed Backups - Weekly
  - Retention of 28 Days Minimum
  - Weekly Schedule
  - 50 GB

– **Bandwidth**

- 500 GB Bandwidth Included
  - 100% Network Uptime SLA
  - DDoS Mitigation / Prevention
  - Intelligent Routed Network
  - Burstable to 100 Mbps

– **Pricing**

- **Monthly Price**
  - $1,339.95 with 1-year commitment
- **Installation Fee**
  - $699.00 WAIVED with 2-year agreement
- **Additional Monthly Bandwidth**
  - Price per GB ($3.00/GB)

Service quality agreements and privacy policies:

– SLAs [97]

– Acceptable Use Policy [98]

– Privacy Policy [99]

– Support that never sleeps [100]

C.2.6. Amazon

- Their data centers are in Europe (Stockholm, Paris, London, Amsterdam, Dublin and Frankfurt), Asia (Hong Kong, Singapur and Tokio) and North America (ten different data centers).
- Amazon EC2 provides virtualized server instances. While some resources like CPU, memory and instance storage are dedicated to a particular instance, other resources like the network and the disk subsystem are shared among instances.
- If it is wanted that all components are dedicated to one account, it is needed to use Dedicated Instances, which means that an isolated hardware is dedicated to a single customer.
- They offer the choice of running a cluster in an isolated server.
- They have said that multicast is not supported within the VPC.

Their products are:

– **Elastic Compute Cloud (EC2)**

Table C.16. Amazon EC2 server instances [101]

ECU (EC2 Compute Unit)*	**RAM (GB)**	**Storage Space (GB)**
2 virtual cores with 3.25 ECUs each (6.5 ECUs) *(High-Memory Extra Large Instance*	17.1	420
4 virtual cores with 3.25 ECUs each (13 ECUs) *(High-Memory Double Extra Large Instance*	34.2	850
8 virtual cores with 3.25 ECUs each (26 ECUs) *(High-Memory Quadruple Extra Large Instance*	68.4	1 690
33.5 EC2 Compute Units (2 x Intel Xeon X5570, quad-core "Nehalem" architecture) *(Cluster Compute Quadruple Extra Large Instance*	23	1 690

* EC2 Compute Unit (ECU) - One EC2 Compute Unit provides the equivalent CPU capacity of a 1.0-1.2 GHz 2007 Opteron or 2007 Xeon processor.

- Pay only for what is used.
- There is no minimum fee.

– **Dedicated Instances**

- It lets the customer take full advantage of the benefits of Amazon Virtual Private Cloud (VPC) and the Amazon Web Services (AWS) Cloud - on-demand elastic provisioning, pay only for what is used, and a private, isolated virtual network, all while ensuring that the Amazon EC2 compute instances will be isolated at the hardware level.
- Pay only for what is used with no long-term commitments. Dedicated Instance pricing has two components: (1) an hourly per instance usage fee and (2) a dedicated per region fee (note that this payment is done once per hour regardless of how many Dedicated Instances are running).
  – Dedicated Per Region Fee
    $10 per hour - An additional fee is charged once per hour in which at least one Dedicate Instance of any type is running in a Region.
  – Dedicated On-Demand Instances

Table C.17. Amazon Dedicated instances' prices. Adapted from [102]

High-Memory Instances	**Linux / Unix**	**SUSE Linux Enterprise**
Extra Large	$ 0.71 per hour	$ 0.83 per hour
Double Extra Large	$ 1.42 per hour	$ 1.54 per hour
Quadruple Extra Large	$ 2.84 per hour	$ 2.95 per hour

Technical requirements:

– Multicast is not supported within the VPC.

– Bandwidth 99.95% guaranteed (virtual servers burst beyond 100 Mbps closer to 10 Gbps).

– They provide 24/7 service support.

– The Cluster Compute instance family permits to use lower latency and high-bandwidth connectivity (10 Gbps) between the instances using Placement group and to use HVM virtualization. HVM virtualization lets the guest VM run as though it is on a native hardware platform with the exception that it still uses paravirtual (PV) network and storage drivers for improved performance.

– All their services are unmanaged, full root access to the virtual server is given.

Service quality agreements and privacy policies:

– SLA [103]

– Terms of service [104]

- Terms of use [105]
- Privacy Policy [106]
- AWS Security White Paper (Instance Isolation and Network Security) [107]

Appendix D

Selected Provider Benchmarking scripts

This appendix includes all the *awk* scripts that are used during the tests performed in the selected provider (**Rackspace**) that are explained in Section 5.5.1. All these scripts have the function of acting as a parser for the output obtained from the tests, in order to gather the information needed.

D.1. *awk_udp*

This script was used as parser for the *iperf* output to gather the results in another files called "*udp_BW.txt*", "*udp_jitter.txt*", and "*udp_lost.txt*". This script is written by the authors of this project.

```
BEGIN  { found = 0; t="0"; j="0"; l="0" }

$10 == "ms" { found = 1; t = $7; j = $9; l = $12; }

END { if (found == 1) {
            print t >> "udp_BW.txt";
            print j >> "udp_jitter.txt";
            split(l, parts, ")");
            split(parts[1], parts2, "(");
            print parts2[2] >> "udp_lost.txt";
        }
}
```

Listing D.1: Awk parser to gather information from UDP test's output.

D.2. *awk_BW*

This script was used as parser for the *iperf* output to gather the results in another file called "*bandwidth.txt*". This script is written by the authors of this project.

```
BEGIN  { found = 0; t="0" }

$9 == "Mbits/sec" { found = 1; t = $8; }

END { if (found == 1) {
          print t >> "bandwidth.txt";
      }
}
```

Listing D.2: Awk parser to gather information from TCP test's output.

D.3. *awk_average_RTT*

This script was used as parser for the *PING* output to gather the results in another file called "*RTT.txt*". This script is written by the authors of this project.

```
BEGIN  { found = 0; t="0" }

$1 == "rtt" { found = 1; t = $4; }

END { if (found == 1) {
          split(t, parts, "/");
          print parts[2] >> "RTT.txt";
      }
}
```

Listing D.3: Awk parser to gather information about the average RTT.

D.4. *awk_loss*

This script was used as parser for the *PING* output to gather the results in another file called "*packet_loss.txt*". This script is written by the authors of this project.

```
BEGIN  { found = 0; t="0" }

$2 == "packets" { found = 1; t = $6; }

END { if (found == 1) {
          split(t, parts, "%");
          print parts[1] >> "packet_loss.txt";
      }
}
```

Listing D.4: Awk parser to gather information about packet loss.

D.5. *awk_MAX_RTT*

This script was used as parser for the *PING* output to gather the results in another file called "*MAX_delay.txt*". This script is written by the authors of this project.

```
BEGIN  { found = 0; t="0" }

$1 == "rtt" { found = 1; t = $4; }

END { if (found == 1) {
          split(t, parts, "/");
          print parts[3] >> "MAX_delay.txt";
      }
}
```

Listing D.5: Awk parser to gather information about the maximum RTT.

D.6. *awk_MIN_RTT*

This script was used as parser for the *PING* output to gather the results in another file called "*MIN_delay.txt*". This script is written by the authors of this project.

```
BEGIN  { found = 0; t="0" }

$1 == "rtt" { found = 1; t = $4; }

END { if (found == 1) {
          split(t, parts, "/");
          print parts[1] >> "MIN_delay.txt";
      }
}
```

Listing D.6: Awk parser to gather information about the minimum RTT.

D.7. *awk_jitter*

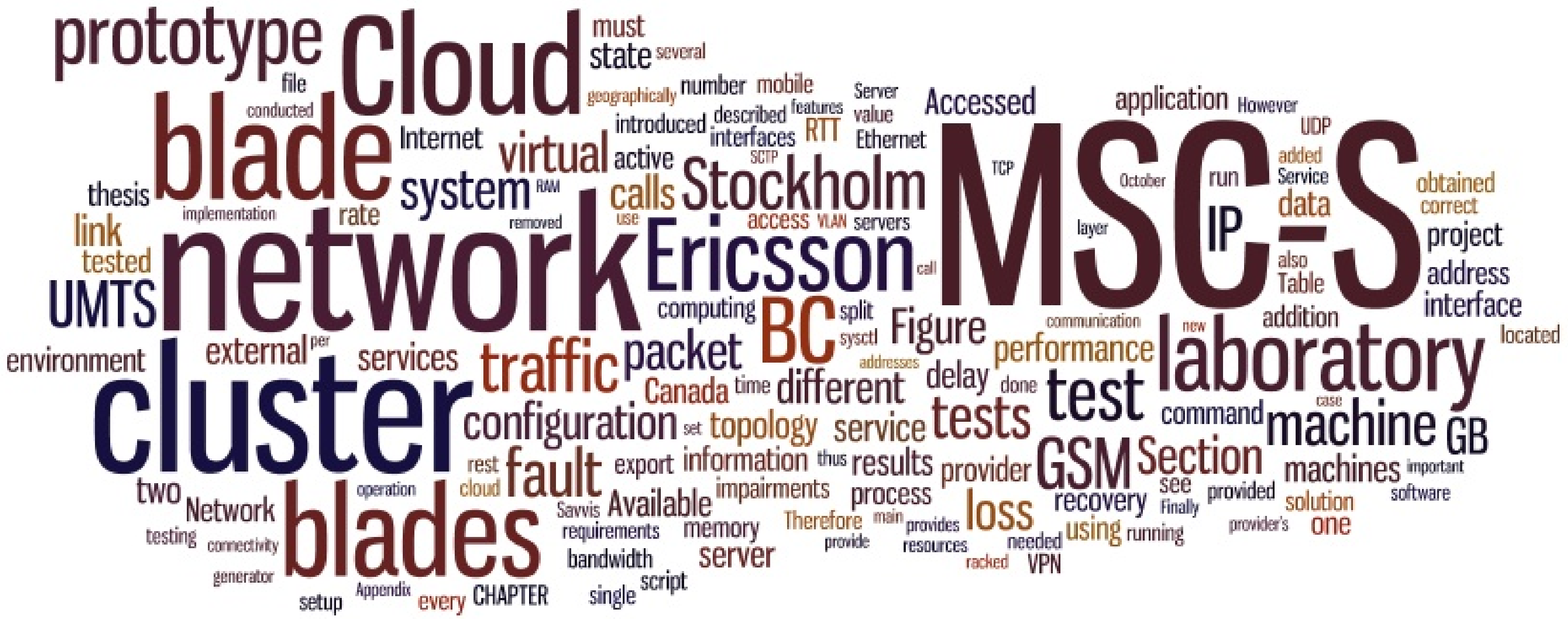
This script was used as parser for the *PING* output to gather the results in another file called "*jitter.txt*". This script is written by the authors of this project.

```
BEGIN  { found = 0; t="0" }

$1 == "rtt" { found = 1; t = $4; }

END { if (found == 1) {
          split(t, parts, "/");
          print parts[4] >> "jitter.txt";
      }
}
```

Listing D.7: Awk parser to gather information about jitter value.


MSC-S
network
cluster
blade
blades
Cloud
prototype
Ericsson
laboratory
Stockholm
BC
traffic
test
tests
machine
GSM
fault
configuration
packet
system
virtual
UMTS
Section
IP
different
service
calls
loss
Figure
topology
Accessed
application
results
machines
information
GB
performance
Available
external
services
environment
thesis
link
tested
two
Network
Internet
rate
computing
delay
provider
data
command
process
impairments
recovery
server
memory
using
project
address
interface
addition
obtained
correct
located
Canada
active
introduced
number
mobile
must
state
several
file
conducted
geographically
described
features
interfaces
RTT
Server
value
Ethernet
servers
access
VLAN
SCTP
RAM
removed
use
implementation
TCP
October
layer
run
UDP
However
added
Service
also
Table
call
split
sysctl
communication
new
addresses
time
done
case
set
export
rest
cloud
thus
see
provided
solution
important
software
operation
Savvis
requirements
Therefore
main
provides
Finally
provider's
one
testing
connectivity
provide
resources
needed
running
racked
VPN
bandwidth
script
single
generator
setup
Appendix
every
CHAPTER
per

TRITA-ICT-EX-2012:54