Performance Evaluation and Elastic Scaling of an IP Multimedia Subsystem Implemented in a Cloud

MUHAMMAD UMAIR

KTH Information and Communication Technology

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

Performance Evaluation and Elastic Scaling of an IP Multimedia Subsystem Implemented in a Cloud

MUHAMMAD UMAIR
<mumair@kth.se>

Master of Science Thesis
Project performed at Fraunhofer FOKUS Competence Center NGNI, Berlin, Germany

School of Information and Communication Technologies
KTH Royal Institute of Technology
Stockholm, Sweden

July 13, 2013

Examiner: Prof. Gerald Q. Maguire Jr.

Abstract

The IP Multimedia Subsystem (IMS) framework is a Next Generation Network (NGN) technology which enables telecommunication operators to provide multimedia services over fixed and mobile networks. All of the IMS infrastructure protocols work over IP which makes IMS easy to deploy on a cloud platform. The purpose of this thesis is to analysis a novel technique of "cloudifying" the OpenIMS core infrastructure. The primary goal of running OpenIMS in the cloud is to enable a highly available and horizontally scalable Home Subscriber Server (HSS). The resulting database should offer high availability, and high scalability.

The prototype developed in this thesis project demonstrates a virtualized OpenIMS core with an integrated horizontal scalable HSS. Functional and performance measurements of the system under test (i.e. the virtualized OpenIMS core with horizontally scalable HSS) were conducted. The results of this testing include an analysis of benchmarking scenarios, the CPU utilization, and the available memory of the virtual machines. Based on these results we conclude that it is both feasible and desirable to deploy the OpenIMS core in a cloud.

Sammanfattning

IP Multimedia Subsystem (IMS) ramverk är ett Next Generation Network (NGN) teknik som möjliggör teleoperatörer att erbjuda multimediatjänster via fasta och mobila nät. Alla IMS infrastruktur protokollen fungera över IP som gör IMS lätt att distribuera på ett moln plattform. Syftet med denna uppsats är att analysera en ny teknik för “cloudifying” den OpenIMS kärninfrastrukturen. Det primära målet med att köra OpenIMS i molnet är att möjliggöra en hög tillgänglighet och horisontellt skalbara Server Home Subscriber (HSS). Den resulterande databasen bör erbjuda hög tillgänglighet och hög skalbarhet.

Prototypen utvecklas i detta examensarbete visar en virtualiserad OpenIMS kärna med en integrerad horisontell skalbar HSS. Funktionella och prestanda mätningar av systemet under test (dvs. virtualiserade OpenIMS kärnan med horisontellt skalbara HSS) genomfördes. Resultaten av detta test inkluderar en analys av benchmarking scenarier, CPU-användning, och tillgängligt minne för de virtuella maskinerna. Baserat på dessa resultat drar vi slutsatsen att det är både möjligt och önskvärt att distribuera OpenIMS kärnan i ett moln.

Acknowledgements

I would like to express my deepest gratitude to Prof. Gerald Q. Maguire Jr. for accepting to be my examiner, his wonderful guidance, inspiration, and valuable feedback throughout this master's thesis.

My sincere thanks to Fraunhofer Institute FOKUS, Berlin Germany for offering this master thesis opportunity.

Finally, my deepest love for my family, friends, and especially my deepest thanks to my brother for supporting expenditures during my master's degree.

Contents

List of Figures

List of Tables

List of Listings

List of Acronyms and Abbreviations

ANI	Application-Network Interface
AS	Application Server
AaaS	Application as a Service
API	Application Programming Interface
AuC	Authentication Center
CRM	Customer Relationships Management
CaaS	Computing as a Service
CSCFs	Call/Session Control Functions
CDP	CDiameterPeer
CPS	Calls per Second
CAPEX	Capital Expenditures
DaaS	Database as a Service
DIL	Diameter Interface Layer
DAL	Data Access Layer
DB	Database
ETSI	European Telecommunication Standard Institute
FQDN	Fully Qualified Domain Name
FHoSS	FOKUS Home Subscriber Server
GSM	Global System for Mobile Communication
GUI	Graphical User Interface
HSS	Home Subscriber Server
HLR	Home Location Register
HaaS	Hardware as a Service
HTML	HyperText Markup Language
IP	Internet Protocol
IMS	IP Multimedia Subsystem
IMSaaS	IP Multimedia Subsystem as a Service
I-CSCF	Interrogating Call Secession Control Function
IaaS	Infrastructure as a Service
ISC	IMS Service Control
IHS	Inadequately Handled Scenarios
IO	Input/Output
JDBC	Java Database Connectivity
JPA	Java Persistence API

KPIs	Key Performance Indicators
LTE	Long Term Evolution
LB	Load Balancer
MME	Mobility Management Entity
MHT	Microsoft Hypertext Archive
MB	MegaBytes
NGN	Next Generation Network
NNI	Network-Network Interface
NAI	Network Access Identifier
NIST	National Institute of Standards and Technology
OPEX	Operating Costs
P-CSCF	Proxy Call Secession Control Function
PDF	Policy Decision Function
PaaS	Platform as a Service
QoS	Quality of Service
RR	Round Robin
RRD	Registration Request Delay
RAN	Radio Access Network
SIP	Session Initiation Protocol
S-CSCF	Serving Call Secession Control Function
SLF	Subscriber Location Function
SUT	System Under Test
SaaS	Software as a Service
SaaS	Storage as a Service
SER	SIP Express Router
SMS	Short Message Service
SAPS	Scenarios Attempts Per Second
SDN	Software Defined Network
TTCN-3	Testing and Test Control Notation Version 3
URL	Uniform Resource Locator
UNI	User-Network Interface
UMTS	Universal Mobile Telecommunication Systems
VMM	Virtual Machine Manager
VoIP	Voice over IP
VRRP	Virtual Router Redundancy Protocol
XaaS	Every thing as a Service
3GPP	3rd Generation Partnership Project

Chapter 1

Introduction

This chapter presents the motivation, the problem description, and the goals of this thesis project. This master's thesis project took place at the Fraunhofer FOKUS Competence Center NGNI, Berlin, Germany. We will refer to this center in the remainder of the thesis simply as FOKUS. The overall structure of this thesis is presented in Section 1.2.

1.1 Motivation

Mobile access to services such as telephony, Internet access, and short message service (SMS) is increasing day by day, and today each mobile user expects the same quality of experience as the end user experience when a using wired network device. The large scale deployment of service resources involves a huge capital expenditure in order to fulfill the service user's demands, and telecommunication (telecom) operators have major concerns about sustaining a large scale service deployment. Therefore, researchers have proposed a novel solution, which not only reduces the initial cost of deployment but also provides an easy way to scale the service *without* any expensive modification to the infrastructure. Such a creative solution has been shown using cloud computing in a recent master's thesis by Isaac Albarrán and Manuel Parras [1].

Cloud computing is a emerging Internet driven trend. Cloud computing is based on virtualization technology and together with high bandwidth networks provides a pool of computing resources and potentially a very large storage capacity. Cloud computing achieves resource sharing for various types of services, such as Infrastructure as a service (IaaS), Platform as a service (PaaS), Software as a service (SaaS), etc. [2]

Creating the traditional telecom infrastructure involved a huge investment of money, but today operators are facing a gradual decline in profits for strictly telecom businesses. For this reason telecom operators want to offer a variety of broadband-

based value-added services which could be profitable together with a change in the traditional telecom operator model. Xu Lei, et al. [2], suggested that telecom operators integrate cloud computing services with their existing telecom services to create a new business. A. Roozbeh [3] have proposed that the increasing interest in carrier clouds, also called embedded clouds, which integrate the computing and storage resources in a distributed fashion with the network. G. Caryer, et al. [4], proposed a solution based upon combining grid/cloud technology services to deploy Next Generation Network (NGN) functionality. Jiann-Liang Chen, et al. [5], presented a solution for offering high quality multimedia applications by combing the IP Multimedia Subsystem (IMS) architecture with a cloud computing infrastructure. Many researchers think that an IMS infrastructure can be deployed on a large scale together with next-generation networks to fulfill the increasing demands for Internet based services. Unfortunately, according to Makaya, et al. the IMS core network components do *not* scale easily due to its complex nature [6]. This thesis investigates a solution of how to combine the IMS core architecture with a cloud computing infrastructure. P. Bellavista, et al. [7], have proposed a novel cloud brokering system for IMS based services. They implemented a solution that guarantees QoS for cloud based IMS service by dynamically scaling-up/scaling-down server computing resources, and enabling services to migrate across multiple cloud platforms.

A key part of this thesis project is to analyze a solution for the deployment of an IMS core network infrastructure in a cloud computing architecture. X. Zhiqun, et al. [8], have proposed the virtualization of the IMS core network and radio access network (RAN) as an application of telco cloud. Tinniam V. Ganesh [9] proposed a cloud architecture for an IMS infrastructure. In this proposal the IMS core network components would be deployed on distributed cloud platforms either in public or private clouds. The high level cloud-based architecture for such a IMS core network framework is shown in Figure 1.1.

In an IMS network, the Home Subscriber Server (HSS) is a critical component as it supports both user mobility and call control. Since the growth in the number of subscribers and the data associated with these subscribers is increasing rapidly, sufficient HSS data storage capacity is critical to an IMS system [10]. The HSS data storage is not only critical in terms of having sufficient capacity, but also in terms of having high availability. If a single HSS database instance can not handle a sufficiently large number of simultaneously requests, then the delay in response time would increase. As this delay due to the limited HSS database performance may cause incoming requests to the HSS to be lost, this means that calls might not be able to be set up or calls might be unable to be modified to support the mobility of users and their terminals. In this thesis project, we investigate a solution for horizontal scalability of the database underlying the HSS, in order to scale the number of database instances. The goals of this scaling are to enhance both database availability and performance. All of these goals are to be met while avoiding the need

for large capital expenditures (CAPEX) or causing high operating costs (OPEX). Cloud computing based telecom services will be significantly reduced the cost of CAPEX and OPEX [8]. Cloud computing naturally fits these requirements since the cloud operator pays the CAPEX and the telecom operator need only pay OPEX proportional to their needs to support their customer's current requirements.

Considerable efforts have been made to define an accurate load generator for benchmarking the proposed solution. In the context of IMS benchmarking, where millions of subscribers need to be supported scaling of the load generator itself required. The benchmarking system needs to evaluate the system's performance when the number of active subscribers and the number of calls per second (CPS) increases to very high values. George Din [12] presented a benchmark specification implementation for a IMS system based on the Testing and Test Control Notation Version 3 (TTCN-3) language from the ETSI. Niranjanan Kalaichejvan [13] in his master thesis designed a S6a load application based on a distributed scheduling architecture using Ericsson's TITANSim [11] framework.[1] S6a is a Diameter interface between a Mobility Management Entity (MME) and the HSS in long term evolution (LTE) networks. Kalaichejvan's motivation was to design a highly scalable load generator tool for testing a HSS as the system under test (SUT).

Dirk et al. [14], evaluated the performance of the core components of an IMS network using an open source implementation of the IMS/NGN Performance Benchmark Specification (i.e., IMS Bench SIPp [51]). In their evaluation of the FOKUS OpenIMS core [42] performance, they configured all the IMS core components on a single machine. In contrast in this thesis project, we will evaluate the performance of a *virtualized* OpenIMS core with an integrated horizontally scalable HSS database by using this same IMS Bench SIPp benchmarking tool.

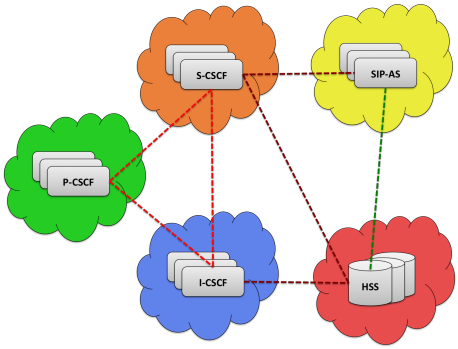


Figure 1.1. High Level Cloud realization of an IMS Core Network Architecture

[1] TITANSim is a TTCN-3 tool

1.2 Thesis outline

Chapter 1 described the motivation, problem description, and the goals of the thesis project. Chapter 2 and Chapter 3 provides background information to ease the reader's understanding of the rest of this thesis. Chapter 2 gives an overview of NGN and IMS. Chapter 3 summarizes the cloud computing paradigm. Chapter 4 describes the open source tools and technologies that were used for the test environment. Chapter 5 specifies the core architecture of the system which was designed as a solution to the thesis problem. Chapter 6 presents the evaluation and performance testing of this proposed solution as the SUT. Finally, Chapter 7 concludes the thesis and suggests directions for future research.

Chapter 2

Overview of Next Generation Networks and the IP Multimedia Subsystem

This chapter presents an overview of a next generation network's infrastructure in terms of both its architecture and the underlaying technologies.

2.1 Next Generation Network (NGN) Architecture

ITU-T has defined the term NGN as: "Next Generation Network is a packet-based network able to provide telecommunication services and able to make use of multiple broadband, QoS-enabled transport technologies and in which service-related functions are independent from underlying transport-related technologies". Also according to the ITU-T, such a NGN provides unrestricted access to networks and services based on user demand; and supports generalized mobility, which permits reliable and ubiquitous delivery of services to users [17].

The ITU-T has identified the following fundamental characteristics of NGN [17]:

- Packet based transfer
- Partition of control functions among bearer capabilities, call/session, and application/service
- Support for decoupling of service from transport and provisioning of open interfaces
- Provides a variety of services, applications, and service-based mechanisms including real time, streaming, and non-real time multimedia services
- Provides broadband capabilities with end-to-end QoS
- Interworking via open interfaces with legacy networks
- Generalized mobility

- Unlimited access by users to different service providers
- Consolidates service characteristics for the same service as identified by the user
- Convergence of services between fixed and mobile networks
- Transparency of service-related functions from underlying transport technologies
- Provision of multiple last mile technologies
- Acquiescent with all regulatory requirements, for example regulations concerning emergency communications, security, privacy, lawful interception, etc.

The NGN architecture as described by Shuji Esaki, Akira Kurokawa, and Kimihide Matsumoto in [18] is shown in Figure 2.1. The transport functions are responsible for transferring multimedia streams over an IP network. The transport control functions module assigns IP address and performs authentication. Service control functions and application support functions & service support functions enable support for functionality such as presence management. [18]

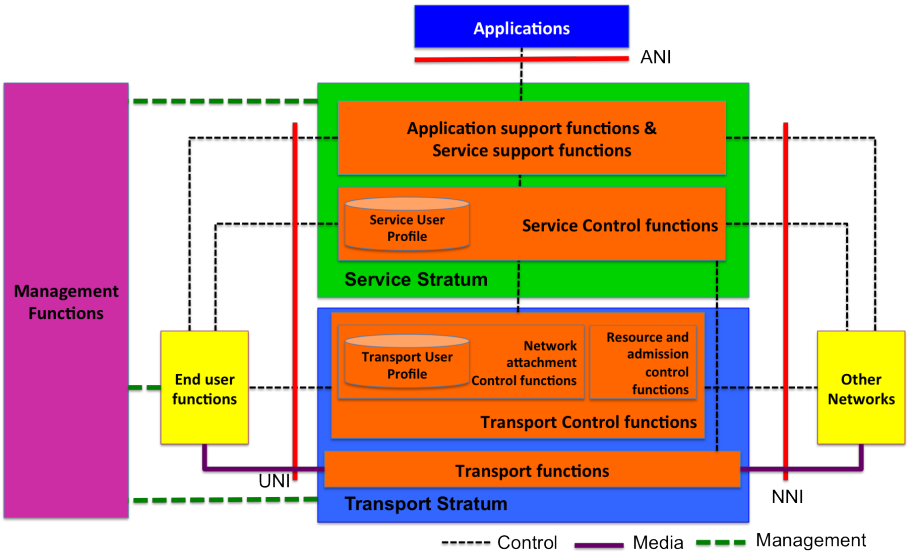


Figure 2.1. A vision of an NGN Architecture (Adapted from Figure 2 of [18])

The transport control functions accept a transport user profile, which includes authentication and bandwidth (limitations) related information. The service control function accepts a service user profile, which contains service related information [18]. ITU-T Recommendation Y.2012 [19] defines three types of interfaces: the

user-network interface (UNI), which is an interface to end-user functions; the network-network interface (NNI), which is an interface to other networks; and the application-network interface (ANI), which provides an interface to the third-party application services.

2.2 IP based Fixed/Mobile Network Convergence

Convergence of fixed and mobile networks is an evolutionary trend in the telecommunication industry. This convergence supports an intelligent IP-based end-to-end network and enables end-user devices to seamlessly access services over fixed or mobile networks. Figure 2.2 shows S. Dixit's vision of NGN based upon all IP-based fixed/mobile convergence.

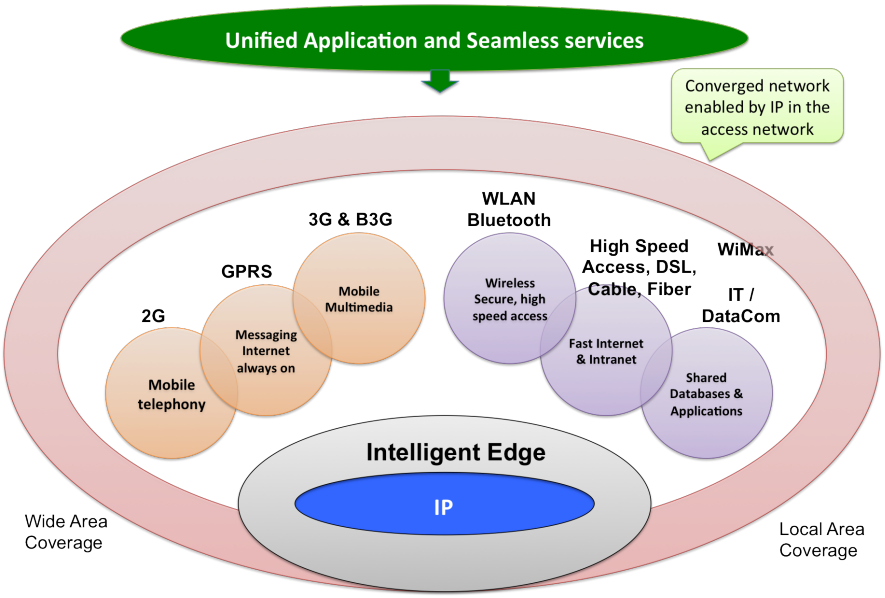


Figure 2.2. A vision of an Fixed/Mobile network convergence (Adapted from Figure 3 of [21])

2.3 IP Multimedia Subsystem (IMS)

The IP Multimedia Subsystem began as an subsystem for Universal Mobile Telecommunication Systems (UMTS) networks. This subsystem was designed to deliver Internet Protocol based multimedia services to mobile subscribers. The IMS specification was originally designed by the 3rd Generation Partnership Project (3GPP) in their Release 5 standard. Further IMS functionality has been specified

in subsequent 3GPP Releases [22].

IMS enables cellular network operators to provision interactive multimedia services cost effectively by making use of IP networks. IMS uses the Session Initiation Protocol (SIP) as a signaling protocol for initiating and managing sessions. 3GPP collaborated with European Telecommunication Standard Institute (ETSI) experts to ensure efficient re-use of Internet standards. Figure 2.3 depicts the IMS architecture. This architecture is divided into three different layers: service/application layer, IMS layer, and transport layer [23].

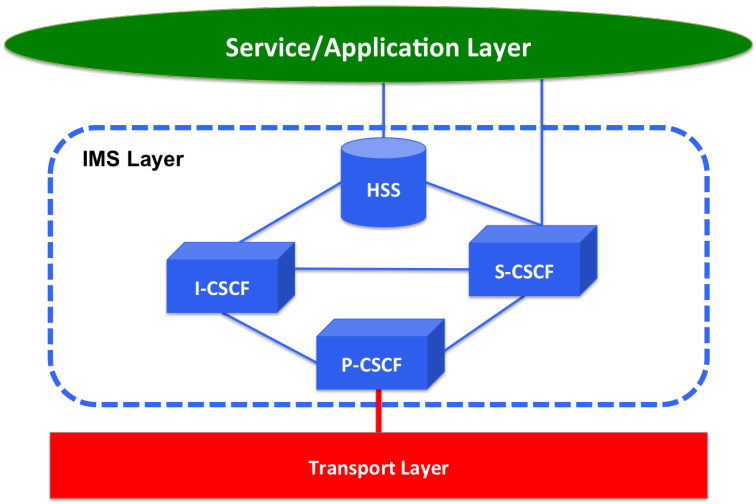


Figure 2.3. A vision of an simplified IMS Architecture

The IMS core architecture is divided into Call/Session Control Functions (CSCFs) and a Home Subscriber Server (HSS). Each of these elements is described further in the following sections.

2.3.1 Call/Session Control Functions (CSCFs)

The CSCFs play important roles in IMS. It is important to note that while the functions of the various types of CSCFs are described separately, all of them could be running on a single computer or distributed over multiple computers.

2.3.1.1 Proxy Call Secession Control Function (P-CSCF)

The P-CSCF is the entry point within an IMS network for IMS subscriber within the access networks which are connected to this specific IMS. It acts as a proxy to accept and serve requests from IMS compatible SIP user agents. The P-CSCF may also act as a SIP user agent, this means that it may generate and terminate SIP

transactions.

The Policy Decision Function (PDF) is a logical component of the P-CSCF, but it may be implemented in a separate node. If the PDF is implemented in a separate physical node, then the interface between the PDF and the P-CSCF is the Gq interface standardized in TS 182 006 [20].

The P-CSCF forwards SIP register requests based on the subscriber's home domain name to the appropriate I-CSCF. These SIP register requests are received from a IMS capable SIP user agent running on user equipment (UE). The P-CSCF forwards SIP requests or responses from and to the UE. The P-CSCF forwards SIP messages to the S-CSCF whose name was been received during the registration procedure. (i.e., as part of the registration procedure the subscriber is assigned to a S-CSCF).

The P-CSCF utilizes IPsec security associations for all communication with each UE. The P-CSCF is also responsible for performing compression/decompression of SIP messages [23].

2.3.1.2 Serving Call Secession Control Function (S-CSCF)

The Serving CSCF (S-CSCF) is a central component in IMS. It provides all of the session control service mechanisms. It may also maintain session state as needed by the network operator (i.e., the S-CSCF may be a statefull SIP proxy). An IMS operator's network may contain multiple S-CSCFs. Each of these may have different functionality.

The S-CSCF accepts registration requests and processes these requests based upon the registration information and its interaction with a location server. The S-CSCF performs session control for all SIP sessions (via this specific IMS).

The S-CSCF may act as a SIP proxy server, as described in RFC 3261 [25], in order to accept and forward SIP requests. It may act also like a SIP user agent, as described in RFC 3261 [25], in order to independently terminate and generate SIP transactions [23].

2.3.1.3 Interrogating Call Secession Control Function (I-CSCF)

The Interrogating CSCF (I-CSCF) is an entry/exit point for all SIP sessions destined to another network operator or from a roaming subscriber currently located within the service area of another IMS operator.

The I-CSCF assigns a S-CSCF based upon information obtained from the HSS after a SIP registration request from a IMS subscriber's SIP user agent.

The I-CSCF is also responsible for routing SIP requests to the S-CSCF for SIP requests received from another network and forwarding SIP requests and responses to the S-CSCF [23].

2.3.2 Home Subscriber Server (HSS)

The Home Subscriber Server (HSS) is an evolution of the Global System for Mobile Communication (GSM) Home Location Register (HLR) and Authentication Center (AuC). The HSS provides a central database of subscriber information. This database enables the IMS network entities to handle SIP sessions. The HSS performs authentication and authorization of the subscriber and stores all of the subscription-related information (i.e., the subscriber profile). The HSS also stores subscriber location information.

An IMS network may contain multiple HSSs, but a particular subscriber's user profile information is stored in a single HSS [23].

2.3.2.1 Subscriber Location Function (SLF)

The Subscriber Location Function (SLF) in needed when multiple HSSs are used within an IMS network. The SLF indicates which HSS stores a specific subscriber's user profile. The SLF is accessed via the Dx interface by the CSCFs and accessed via the Dh interface by application servers (AS) [23].

2.3.3 User Identities

There are numerous identities that may be utilized within a IMS. The sections below described two of these: a private user identity used by an IMS operator to identify a subscriber and a public user identity (or identities) used by others to initiate communication sessions with a subscriber.

2.3.3.1 Private User Identity

The home IMS operator assigns a unique permanent global identity known as a private user identity to each of their subscribers. This identity is used for registration, authorization, administration, and accounting purposes. According to the 3GPP specifications [23] the private user identity shall take the form of a Network Access Identifier (NAI) as described in RFC 2486 [26], such as "user@ims-network-domain" [26].

2.3.3.2 Public User Identity

The public user identity/identities are used for initiating a communication session between subscribers. There might be multiple public user identities per private user

identity. According to the 3GPP specifications [23] each public user identifier shall take the format of a SIP URL as described in RFC 3261 [25], i.e., “sip:user@ims-network-domain” or a telephone URL “sip:+358-555-1234567;postd=@ims-network-domain;user=phone” [25].

Chapter 3

Cloud Computing

This chapter presents emerging cloud computing technologies which are relevant to this thesis.

3.1 Introduction

Computing is being reorganize according to a service model in which services are commoditized and delivered to users. In this model, users access services based on their demands irrespective of how (or where) the services are provisioned. Numerous computing paradigms such as cluster computing, grid computing, and more recently cloud computing, are designed to realize this utility computing vision. Cloud computing is an evolution of several computing paradigms, such as Internet delivery, Pay-per-Use-On-Demand utility computing, virtualization, grid computing, distributed computing, storage elasticity, content outsourcing, and Web 2.0 [31, 32, 33]. The infrastructure referred to as a "cloud", enables on demand provisioning of services across the world [27].

There are numerous definitions of cloud computing [27, 28, 29]. According to the U.S. National Institute of Standards and Technology (NIST) Definition of Cloud Computing, NIST SP 800-145 [30]:

"Cloud computing is 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. This cloud model is composed of five essential characteristics, three service models, and four deployment models."

Cloud computing has created a variety of market opportunities for many organizations [38]. A cloud provider considers specific QoS key performance indicators (KPIs) with respect to fulfilling their customer's demands. Figure 3.1

shows a market-oriented vision of a cloud architecture.

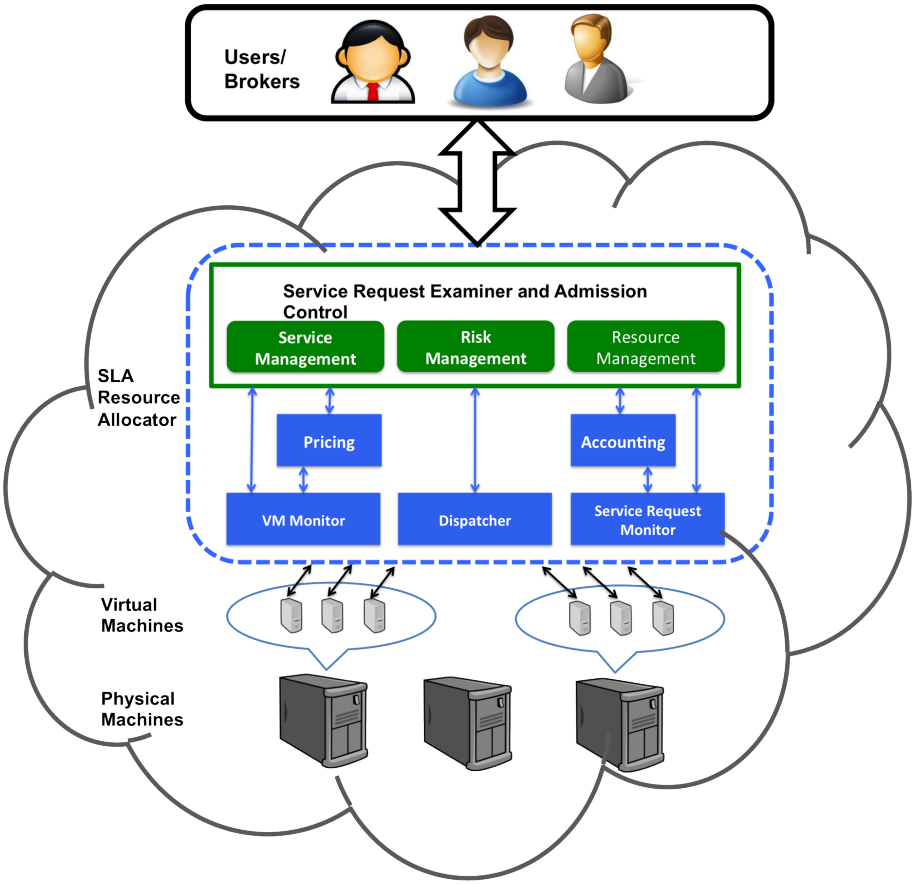


Figure 3.1. Market-oriented vision of a Cloud Architecture (Adapted from Figure 3 of [27])

3.2 Cloud Computing Characteristics

A vision of cloud computing associated with the essential fundamental characteristics described in NIST SP 800-145 is shown in Figure 3.2.

On-demand self-service: Consumers can automatically provision computing capabilities based on demand without requiring human intervention with a cloud service provider.

Broad network access: Consumers can easily access the capabilities from different devices (e.g. mobile phones, tablets, laptops, and workstations) over a

network.

Resource pooling: The cloud provider's pool of physical hosts and virtual resources are dynamically assigned and reassigned using a multi-tenant model, to serve multiple consumers' demands. Resource allocation is transparent to the consumers, but consumers may specify location parameters such as county, state, or a specific data center at a higher level of abstraction. This resource pooling concept was illustrate in Figure 3.1.

Rapid elasticity: The cloud computing resources automatically dynamic scale up and scale down based on consumer demand. Therefore, the service is available to the consumers at any time without degrading the QoS that they experience.

Measured service: Cloud systems automatically control and optimize resources based on consumers, demands, and required service types (such as storage, processing, bandwidth, and number of active user accounts). The usage of all the services are monitored, controlled, and reported transparently.

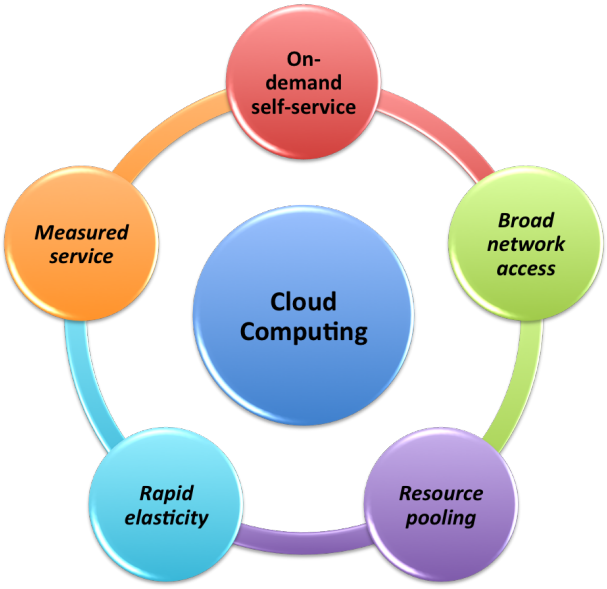


Figure 3.2. A vision of an Cloud Computing Essential Characteristics

3.3 Service Models

Cloud computing can support everything as a service (i.e. XaaS). These services can be categorized into three different service models as described by NIST SP

800-145 [30]. This cloud computing service model infrastructure is shown in Figure 3.3. These different service models will be described in the subsequent subsections.

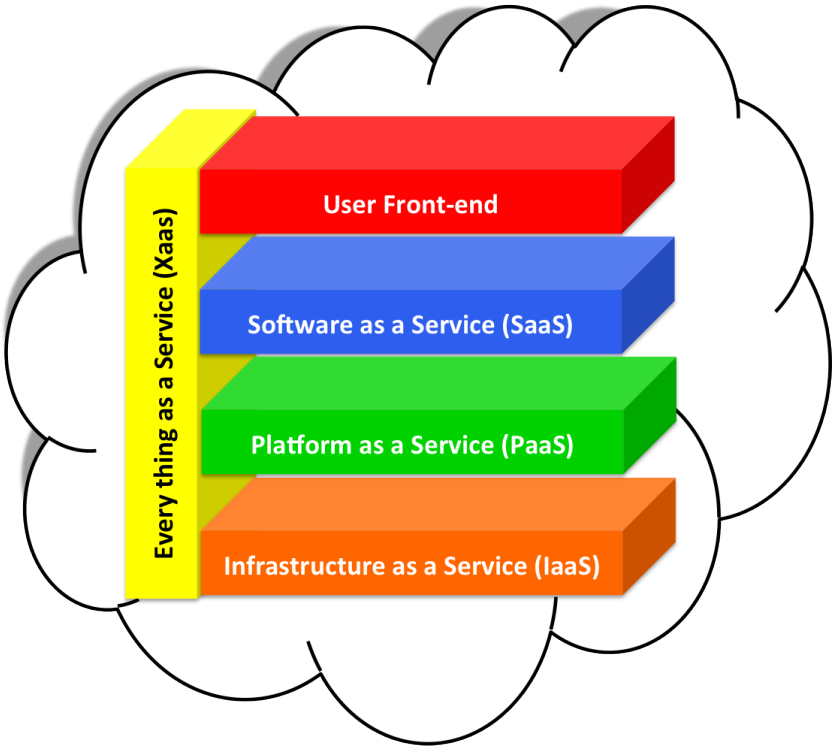


Figure 3.3. A vision of an Cloud Computing Service Models (Adapted from Figure 1 of [33])

3.3.1 Software as a Service (SaaS)

SaaS enables on demand availability of software or an application to consumers over the internet and makes this software accessible through either a web browser or program interface. These software services are hosted transparently on a cloud infrastructure. The consumer does not directly access the underlying cloud provider's infrastructure. SaaS is also called Application as a Service (AaaS). Examples of SaaS providers are Salesforce Customer Relationships Management (CRM) system, NetSuite, and Google Office Productivity application [30, 33, 2].

3.3.2 Platform as a Service (PaaS)

PaaS provides a development environment for the development of applications that will be deployed in a cloud infrastructure. PaaS enables developers to develop applications that are hosted as a service delivered by a cloud platform. The

developers use the programming language, libraries, services, and tools made available in the PaaS for their application development. The consumer do not directly access the underlying cloud provider's infrastructure. Examples of PaaS providers are Facebook F8, Salesforge App Exchange, Google App Engine, Bunzee connect, Windows Azure, IBM Websphere Cloudburst, Force.com, and Amazon EC2 [30, 33, 2, 34].

3.3.3 Infrastructure as a Service (IaaS)

In IaaS the cloud service provider delivers processing, storage, networks, and other computing resources to the consumer elastically on demand. The user pays for only the resources that as they use. The consumer can select their operating system, storage, and deployed applications; as well as having limited access to networking components. The consumers do not control the underlying cloud provider's infrastructure. Examples of IaaS providers are Amazon Web Services, GoGrid, MossoRackspace, MSP On-Demand, and masterIT. Sometimes IaaS is also referred to as Hardware as a Service (HaaS) [30, 33, 2].

IaaS can be further divided into three subcategories [34]:

- Computing as a Service (CaaS),
- Storage as a Service (SaaS), and
- Database as a Service (DaaS).

3.4 Deployment Models

The cloud computing service model infrastructure can be deployed using the four different deployment models described by NIST SP 800-145 [30]. Figure 3.4 shows a vision of a these cloud computing deployment models. The subsections below describe each of these deployment models.

3.4.1 Private cloud

In a private cloud the cloud infrastructure provisions resources within an enterprise or organization encompassing multiple consumers. A private cloud is owned, control, managed, and operated by an enterprise or organization. There may even be a combination of organization and third-party offerings. The private cloud's resources may be on or off premises and firewall protects the private cloud from access by others [30, 34, 35].

3.4.2 Community cloud

In a community cloud the cloud infrastructure is designed for multiple organizations of a specific user community that have common concerns, such as mission,

security requirements, policy, and compliance considerations. The community cloud infrastructure can be control, managed, and owned by one or more organizations or a third-party [30, 34].

3.4.3 Public cloud

A public cloud's cloud computing resources are available for use by the general public and accessible through standard APIs over the Internet. The public cloud may be the property of an organization, such as a business, an academic institution, or a government, or combination of them and the cloud can be managed by the organization itself [30, 34, 35].

3.4.4 Hybrid cloud

A hybrid cloud infrastructure is a combination of two or more distinct deployment models (such as private, community, or public) that persist as new cloud infrastructure. The hybrid cloud remains bounded by standardized technologies that ensure portability of data and applications, such as cloud bursting for load balancing between cloud infrastructures [30, 34].

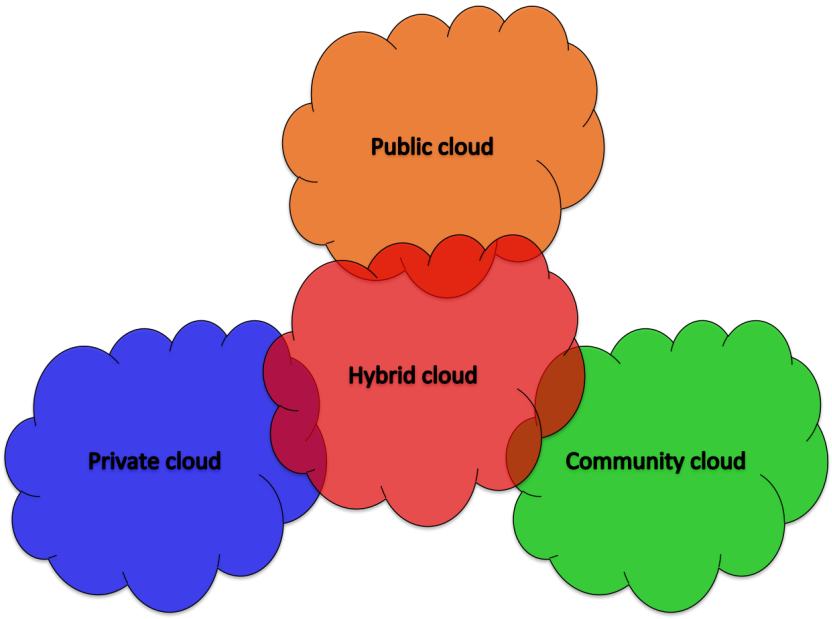


Figure 3.4. Cloud Computing Deployment Models

3.5 Scalability and Elasticity

The prime aim of the cloud computing is provisioning of nearly infinite scalability. The cloud architecture can scale using two different approaches to serve consumer's demands: vertically or horizontally. These two approaches are as illustrated in Figure 3.5 and described in the subsections below.

3.5.1 Vertical Scaling

Vertical scaling deploys more powerful computing resources to accommodate the demand. This scale-up approach usually works well, but involves either a huge capital expenditure or the demand may exceed the available capacity *before* the new more powerful computing resource is deployed.

3.5.2 Horizontal Scaling

The traditional scale-out approach gradually scales computing resources in small chunks to accommodate the demand. Most large-scale business organizations employ a service-oriented design by following this scale-out approach. However, horizontal scaling requires monitoring of demand on a regular basis and then scaling the infrastructure to serve the demand.

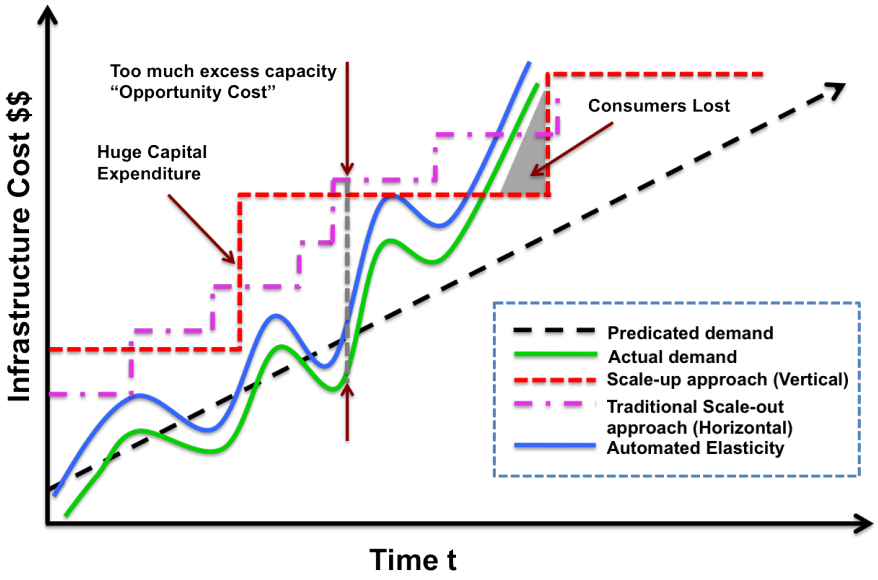
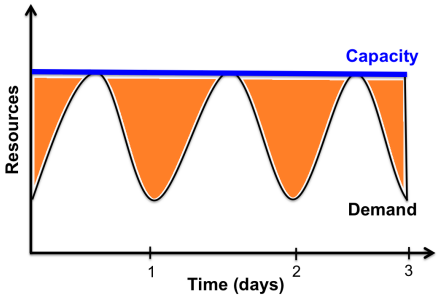


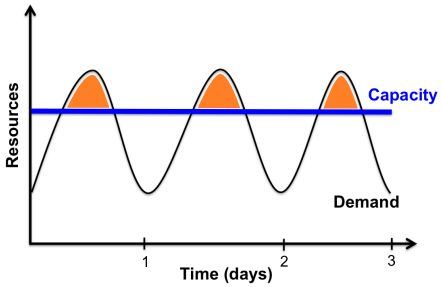
Figure 3.5. Automated Cloud Elasticity (Adapted from Figure 2 of [36])

3.5.3 Cloud Elastically

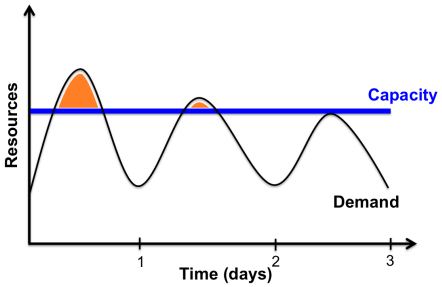
Elasticity is one of the essential characteristics of the cloud computing. Cloud computing's on-demand automated elasticity enables more optimal utilization of the computing infrastructure's resources and aligns the dynamic scale-up and scale-down of resources with actual demand. Figure 3.5 illustrate the concept of automated elasticity.



(a) Over-provisioning



(b) Under-provisioning Case A



(c) Under-provisioning Case B

Figure 3.6. Over-provisioning (the orange shows wasted resources) and Under-provisioning (Adapted from Figure 2 of [37])

When a cloud engineer enables elasticity in an application service, there are two possible scenarios that can occur:

Over-provisioning: In an over-provisioning scenario, if the service operator predicts the maximum demand correctly, then capacity is wasted during non-peak times. This means, resources are wasted as shown by the shared area in Figure 3.6(a). However, this approach guarantees QoS *even* during peak hours. [37]

Under-provisioning: In an under-provisioning scenario, the service operator does not take into account the potential revenue loss due to failing to serve the user demand which is shown by the shaded area of Figure 3.6(b). This means, that the service operator provides poor QoS during peak hours due to the service demand exceeding the actual capacity.

Furthermore, after suffering poor QoS, some of the users start to leave the site permanently until the peak user demand equals the actual capacity. Again users start to receive acceptable QoS, as shown in Figure 3.6(c). [37]

Chapter 4

Tools and Technologies

This chapter presents an overview of the open source technologies that are used for setting up the test environment for the proposed SUT.

4.1 Virtualization Technologies

Virtualization enables a computer to run one or more virtual machines. Virtualization technologies enable energy-efficient computing, and more optimal hardware utilization of processors, memory, and storage resources. Virtualization technologies enable the creation and execution of virtual machines, providing virtualized storage, networks, and virtualizing applications.

A hypervisor is used as the underlying virtualization technology. This hypervisor is a computer program that enables the user to create and run a pool of virtual machines on top of a physical machine. The physical machine on which this hypervisor is running virtual machines is known as a host machine. Figure 4.1 shows the high level architecture of this virtualization.

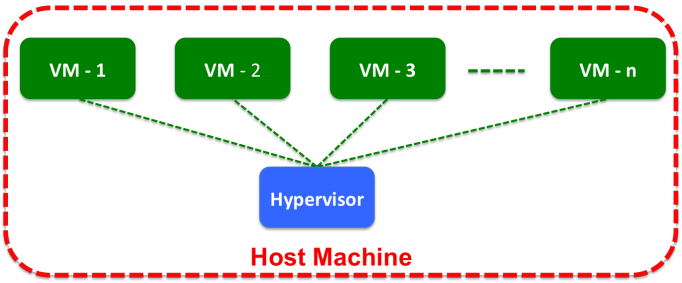


Figure 4.1. High Level Virtualization Architecture

There are many virtualization tools such as Virtual Machine Manager (VMM), VMware, VirtualBox, and etc. In this project, the Virtual Machine Manager is used

as an virtualization technology to setup the test environment for the SUT.

4.1.1 Virtual Machine Manager (VMM)

The Virtual Machine Manager (VMM) application (from http://virt-manager.org/) is a graphical user interface (GUI) for creating and managing virtual machines. It provides a summery view of the running virtual machines, showing their performance and resource utilization statistics [39]. Figure 4.2 depicts the VMM stack and VMM's supporting tools are:

Virt-Install: A tool which enables install operating systems into virtual machines. It also provides a GUI for VM creation.

Virt-Clone: A tool for cloning an existing inactive virtual machine. It copies the disk images to create a new configuration. It is also possible to clone the disk image using the GUI.

Virt-Image: A tool for installing operating systems based on pre-defined image.

Virt-Viewer: A lightweight interface with graphical display of the virtualized guest OS.

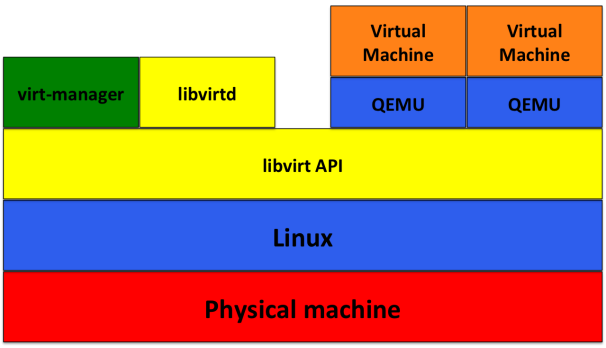


Figure 4.2. Representation of VMM stack with QEMU (Adapted from Figure 1 of [40])

4.2 OpenIMS Core

OpenIMS Core is an open source implementation of IMS's core components (Call Session Control Functions (CSCFs) and Home Subscriber Server (HSS)) according to the 3GPP standard [23]. OpenIMS Core was developed by Fraunhofer Institute FOKUS [42].

The high level architecture of OpenIMS Core is shown in Figure 4.3. The OpenIMS Core's CSCFs are based on a SIP Express Router (SER) [44]. SER is an open source SIP server widely used to implement Voice over IP (VoIP) services, even for a large VoIP infrastructure. Each CSCF component was developed so that it acts as an independent node in the IMS architecture. The implemented CSCF components are P-CSCF, S-CSCF, and I-CSCF (these were described in Section 2.3.1).

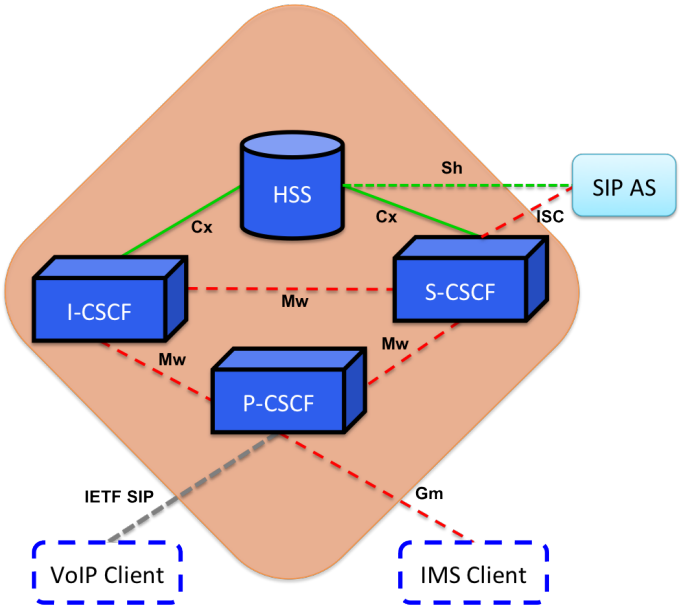


Figure 4.3. A vision of an OpenIMS Core Architecture (Adapted from the figure shown in [43])

As seen from Figure 4.3, the P-CSCF communicates with a UE via the Gm interface, and the CSCFs communicate with each other via the Mw interface. HSS communicates with the I-CSCF and S-CSCFs via the Cx interface over Diameter and with Application Servers (ASs) via the Sh interface.

4.2.1 Modules use to realize the OpenIMS CSCFs

The main modules of OpenIMS CSCFs are described in the following subsections.

4.2.1.1 CDiameterPeer (CDP)

The CDP component is used for realm routing based on the fully qualified domain name (FQDN) of the destination host. CDP enables efficient bidirectional Diameter communication for SER.

4.2.1.2 IMS Service Control (ISC)

ISC provides an interface that connects the S-CSCFs with application servers. It also enables access to an IMS network through a P-CSCF via a SIP-to-IMS gateway by providing an authentication translation mechanism.

4.2.2 Modules of FOKUS HSS (FHoSS)

The OpenIMS Core uses the FOKUS HSS to store user profiles, along with authentication & authorization information, and user location information. Additionally, FHoSS provides a web-based management console. The high level architecture of FHoSS is shown in Figure 4.4. The FOKUS HSS architecture is divided into two main layers: Diameter Interface Layer and Data Access Layer.

4.2.2.1 Diameter Interface Layer (DIL)

The core module of the FHoSS is a HssDiameterStack implementation, which is written entirely in Java. It uses the DiameterPeer object to send requests to other elements and retrieves responses via a CommandListener object. FHoSS supports three types of interfaces: Sh, Cx, and Zh. These interface implementations can be found in the de.fhg.fokus.cx, de.fhg.fokus.sh, and de.fhg.fokus.zh packages.

4.2.2.2 Data Access Layer (DAL)

DAL is based on the Hibernate persistence framework. FHoSS uses MySQL as a backend database to store operational data. The DAL implementation can found in the de.fhg.fokus.hss.model package. DAL utilize the Java Database Connectivity (JDBC) driver, hence any database that has a JDBC driver could be used.

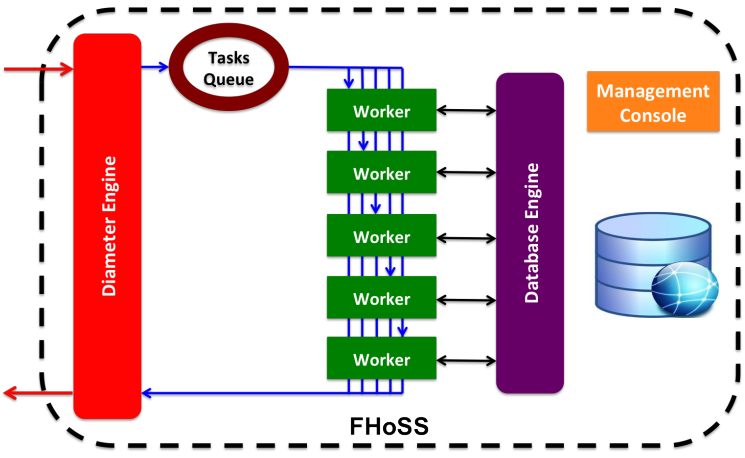


Figure 4.4. Highe Level FHoSS Architecture (Adapted from the figure shown in [41])

4.3 HAProxy Load Balancer

HAProxy is a open source layer 4 and layer 7 load balancer which provides high availability and high performance for TCP and HTTP-based applications. HAProxy distributes the workload across a set of servers in order to optimize resource usage and maximize performance.

The front-end applications that depend on a backend database can easily scale by utilizing many parallel connections to the database. HAProxy provides throttling of connections towards one or more database servers and prevents overloading of a single server with too many requests (thus it acts as a database transaction monitor). All front-end clients connect to the HAProxy instance, and then the HAProxy forwards the client's request to one of the available database servers based on the selected load-balancing scheme. Currently, HAProxy supports different load balancing mechanisms, specifically: roundrobin (rr), static-rr, leastconn, source, url, url_pram, hdr, and rdp-cookie [45].

In this project, HAProxy load balancing is employed in-front of a MySQL cluster to transparently distribute the front-end client's requests to the MySQL servers in the cluster.

4.4 MySQL Cluster Technology

The architecture of a MySQL cluster is designed to accommodate all dimensions of scalability. The MySQL cluster architecture supports the following fundamental characteristics of scalability [46]:

- Auto-sharding for write-scalability,
- Real-time responsiveness,
- Active / active geographic replication,
- Online scaling and schema upgrades,
- SQL and NoSQL interfaces, and
- 99.99% availability.

Figure 4.5 shows the high level architecture of MySQL cluster which supports high write scalability across multiple SQL, and NoSQL APIs. The MySQL cluster architecture consist of three types of nodes which collectively provide service to the end application:

Data Nodes: The data nodes are responsible for managing data storage and access to data. MySQL cluster technology by default provides automatic

transparent partitioning/sharding, load balancing, replication, failover, and self-healing of tables across different data nodes.

Application Nodes: The application nodes are the front-end of the MySQL cluster nodes which provide connectivity to the backend data nodes. MySQL cluster provides a standard SQL interface, including connectively with different programming languages and frameworks such as NoSQL (memcached), REST/HTTP, C++ (NDB-API), Java, and Java Persistence API (JPA).

Management Nodes: The management nodes are used to manage the MySQL cluster configuration and to provide arbitration in the event of a network partition.

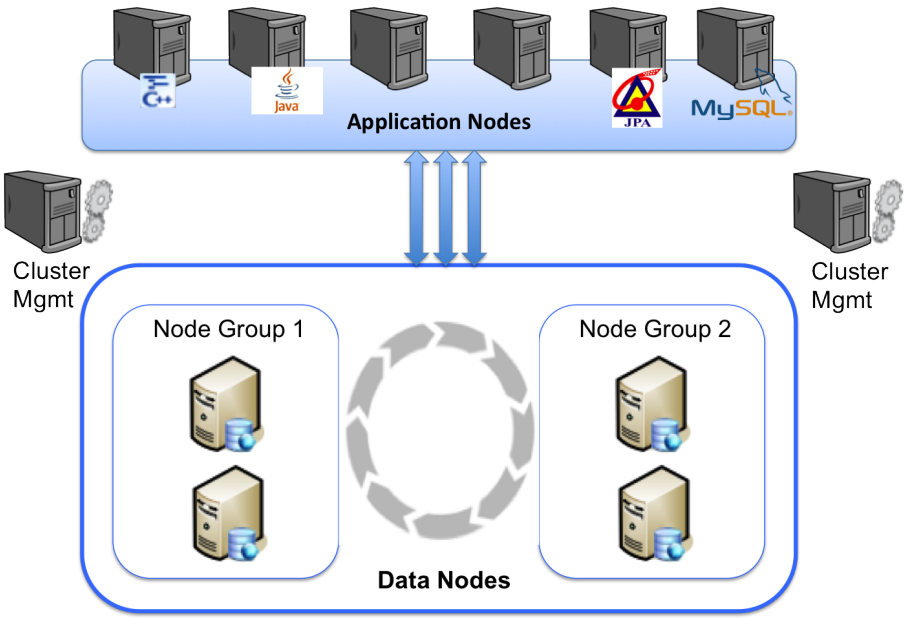


Figure 4.5. MySQL cluster Architecture (Adapted from Figure 1 of [46])

4.5 IMS Bench SIPp

IMS Bench SIPp is an open source implementation of a test system designed by Intel Cooperation in accordance with the IMS/NGN Performance Benchmark specification ETSI TS 186 008 [47, 48, 49]. It is an extended version of an earlier open source SIP traffic generator, with built-in default scenario files to handle a large number of users. The following default scenario files are available [51]:

- Successful call,

- Successful messaging,
- Registration,
- De-registration, and
- Re-registration.

Figure 4.6 shows a high level architectural overview of IMS Bench SIPp, which consist of a test system and a IMS SUT. The test system consist of a manager and one or more SIPp traffic generator instances, which originate IMS events such as registration and de-registration, session set-up or tear-down, and messaging to the SUT. The IMS SUT responds to these events. Additionally, we utilizes one or more system monitoring agents for monitoring CPU and memory utilization. The brain of the IMS benchmark test system is a collection of traffic set scenarios, associated with the probability of occurrence in the set of test procedures, which resemble the load on the test system that might occur in the real world [51, 50].

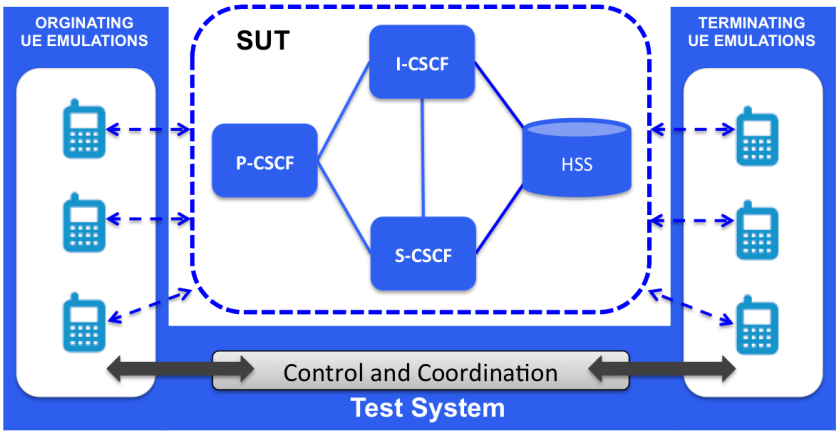


Figure 4.6. High Level Architecture of IMS Bench SIPp (Adapted from Figure 2 of [50])

The IMS Bench SIPp test system offers the following benefits [51]:

- Supports concurrent execution of call session setup, registration, de-registration, and instant messaging scenarios.
- Supports random selection of scenarios with a defined probability of occurrence for each type of scenario (e.g. 30% calling, 30% registration, 20% messaging, 10% de-registration, and 10% re-registration).
- The number scenario attempts to follow a Poisson distribution per unit time.

- Supports random selection of users from a pool of users that are suitable for a particular scenario.

The SIPp logical component of IMS Bench SiPp test system can be run on one or more machines. The SIPp instance originates scenario attempts and each has its own static set of users. A manager is responsible for configuring the SIPp instances, monitoring the failure rate in order to stop the benchmark, and logging (CPU and MEM) utilization of the SUT as reported by monitoring agents. Figure 4.7 shows the high level internal architecture of the IMS Bench SIPp test system.

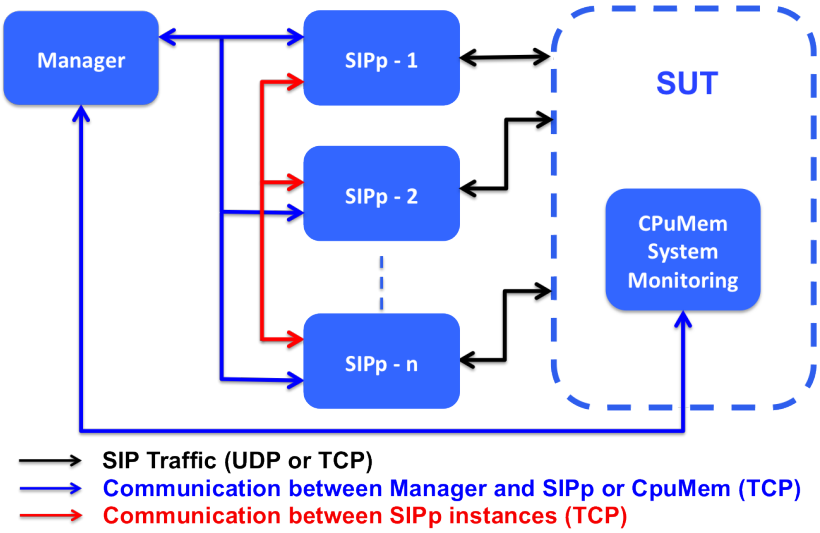


Figure 4.7. High Level Internal Architecture of IMS Bench SIPp (Adapted from the figure shown in [51])

In this project, IMS Bench SIPp scenarios are employed for performance evaluation of the SUT. Specifically, the registration scenario will be used to test HSS horizontal elastic scalability. SIPp clients originate registration requests to the SUT (the OpenIMS core), and then wait for a registration success (or failure) response.

4.6 Zabbix

Zabbix is an enterprise-class open source monitoring solution which supports the following advanced monitoring features [52]:

Monitor Everything: Supports agent-less monitoring of network devices, databases, hardware monitoring, and provides a centralized web-based monitoring system. It accurately gathers KPIs and statistics.

Enterprise Ready: Zabbix is specifically designed to provide highly available, high performance optimized monitoring for a large-scale distributed environment.

Pro-active Monitoring: Zabbix provides improved service quality by sending notification messages via email, SMS, or Jabber for each notable event. Remote commands can be sent to Zabbix to reduce operating cost by avoiding downtime.

Capacity Planning: Capacity planning is done in order to plan for business growth and to predict the future resource need, while avoiding wasting resources.

4.6.1 Zabbix Server

The Zabbix Server is a central component to which Zabbix agents and proxies report data on the availability and integrity of a system that is being monitored. The functionality of the Zabbix server is divided into three components: Zabbix server, web front-end, and database storage. All of the configuration information is stored in the database, which the Zabbix server and the web front-end interact with. The Zabbix server web interface enables creation of a group of hosts and configuration of each host according to a predefined template. Zabbix also provides a mechanism for auto registration and auto discovery of the new hosts in a particular network.

4.6.2 Zabbix Agent

The Zabbix agent is a process that is deployed on a monitored host to actively monitor local resources and applications. The Zabbix agent locally gathers system information, then sends this data to the Zabbix server encoded in a JSON format. Zabbix agents can perform two types of monitoring: passive and active.

In passive monitoring, the Zabbix server sends data, for example, the CPU load of a virtual machine, to a Zabbix agent which forwards this measurement to a Zabbix server.

In active monitoring, the Zabbix agent first retrieves a list of items to monitor from a Zabbix server. Afterwords, the Zabbix agent will periodically send the latest values of these monitored items to the Zabbix server.

Whether to perform passive or active monitoring is configured by selecting the respective “Zabbix agent” or “Zabbix agent (active)” item types.

Chapter 5

System Architecture Design

This chapter presents all the use case scenarios which were considered when in designing the testbed system's architecture.

5.1 IMS Core Virtualization

The virtualization of OpenIMS core consist of two steps, which are described in subsections 5.1.1, and 5.1.2. Furthermore, the highly available and highly scalable database system's architecture design for FHoSS database system is described in Section 5.2.

5.1.1 OpenIMS core Virtualization

This section describes the virtualization of the OpenIMS Core. After creating and instantiating virtual machines on a host machine, all the OpenIMS core nodes (P-CSCF, S-CSCF, I-CSCF, and FHoSS) are deployed on separate VMs. In this scenario, all of the FHoSS components (DIL, DAL, and MySQL database) are running on a single VM. At this point, all of the OpenIMS core has been virtualized. The installation and configuration of an OpenIMS core are described in Appendix A. Figure 5.1 illustrates the virtualized high level OpenIMS core architecture. The functional testing of a virtualized OpenIMS core will be described in Section 6.1.

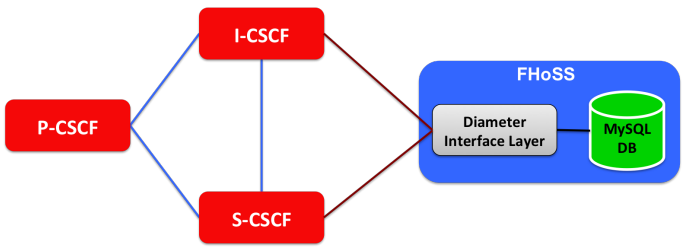


Figure 5.1. High Level Virtualized OpenIMS Architecture

5.1.2 OpenIMS core with virtualization of FHoSS layers

This scenario describes the virtualization of the FHoSS layers. FHoSS is divided into two main layers, as was described in Section 4.2.2. DIL acts as a front-end interface to handle the Cx Diameter interface traffic and DAL acts as a back-end interface to communicate with the database via a JDBC driver. Figure 5.2 shows the high level internal architecture of FHoSS.

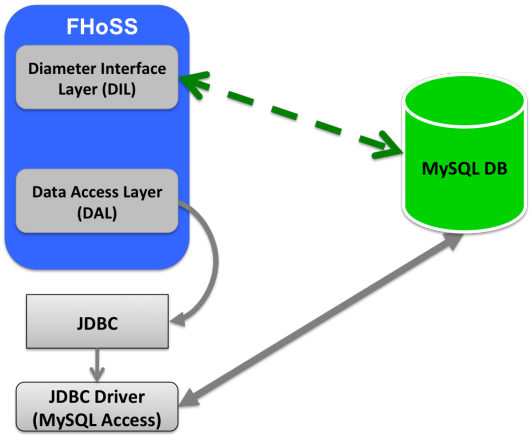


Figure 5.2. FHoSS High Level Internal Architecture

The previous section described the virtualization of OpenIMS core. In this section, we describe the virtualization of each of the FHoSS components. Both FHoSS DIL and DAL are running on a single VM, but the MySQL database runs on another VM. In this way, FHoSS's front-end diameter interface and the actual MySQL database are virtualized independently. Virtualizing the MySQL database provides a way to horizontally scale the database underlying FHoSS. In order to do this we must configure the VM's IP address in both the files "DiameterPeerHSS.xml" and 'hss.properties" (i.e., for the Diameter Engine and tomcat processes, respectively). The Hibernate persistence framework, an implementation of DAL, is running on the same VM on which DIL is running. We explicitly specify the MySQL database VM's IP address <ip address> in the "hibernate.properties" parameter (hibernate.connection.url=jdbc:mysql: //<ip address>:3306/hss_db). We utilize the default MySQL TCP port. The resulting high-level architecture is shown in Figure 5.3, which illustrates the complete virtualization of all of the OpenIMS core components, including the FHoSS.

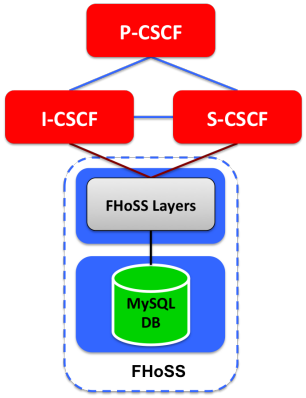


Figure 5.3. High Level Virtualized OpenIMS Core including HSS planes

Now that we have successfully deployed and configured the OpenIMS core components into different VMs, the functional testing of the fully virtualized OpenIMS core components can be performed, as will be described in Section 6.1.

5.2 FHoSS database system with high availability and high scalability

FHoSS uses a MySQL database as its backend database technology to store each subscriber's data. The traditional MySQL database technology uses "MyISAM" as an storage engine. Typically MyISAM resides on the same machine where the MySQL server is running, thus, the database's performance does not scale well. In contrast, the FHoSS database architecture will be designed to ensure high availability (by avoiding a single point of failure) and high scalability (in order to enhance the performance of the database system).

The concept of high availability, and a high scalability solution for MySQL database systems is described in the following subsections.

5.2.1 High availability and high scalability

A system that ensures full time availability without any loss or degradation of service is know as a high availability system [56]. A highly available system typically consist of redundant software and hardware resources that delivers the service at any time, despite the failure of individual service nodes [53].

According to S. Galiano [53], high availability can be achieved by introducing either active redundancy or passive redundancy. Active redundancy ensures the service's availability by forming a cluster of service nodes that are running

concurrently. Passive redundancy is based on the master/slave architecture concept. The master is in the active state and acts as the primacy service node, while a slave is in passive/hot standby state and acts a secondary backup service node. The slave will be substituted for the master when the master node fails.

Active redundancy can be achieved by creating a cluster that consists of a pool of service nodes running in parallel, along with a load balancing mechanism. The load balancing system is used on top of the cluster nodes to provide a single access point to the system. This type of architecture ensures the high availability and high scalability of a system. The load balancing mechanism distributes the workload across multiple service nodes, enabling high availability, and while also ensuring high scalability as new service nodes cab be added to the cluster without any major modifications. However, the load balancer might introduce a single point of failure to whole system architecture.

Passive redundancy can be achieved by setting up a backup service node in conjunction with open source packages, such as Red Hat Piranha, UltraMonkey, heartbeat plus ldirectord, heartbeat plus mon, and Keepalived [54].

To avoid introducing a single point of failure, we can merge both the active and passive redundancy approaches. This combined approach should provide high availability, and high scalability of the overall system. Figure 5.4 illustrates the high level architecture of such a highly available and highly scalable system.

Both active and passive architectural topologies will be analyzed in the context of the MySQL database.

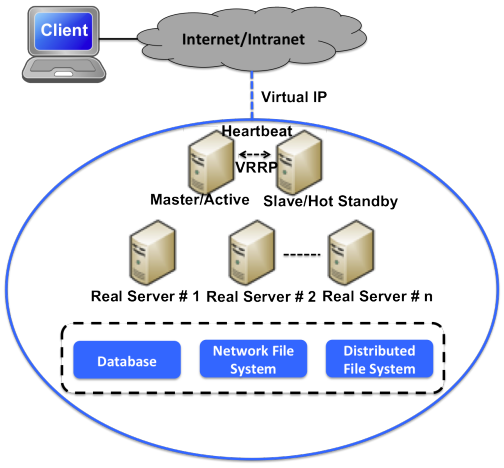


Figure 5.4. High level architecture of a highly available and highly scalable system (Adapted from the figure shown in [54])

5.2.2 MySQL database replication

MySQL supports two types of replication models, which are explained in the following subsections. MySQL's replication models are based on a master/slave architecture, i.e., one server acts a master and one or more server act as slaves [55].

5.2.2.1 Asynchronous replication

MySQL supports one way asynchronous replication, which means that the data is replicated from one MySQL database server (the master) to one or more MySQL database servers (slaves). Asynchronous replication introduces a delay in the actual copying of data from one server to another server.

5.2.2.2 Synchronous replication

In synchronous replication, data is committed to one to more machines simultaneously. This type of commit is commonly known as "two-phase commit". Synchronous replication is a native characteristic of MySQL's cluster technology.

Figure 5.5 illustrates the concept of asynchronous and synchronous replication. Replication offers the benefits of performance, ease of use, and reliability. As a result:

- A high available system can be designed to transparently switch incoming requests to a slave server in the event of a master server fails. This is typically achieved by using an passive redundancy approach as described in Section 5.2.1.
- A faster response time can be achieved by distributing the load over both the master and slave database servers, thus all of these servers can be used for processing client queries. The load balancer priovides a single entry point to the database system. This configuration is typically based upon using active redundancy approaches, as described in Section 5.2.1.

(a) Asynchronous replication

(b) Synchronous replication

Figure 5.5. MySQL Replication Techniques (Adapted from Figure 1 of [55])

Table 5.1 illustrates a comparison of the MySQL database technologies.

Table 5.1. MySQL Architecture Comparison (Adapted from Figure 5 of [55])

Requirements	MySQL Replication	MySQL Cluster
Availability		
Automated IP Fail Over	No	No
Automated DB Fail Over	No	Yes
Typical Fail Over Time	Varies	<3 secs
Auto Resynch of Data	No	Yes
Geographic Redundancy	Yes	MySQL Replication
Scalability		
Built-in Load Balancing	MySQL Replication	Yes
Read Intensive	Yes	Yes
Write Intensive	No	Yes
# of Nodes per Cluster	Master/Slave(s)	255
# of Slaves	Dozens for Reads	Dozens for Reads

5.2.3 Migration of a MySQL database to a MySQL Cluster

The MySQL cluster architecture was designed to meet the throughput and response time requirements of a large database system. MySQL cluster technology provides built-in support for synchronous replication, automatic load balancing across data nodes, data partitioning across data nodes, and automatic fail over. Designing a system by using MySQL cluster as the underlying database technology delivers high availability, high scaleability, and high reliability.

MySQL cluster technology is designed to support horizontal scalability of a database on three levels: application nodes (MySQL server) nodes, data storage nodes, and management nodes. MySQL cluster technology separates the application and data nodes and uses an "NDBCLUSTER" as an storage engine. In this way, a MySQL cluster stores its data on the data nodes, while the application nodes execute on separate machines. The advantage of a MySQL cluster is that it automatically provides transparent auto-sharding of the tables and load balancing across the data nodes, which enables the database to support horizontal scalability.

MySQL cluster technology allows on-line scaling of both database performance and capacity by adding additional application and data nodes, in turn enabling up and down scaling of the number of cluster nodes. The migration from a MySQL native database to a MySQL cluster provides high availability and high scalability of the database. The high level migration architecture from MySQL to MySQL cluster is shown in Figure 5.6. The installation and configuration of a MySQL cluster are described in Appendix B.

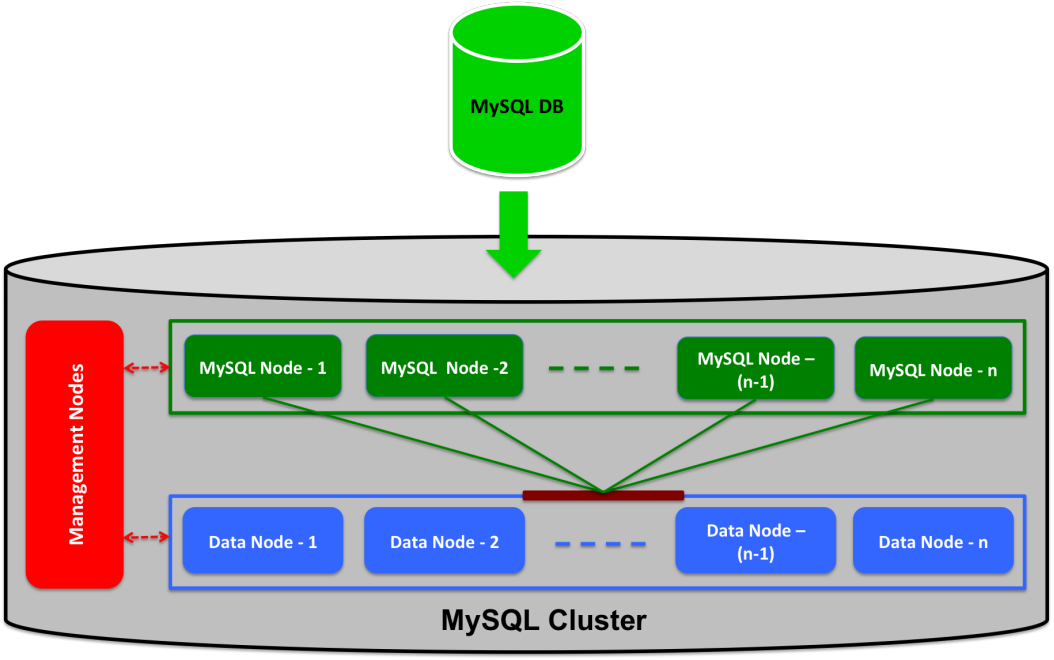


Figure 5.6. High level migration architecture of MySQL to a MySQL cluster

5.2.4 Integration of the FHoSS Diameter Interface Layer and a MySQL Cluster

This subsection describes the integration of the FHoSS Diameter interface layer and a MySQL cluster. The front-end Diameter interface of FHoSS connects to the database by opening connections to one of the application nodes of a MySQL cluster through FHoSS's data access interface. Active redundancy is achieved by using multiple application nodes of a MySQL cluster to ensure high availability of MySQL daemons. If an application node fails, the FHoSS data access interface layer reconnects to another application node within a MySQL cluster. Therefore, instead of connecting again and again to the application nodes, a smart solution is to use a load balancer in front of the application nodes of a MySQL cluster. The load balancer transparently distributes traffic sent via the FHoSS Diameter interface layer over the application nodes of a MySQL cluster.

The advantage of this load balancing mechanism are:

- Enables the FHoSS Diameter interface layer node to connect to a MySQL cluster via a single IP address (the address of the load balancer). The MySQL cluster topology is hidden behind the load balancer.
- SQL query requests are load balanced over the available application nodes of

a MySQL cluster.

- Adding or removing applications nodes of a MySQL cluster can be done without any changes to the FHoSS 'hibernate.properties" configuration file, because the load balancer's IP address is configured as the address of the database (as seen by the front end).
- The load balancer provide automated IP fail over for a MySQL cluster.

The HAProxy load balancer is placed between the FHoSS Diameter interface node and the MySQL cluster in order to provides a single entry point to the MySQL cluster. The HAProxy load balancer technology was described in Section 4.3. The HAProxy load balancer acts a front-end to the backend MySQL server application nodes. Without the HAProxy the FHoSS Diameter interface could easily over-load a single database with many concurrent connections. The overall high level integration architecture of the FHoSS Diameter interface layer and MySQL cluster as an horizontal scalable database technology is shown in Figure 5.7.

However, the use of a load balancer creates a single point of failure for the MySQL daemons. In the event a load balancer fails, access to MySQL cluster architecture will be unavailable. Therefore, we utilize passive redundancy at the load balancer layer in order to avoid the load balancer becoming a single point of failure. Further details of this solution to this problem are described in subsection 5.2.4.1.

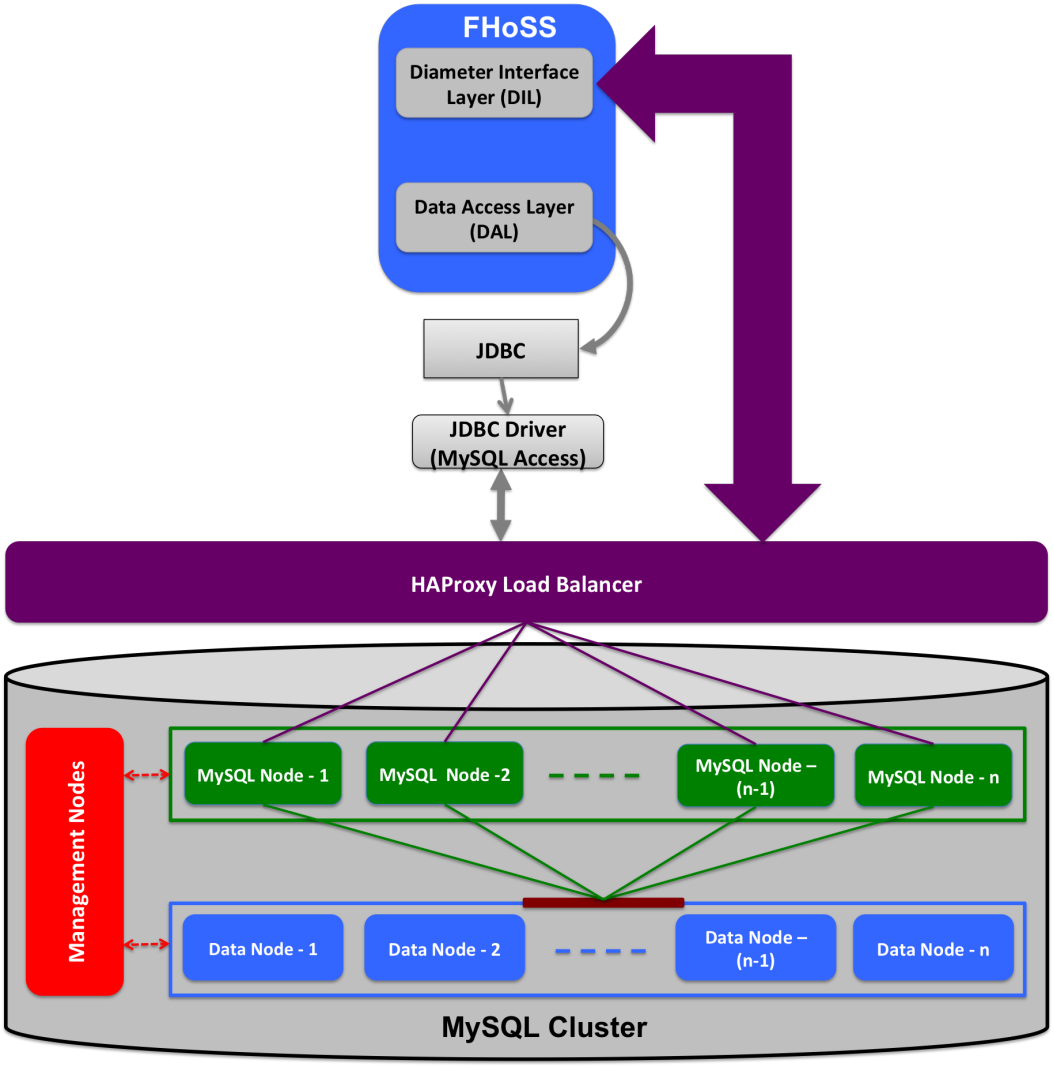


Figure 5.7. High Level Integration Architecture of FHoSS diameter interface layer and MySQL Cluster

5.2.4.1 HAProxy with Keepalived

One of the major goals of this thesis project was to designed a highly available and scalable database system architecture. MySQL cluster technology provides a high availability and scalability database. The use of a single HAProxy load balancer instance in front of a MySQL cluster creates a single point of failure for the entire database system, as if the HAProxy load balancer fails, then there is no longer access to the MySQL cluster.

To avoid a single point of failure, a passive architecture is used in conjunction with an open source software package as an solution to produce a high availability HAProxy load balancer. There are several open source software packages that ensure the high availability of a load balancer service. One method used in a passive mode was described by MySQL [55], while S. Galiano [53] used a Linux heartbeat to ensure high availability of the database service. In [55], a MySQL asynchronous replication method is combined with a Linux heartbeat using a master/slave (passive) architecture for high availability of MySQL servers. In our project, we are using another open source daemon "Keepalived" [57], to ensure high availability of the load balancer service. The keepalived daemon provides robust facilities for load balancing and high availability for a Linux system.

In a passive system architecture two instance of the HAProxy load balancer together with a Keepalived daemon are running on two separate virtual machines, one pair acts as a master and the other pair acts as a slave (backup). Keepalived implements a Virtual Router Redundancy Protocol (VRRP) for failover [58]. The two VRRP instances monitor and synchronize their states. Both HAProxy load balancer instances share a single virtual IP address, hence there will only be one load balancer visible to the outside world at any one time. If the master HAProxy load balancer node crashes, then the backup HAProxy load balance (slave) node automatically takes over the load via the virtual IP address and resumes load balancing. Figure 5.8 illustrates the high level passive architecture of the load balancer integrated with the FHoSS Diameter interface layer and a MySQL cluster.

Keepalived offers several benefits while adopting a passive architecture for high availability of HAProxy load balancer. These benefits are:

- Open source software,
- Easy to configure,
- No separate hardware or networking is required,
- Automatic management of a single virtual IP address, and
- The Virtual IP address enables transparent fail over from one active load balancer instance to another when employing a passive architecture.

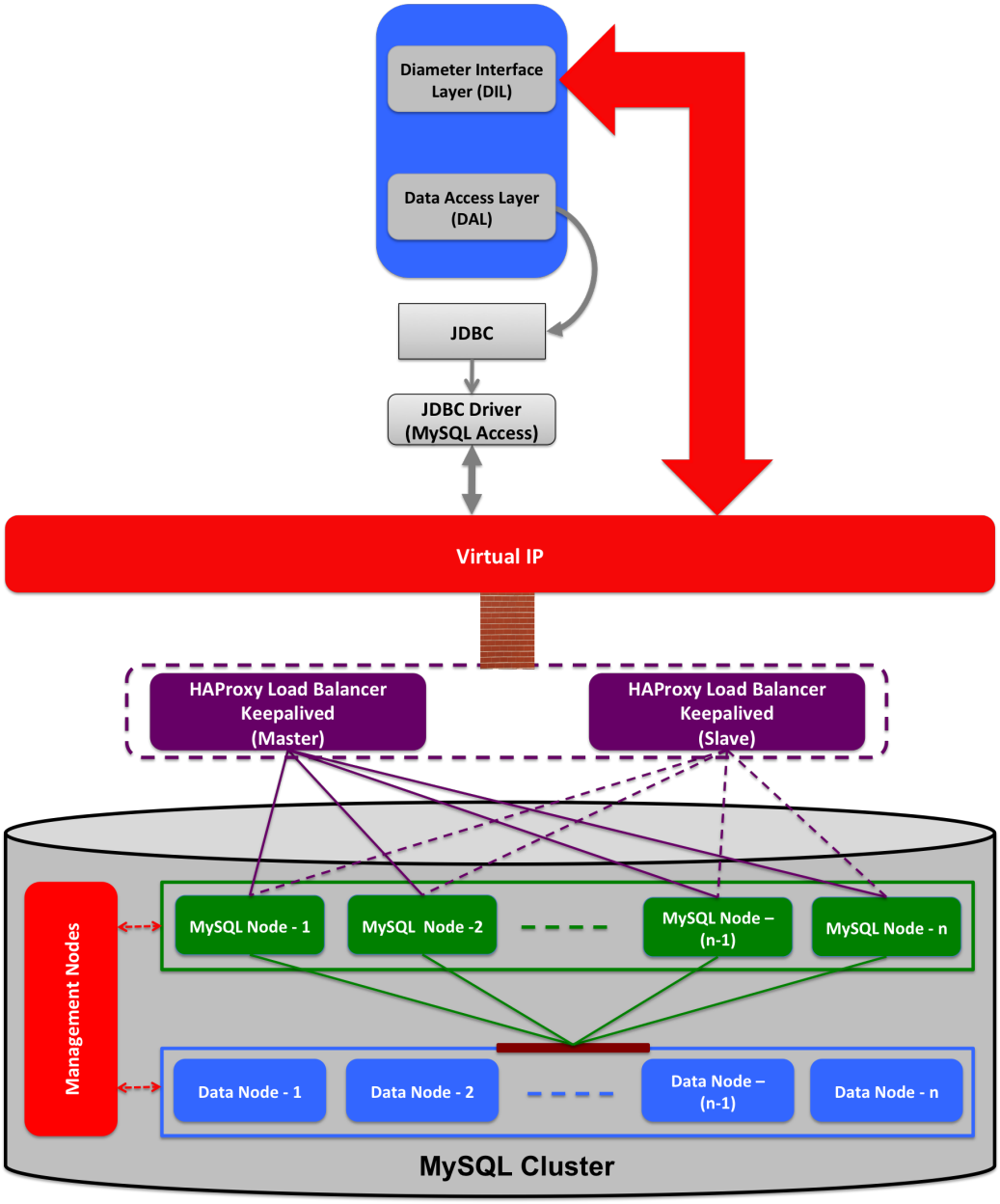


Figure 5.8. High level passive architecture of a HAProxy load balancer integrated with the FHoSS Diameter interface layer and a MySQL cluster

The keepalived configuration file for the master and slave nodes are shown in Listing 5.1 and Listing 5.2, respectively. Both nodes (master and slave) have an identical virtual IP address <ip address>. This virtual IP address was configured in the FHoSS "hibernate.properties" file.

Listing 5.1. Example of an Keepalived Configuration for Master Node

```
vrrp_script chk_haproxy {
        script  ''killall -0 haproxy''
        interval 2
        weight 2
}

vrrp_instance VI_1 {
        interface eth0
        state MASTER
        virtual_router_id 51
        priority 101
        virtual_ipaddress {
                <ip address>
        }
        track_script {
                chk_haproxy
        }
}
```

Listing 5.2. Example of an Keepalived Configuration for Slave Node

```
vrrp_script chk_haproxy {
        script ''killall -0 haproxy''
        interval 2
        weight 2
}

vrrp_instance VI_1 {
        interface eth0
        state MASTER
        virtual_router_id 51
        priority 100
        virtual_ipaddress {
                <ip address>
        }
        track_script {
                chk_haproxy
        }
}
```

Chapter 6

Testing and Evaluation

This chapter presents the functional testing of the virtualized OpenIMS core components and a performance evaluation using the virtualized OpenIMS core testbed infrastructure.

6.1 Functional Testing of OpenIMS Core Virtualization

The functional testing of the OpenIMS Core mainly focused on testing the virtualized OpenIMS Core within the testbed environment that we setup. This functional testing involves testing of the registration, de-registration, and call setup procedures between two IMS subscribers. In functional testing, all three use cases described in Sections 5.1.1, 5.1.2, and 5.2.4 were tested using the registration, de-registration, and call setup scenarios defined in IMS Bench SIPp. In each of the testbeds which were used, all of the components were configured and deployed on separate virtual machines. The specification of these virtual machine in terms of RAM and CPU is shown in Table 6.1.

Table 6.1. Virtual Machine Specification

Type	Amount
RAM	512 MB
CPU Cores	1

The testbeds for OpenIMS core virtualization for functional testing are shown in Figures 6.1, 6.2, and 6.3. In the testbed shown in Figure 6.1 only the OpenIMS core nodes are virtualized. The OpenIMS core with a virtualized MySQL database testbed is shown in Figure 6.2. In this second testbed, the MySQL database is running on a separate virtual machine, which enables further visualization of FHoSS. The testbed for a OpenIMS core integrated with a MySQL cluster is shown in Figure 6.3. In this third testbed, a MySQL cluster is deployed and integrated with FHoSS via a HAProxy load balancer. The MySQL cluster configuration is shown in Table

6.2. Each of the MySQL cluster nodes was configured to execute on separate virtual machines.

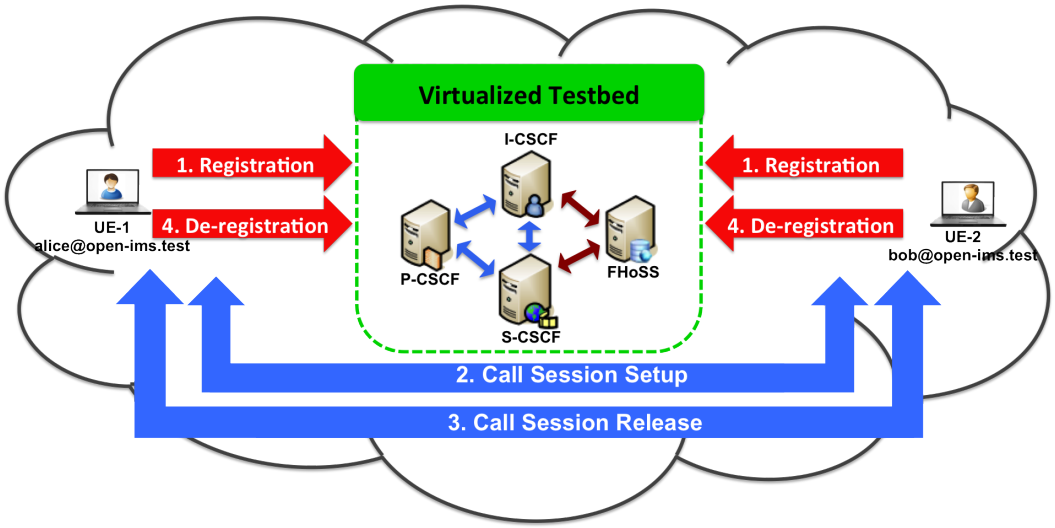


Figure 6.1. Virtualized OpenIMS core testbed

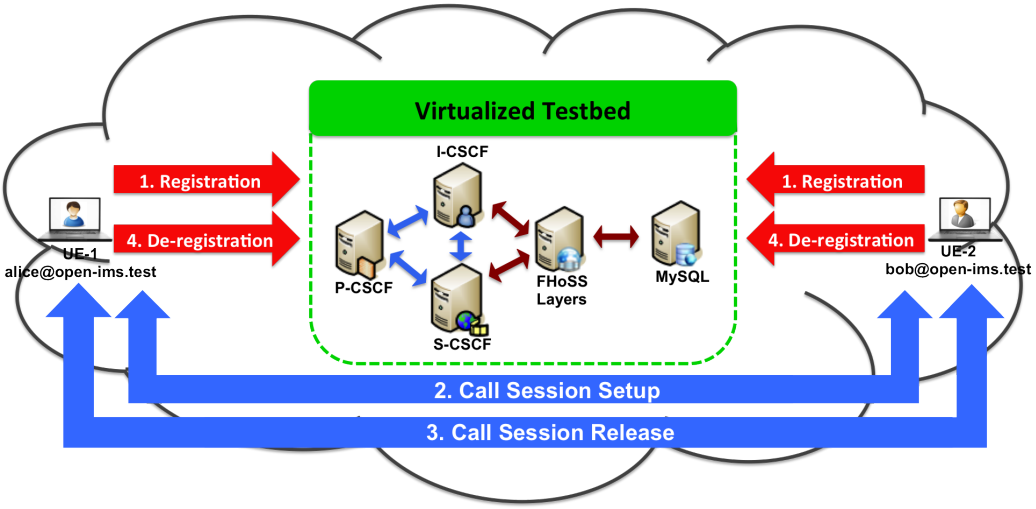


Figure 6.2. Virtualized OpenIMS core with MySQL DB virtualization testbed

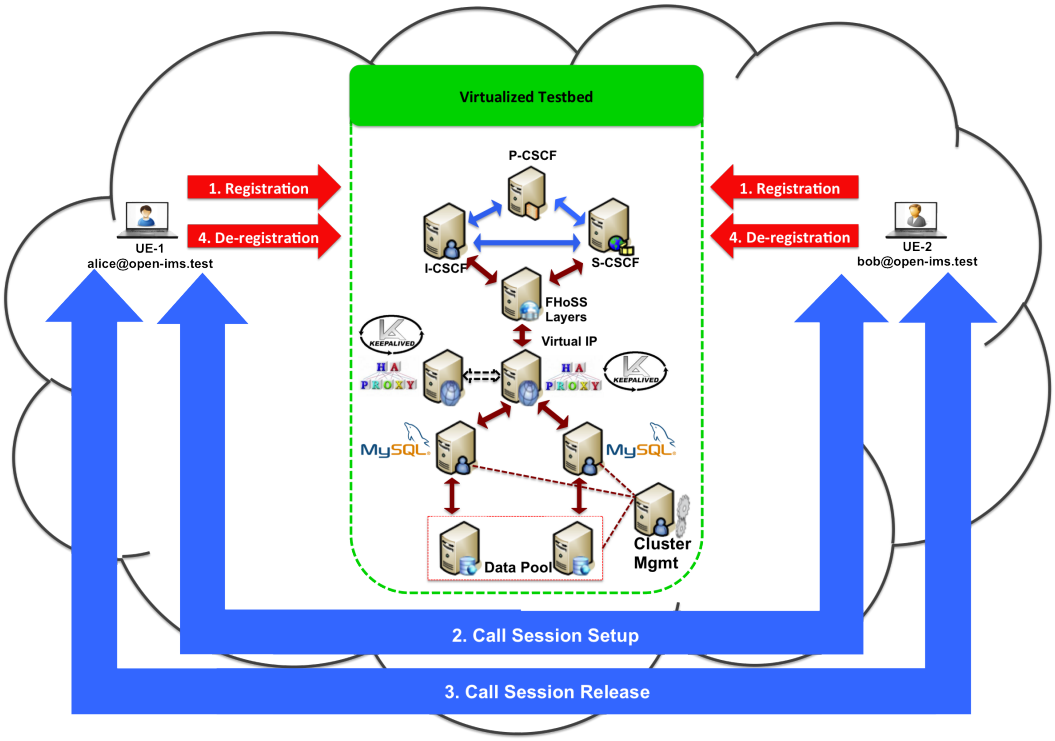


Figure 6.3. Virtualized OpenIMS core with integration of MySQL cluster testbed

Table 6.2. MySQL Cluster Configuration

Node Type	Number of Nodes
Data Node	2
MySQL Node	2
Management Node	1

In [59], OpenIMS core functionality was tested with SIP based soft phones, which are KPhone and Open SIP Client. For all of the functional testing use cases, an open source SIP based soft phone [60], called myMonster Telco Communicator Suit, was used to test the basic SIP methods.Two default users (Alice and Bob) were configured by using the myMonster client. First, Alice and Bob register with the OpenIMS core, then both originate a voice call to each other. After sometime, one or the other ends the voice call and un-registers from the OpenIMS core. All the running testbed VMs on a single physical machine are shown in Figure 6.4.

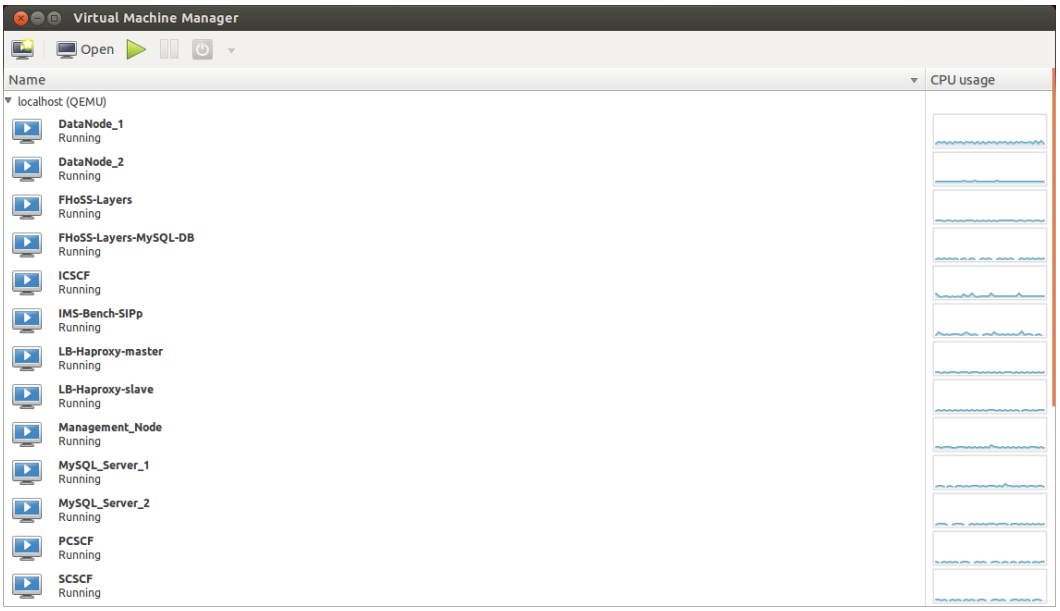


Figure 6.4. VMM screen of testbed VMs

6.1.1 IMS registration

IMS registration involves an exchange of messages between a user agent and a registrar as shown in Figure 6.5. In the testbed, the myMonster client acts as a user agent and the virtualized OpenIMS core acts as a registrar.

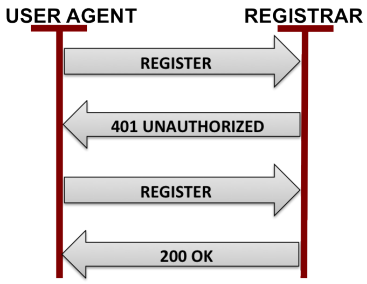


Figure 6.5. Basic IMS registration procedure (Adapted from Figure 2.1 of [61])

During registration, both clients (Alice and Bob) send a SIP Register request to the OpenIMS core and wait for a 200 OK registration response, which indicates that the client has successfully registered with the IMS core. Figure 6.6 illustrates the sequence diagram of the IMS registration procedure. An example of a SIP REGISTER request message is shown in Listing 6.1.

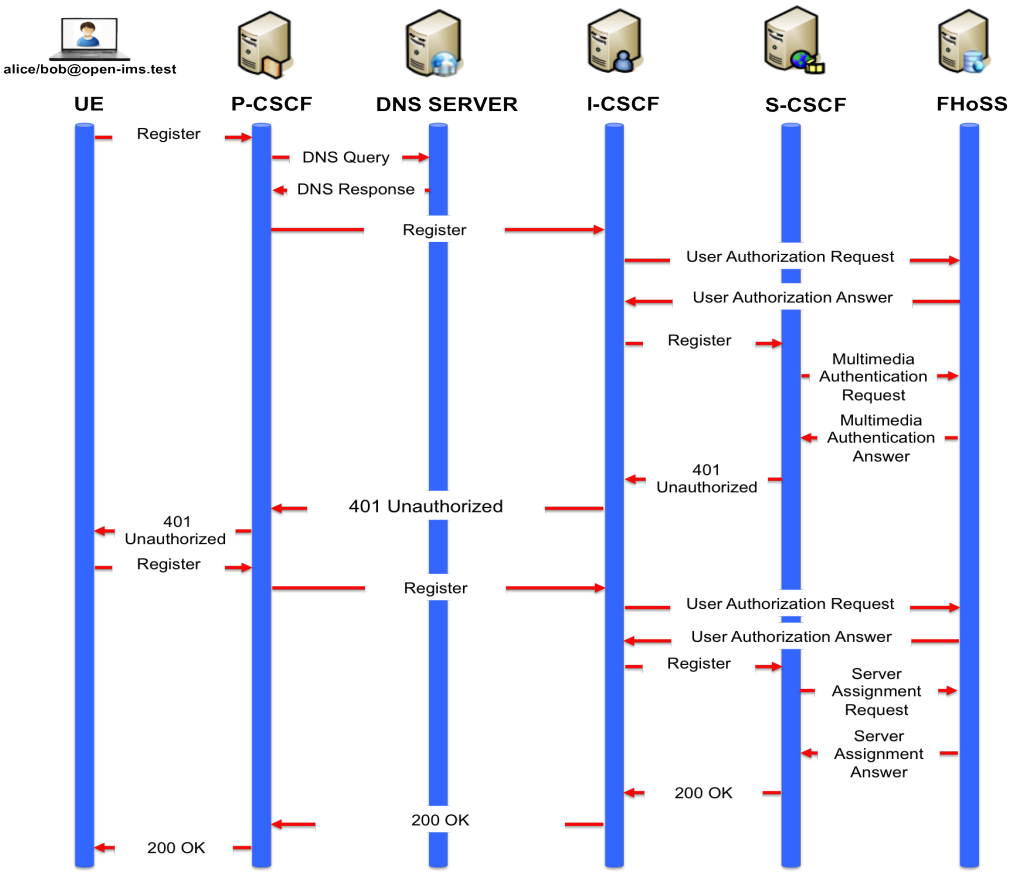


Figure 6.6. IMS registration sequence diagram

Listing 6.1. Example of a SIP REGISTER request

```
REGISTER sip:open-ims.test SIP/2.0
Call-ID: 4b2cee039c17f776ec86467645ae5105@192.168.122.10
CSeq: 10 REGISTER
From: ‘‘Alice’’ <sip:alice@open-ims.test>;tag=1008
To: ‘‘Alice’’ <sip:alice@open-ims.test>
Via: SIP/2.0/UDP 192.168.122.10:5060;
branch=z9hG4bK6c680698f13aa51d35c8725089ebc376333030
Max-Forwards: 20
Expires: 3600
Authorization: Digest username=‘‘alice@open-ims.test’’,
realm=‘‘open-ims.test’’,nonce=‘‘’’,response=‘‘’’,
uri=‘‘sip:open-ims.test’’
Contact: ‘‘Alice’’ <sip:alice@192.168.122.10:5060>;
+sip.instance=d9f148fa-96e8-49f8-bec1-b5e787b5c021
User-Agent: monster Version: 0.9.25
Content-Length: 0
```

6.1.2 IMS call setup

The IMS Call Setup procedure involves an exchange of messages between a caller and a callee as shown in Figure 6.7. In the testbed, Alice either acts as a caller and Bob acts as a callee or vice versa. After successful registration by both Alice and BoB, both user agents will able to receive a voice call (or initiate a voice session) by accepting a SIP INVITE from the other user agent. The call setup session sequence diagram is shown in Figure 6.8. An example of a SIP REGISTER request message is shown in Listing 6.2.

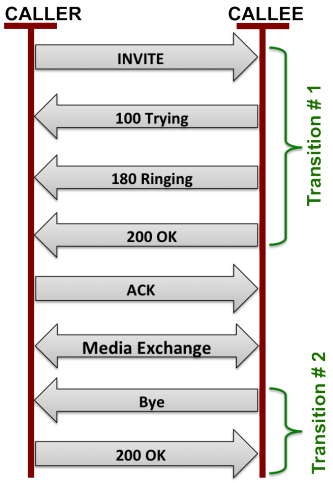


Figure 6.7. Basic IMS Call Setup Procedure (Adapted from Figure 3.1 of [61])

Listing 6.2. Example of a SIP INVITE Request

```
INVITE sip:alice@open-ims.test SIP/2.0
Call-ID: 4b2cee039c17f776ec86467645ae5105@192.168.122.10
CSeq: 1 INVITE
From: ‘‘Alice’’ <sip:alice@open-ims.test>;tag=1008
To: ‘‘Bob’’ <sip:bob@open-ims.test>
Via: SIP/2.0/UDP 192.168.122.10:5060;
branch=z9hG4bK6c680698f13aa51d35c8725089ebc376333030
Max-Forwards: 70
Contact: ‘‘Alice’’ <sip:alice@192.168.122.10:5060>;
Route: <sip:pcscf.open-ims.test:4060;transport=udp>,
<sip:orig@scscf.open-ims.test:6060;lr>
Allow: INVITE,ACK,CANCEL,BYE,MESSAGE,PRACK,UPDATE
P-Preferred-Identity: <sip:alice@open-ims.test>
Privacy: none
Require: precondition
Supported: 100rel,precondition,early-session
P-Access-Network-Info: IEEE-802.11
User-Agent: monster Version: 0.9.25
Content-Length: 674
```

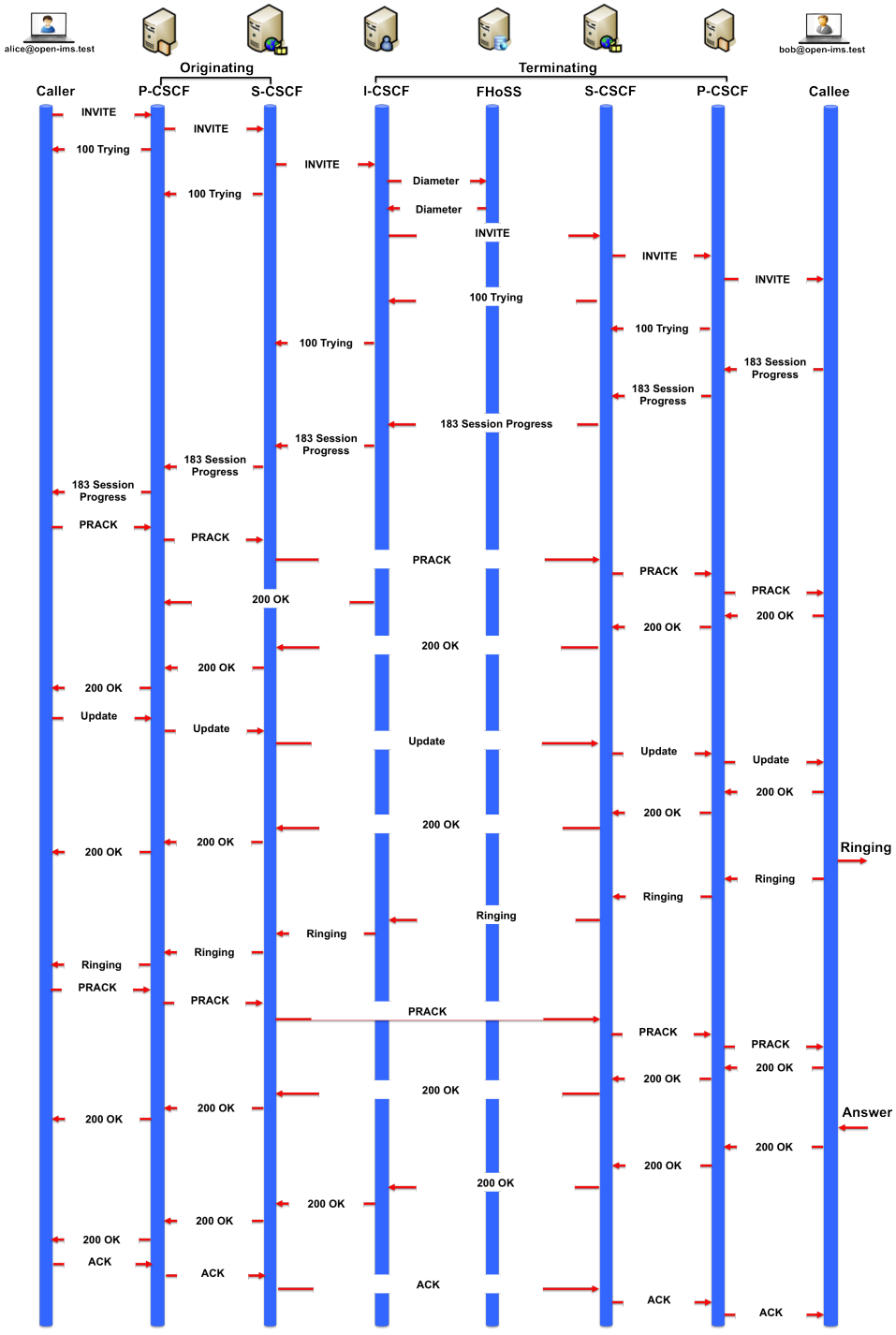
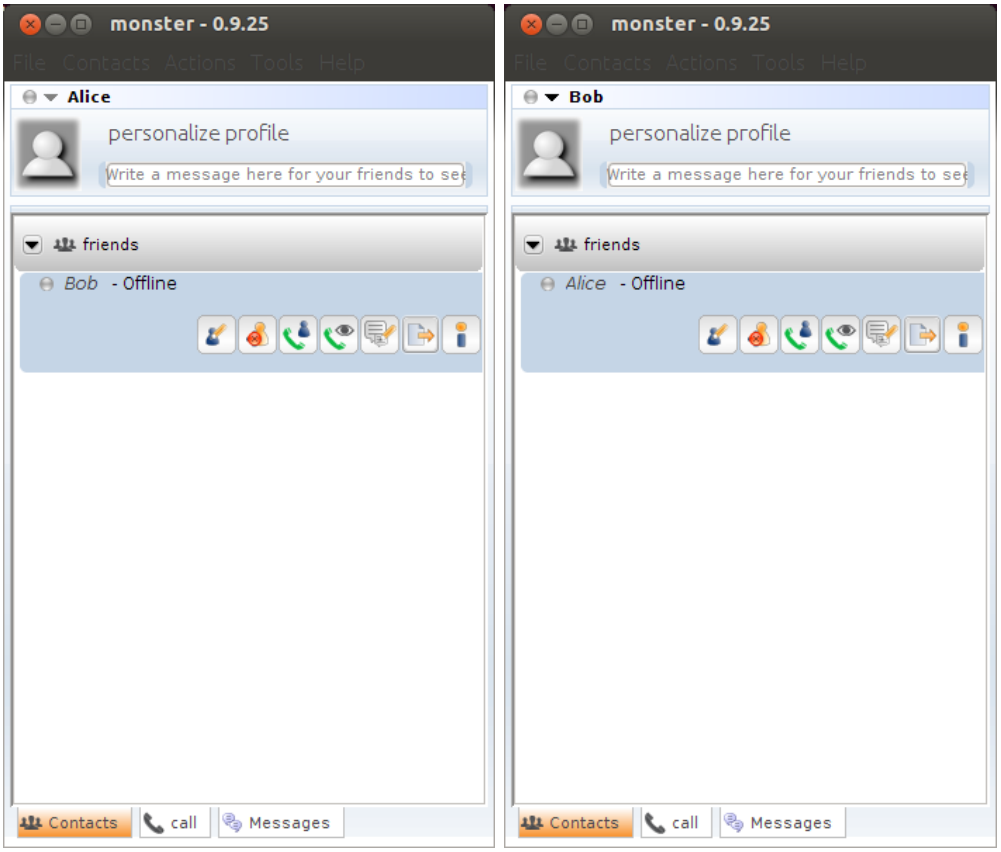


Figure 6.8. IMS Call sequence diagram

6.1.3 Functional testing results

The following are the results we had achieved after performing the functional testing of IMS cloud testbed:

- Two user profiles for Alice and Bob was created by using an "myMonster". Configuration of the IMS network for creating user profiles is described in Appendix G.1. Alice and Bob had successfully performed registration to OpenIMS cloud network. Figure 6.9 shows the Alice and Bob myMonster client window after successfully registration.



(a) Alice Registration (b) Bob Registration

Figure 6.9. myMonster Registration windows

Wireshark (http://www.wireshark.org) is one of world's foremost open source network protocol analyzer. Wireshark was used in this thesis project for analyzing SIP and Diameter protocol messages. Figure 6.10 shows the Wireshark trace of the SIP registration messages.

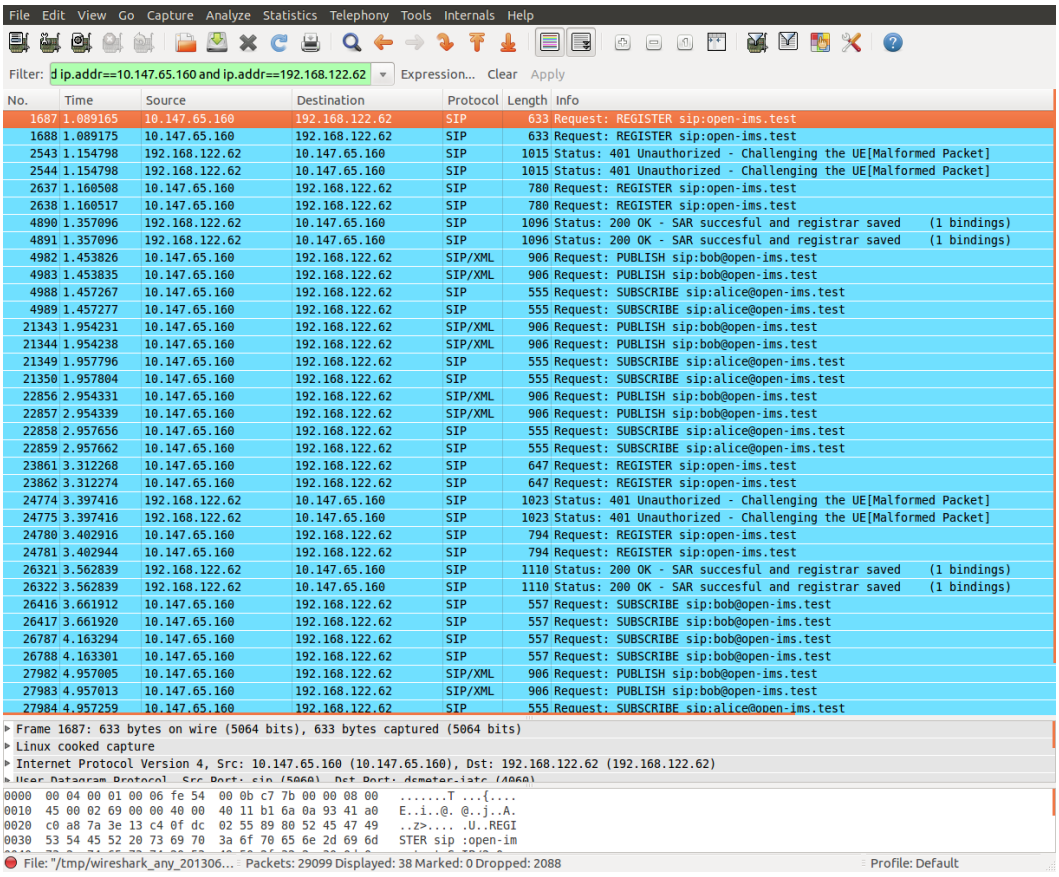
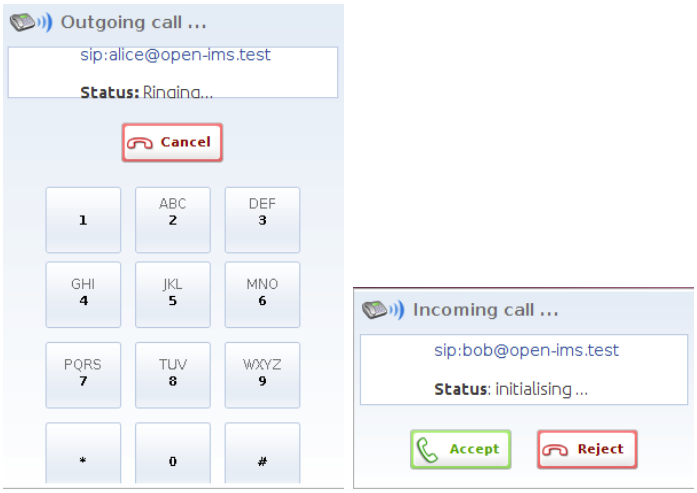


Figure 6.10. Wireshark Trace of SIP Registration messages

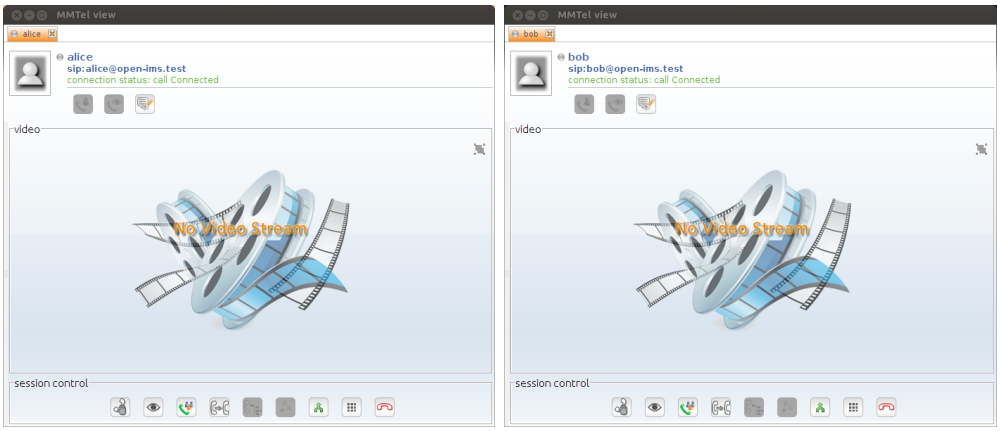
- After successfully registered, Alice initiates audio call to Bob, and waits for Bob to accept the incoming call. Figure 6.11(a) illustrates the myMonster outgoing call window. Bob receives an incoming call from Alice. Figure 6.11(b) illustrates the myMonster incoming call window. Bob can accept or reject the Alice's call.



(a) Outgoing Call (b) Incoming Call

Figure 6.11. myMonster outgoing and incoming call windows

Bob accepts an Alice's call by pressing accept button, then a session between Alice and Bob is created, the resultant windows are shown in Figure 6.12. Alice or Bob can terminate the session by just closing their windows.



(a) Alice's Call Session (b) Bob's Call Session

Figure 6.12. myMonster outgoing and incoming call session windows

Figure 6.13 shows the Wireshark trace of the SIP INVITE messages.

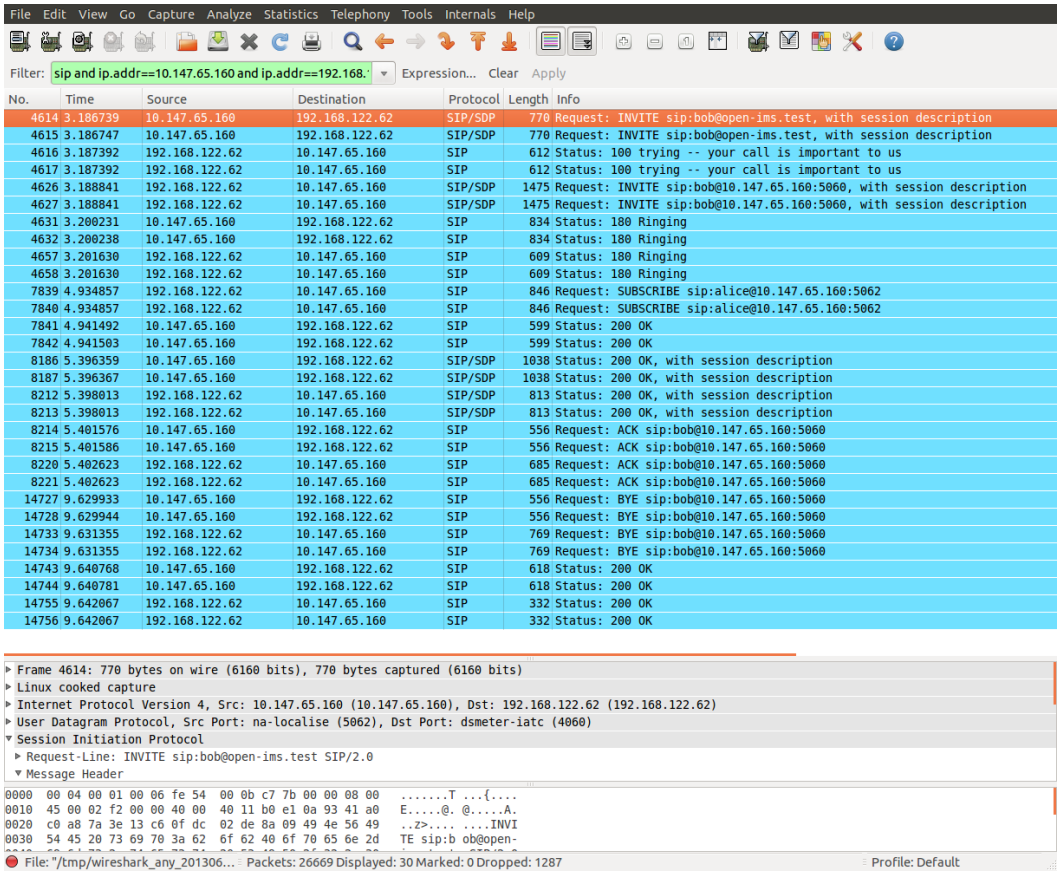


Figure 6.13. Wireshark Trace of SIP INVITE messages

6.2 Performance Evaluation

Performance testing on a telecommunication platform implies the simulation of realistic scenarios. This section described the performance testing that was conducted by using an IMS Bench SIPp test system.

6.2.1 TestBed description

The hardware specification of physical machine used for realizing an IMS cloud testbed is presented in Table 6.3. The operating system of all the virtual machines was Ubuntu Server 12.04.2 LTS. All virtual machines of the IMS cloud testbed were interconnected via in the same private network.

Table 6.3. Physical Machine Hardware Specification

Components	Description
Processor	Intel® Xeon® W3530 @ 2.80 GHz CPU 1596 MHz
Memory	12 GB (3 * 4 GB) DDR3 @ 1600 MHz
Hard Disk	ST3320413AS SATA 320 GB 7200 rpm

For the purpose of real time monitoring of KPIs (CPU usage, Memory, etc.) of the SUT virtual machines during each benchmark run, the Zabbix server was configured on a separate VM and a Zabbix agent was installed in every SUT VM, and these Zabbix agents transmit the latest KPI data of each VM to the Zabbix server.The Zabbix (server and agent) configuration is described in Appendix E. KIP data was collected at Zabbix server node. After the completion of benchmark testing, all the data within the simulated range of timestamps was fetched using scripts [62] in a format of "csv" files. The KPI plots were generated from these "csv" files using MATLAB.

The test system, consisted of a single virtual machine running the IMS bench manager instance and a single SIPp load generator instance. The instructions for running such a manager and SIPp are described in Appendix C. To perform benchmarking, 10,000 subscribers were created in the FHoSS database using a built-in script. All 10,000 subscribers have generated names (public/private user identity) ranging from subs000000 to subs999999 and are in the IMS domain named "open-ims.test". Figure 6.14 shows the FHoSS management console. Additionaly, the following performance parameters was enabled for a MySQL cluster:

- Auto-sharding
- Ndb_cluster_connection_pool = 4

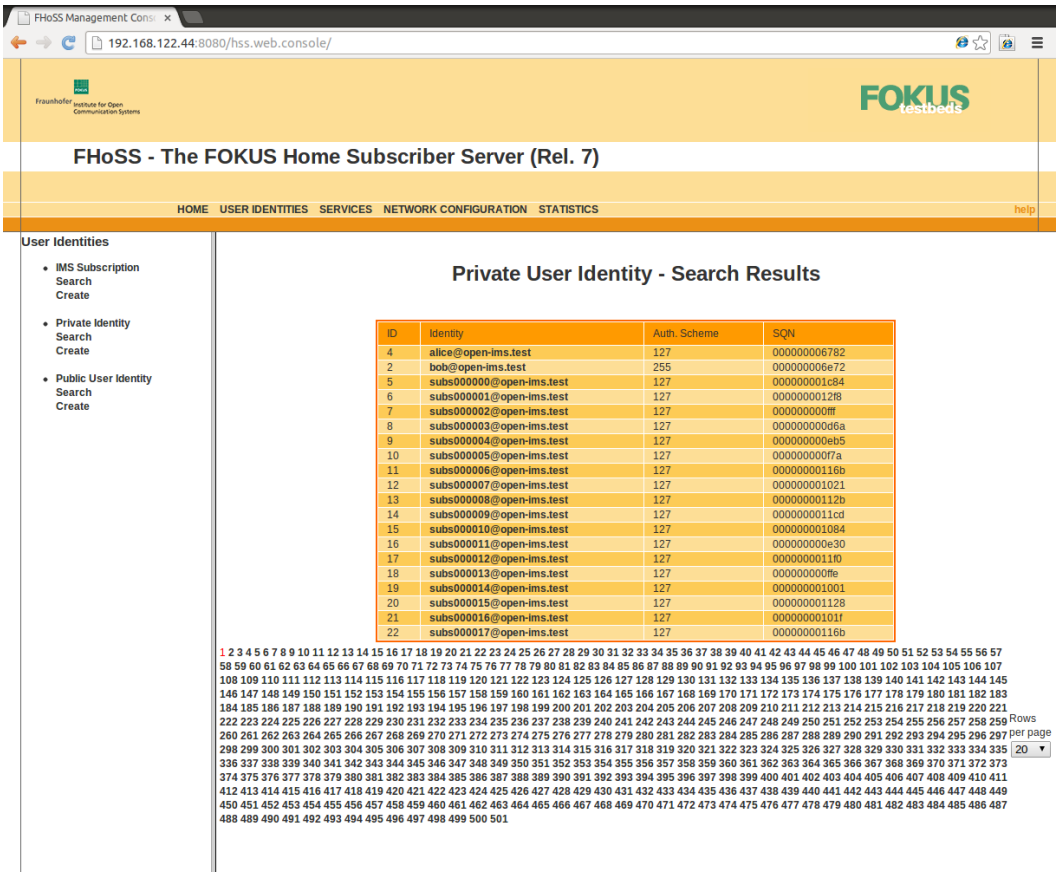


Figure 6.14. FHoSS Management Console

6.2.2 Report Generation Tool

The IMS Bench SIPp built-in tool was used for report generation of our performance benchmarking. Report generation scripts were executed after completing each benchmark test to create a complete summery of the benchmark results in Microsoft Hypertext Archive (MHT) and HyperText Markup Language (HTML) formats. During benchmark testing data was automatically collected in the form of ".csv" files by the manager. The commands shown in Listing 6.3 fetch the results collected during the benchmarking test, and than generate the actual report.

Listing 6.3. Report generation commands

```
pwd
/root/ims_bench/ims_bench_7
../scripts/getResults.pl
../scripts/doReport.pl -r report.xml -c ../scripts/reportConfig.xml
-i ims_bench.xml
```

6.2.3 Benchmark Simulation Setting

The SUT's performance was analyzed for the three different IMS cloud testbed setups described above with respect to the parameter: Scenarios Attempts Per Second (SAPS). Different scenarios with a defined probability of occurrence are initiated during each benchmarking step. The probabilities of each scenario were configured in the benchmark configuration file (i.e. "manager.xml"). Figure 6.15 shows the manager benchmark configuration setting that was used for all the three IMS cloud setups that were tested.

The pre-registration phase of the benchmarking test was configured to start at a rate of 5 cps until a maximum of 300 subscribers were registered during the pre-registration phase. The constant distribution was used for scenario initiation over time and the maximum global of Inadequately Handled Scenarios (IHS) was set to 10%. If the number of actual IHS exceeds 10% the testing would be stopped and an indication that an error had occurred would be output.

The stir phase of the benchmarking test was configured to start with a single instantaneous increase in rate of calls to 10 cps. The duration of this step-wise was 300 seconds, Poisson distribution was used for scenarios (registration, re-registration, de-registration, call session, and message) initiation over time, and the maximum global IHS was again limited to 10%.

The benchmark run phase of the test was configured to start with a calling rate of 15 cps, followed by 5 steps which increased the calling rate by 5 cps at each step. Poisson distribution was used for scenarios (registration, re-registration, de-registration, call session, and message) initiation over time and the IHS threshold of 1% of (registration, re-registration, de-registration, call session, and message) scenario was used instead of maximum global IHS. Therefore, the benchmark execution automatically stopped when the average percentage of IHS in any scenario exceeded the threshold of 1%.

The scenarios in both stir and benchmark run phases were configured with a defined probability of occurrence, specifically the registration scenario of 30%, the re-registration scenario of 3%, the call session scenario of 20%, the de-registration scenario of 7%, and the message scenario of 40%. Additionally, the scenario percentages was chosen by the author and the percentage are not suggested in ETSI's IMS/NGN Performance Benchmark specification.

root@mum: ~/ims_bench/ims_bench_7

```
<?xml version="1.0" encoding="ISO-8859-1" ?><configuration>
  <!-- Test System Parameters -->
  <param name="number_test_systems" value="1"/>
  <param name="prep_offset" value="2000"/>
  <param name="rand_seed" value="0"/>
  <param name="report" value="1"/>
  <param name="log" value="1"/>
  <param name="transient_time" value="30"/>
  <param name="max_time_offset" value="250"/>
  <!-- Scenario Parameters -->
  <scen_param name="RegistrationExpire" value="1000000"/>
  <scen_param name="RingTimeDistr" value="exponential"/>
  <scen_param name="PMMDataSize" value="140"/>
  <scen_param name="HoldTimeDistr" value="exponential"/>
  <scen_param name="HoldTime" value="120000"/>
  <scen_param name="PMMDataSizeDistr" value="uniform"/>
  <scen_param name="RingTime" value="5000"/>
  <!-- Scenario -->
  <scenario name="ims_reg" max_ihs="1"/>
  <scenario name="ims_uac" max_ihs="1"/>
  <scenario name="ims_uas"/>
  <scenario name="ims_dereg" max_ihs="1"/>
  <scenario name="ims_msgc" max_ihs="1"/>
  <scenario name="ims_msgs"/>
  <scenario name="ims_rereg" max_ihs="1"/>
</configuration>
<!-- Pre-Registration Phase -->
<run cps="5" max_calls="300" distribution="constant" sync_mode="off" use_scen_max_
ihs="no" max_global_ihs="10" stats="1000">
   <scenario name="ims_reg" ratio="100"/>
</run>
<!-- Stir Phase -->
<run cps="10" duration="300" step_increase="2" num_steps="1" distribution="poisson
" use_scen_max_ihs="no" max_global_ihs="10" stats="1000" report="no">
  <scenario name="ims_reg" ratio="30"/>
  <scenario name="ims_rereg" ratio="3"/>
  <scenario name="ims_uac" ratio="20"/>
  <scenario name="ims_dereg" ratio="7"/>
  <scenario name="ims_msgc" ratio="40"/>
</run>
<!-- Benchmark Run Phase -->
<run cps="15" duration="300" step_increase="5" num_steps="5" distribution="poisson
" stats="2000">
  <scenario name="ims_reg" ratio="30"/>
  <scenario name="ims_rereg" ratio="3"/>
  <scenario name="ims_uac" ratio="20"/>
  <scenario name="ims_dereg" ratio="7"/>
  <scenario name="ims_msgc" ratio="40"/>
</run>

<!-- Final Pause - Required to allow last step data collection to complete -->
<run cps="0" duration="3"/>
                                                              27,1          All
```

Figure 6.15. Manager benchmark configuration file

6.2.4 Analysis of first testbed scenario

The testbed setup for benchmark testing is shown in Figure 6.16.

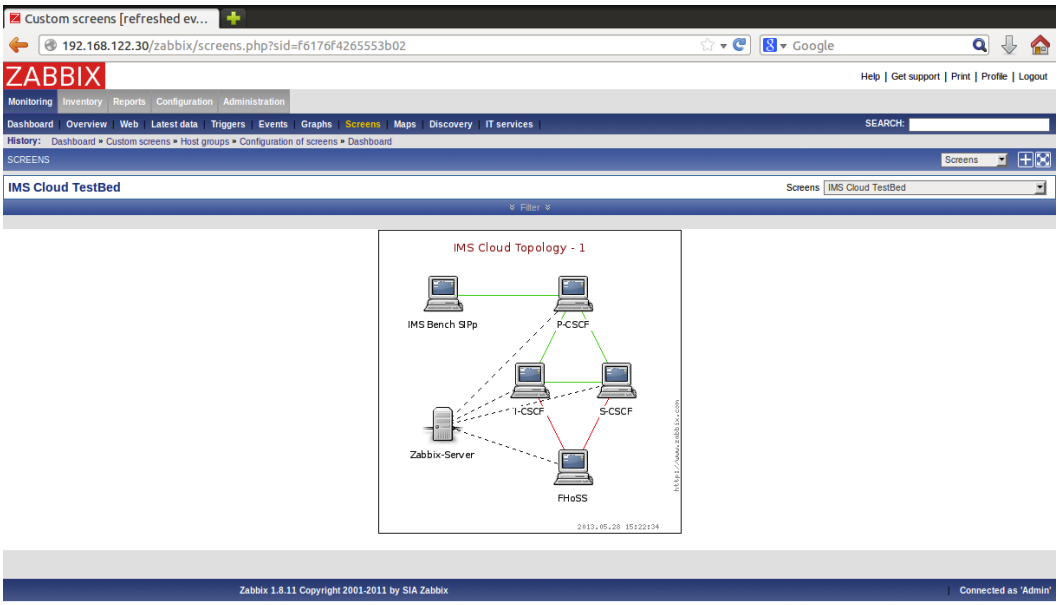


Figure 6.16. First Testbed Topology

The key summary of the benchmarking results for the first testbed topology is shown in Table 6.4. Each step of the benchmarking test is categorized by the request load, the effective load, the global IHS (the global IHS is defined as the sum of all the IHS for this step divided by the number of scenario attempts for this step), the scenario's IHS (number of IHS of this step divided by the number of scenario attempts for this step), the CPU utilization, and the available memory. The requested and effective loads are expressed in SAPS, and the available memory in megabytes (MB).

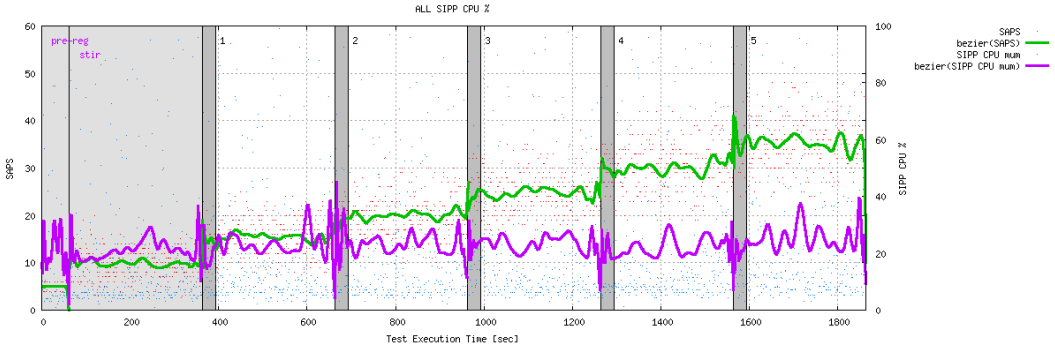
Table 6.4. Summary of Benchmarking Test

	Pre-reg	Step - 1	Step - 2	Step - 3	Step - 4	Step - 5
Requested Load	5	15	20	25	30	35
Effective Load	4.90	15.37	19.89	24.68	30.04	35.33
Ratio ims_reg %	100.00	28.76	30.12	29.75	29.97	30.68
Ratio ims_uac %	0.00	9.83	9.89	9.98	9.71	10.09
Ratio ims_dereg %	0.00	6.47	6.99	6.97	7.30	6.88
Ratio ims_msgc %	0.00	52.03	50.44	50.43	49.87	49.55
Ratio ims_rereg %	0.00	2.91	2.55	2.87	3.16	2.81
SIPP CPU mum	21.09	23.01	23.45	23.57	22.14	25.00
SIPP MEM mum	69.49	68.15	70.61	70.55	69.80	72.93
IHS ims_reg %	0.00	0.00	0.00	0.00	0.12	0.17
IHS ims_dereg %	0.00	0.00	0.00	0.00	0.00	0.15
IHS ims_msgc %	0.00	0.00	0.00	0.00	0.00	0.04
global IHS %	0.00	0.00	0.00	0.00	0.04	0.08

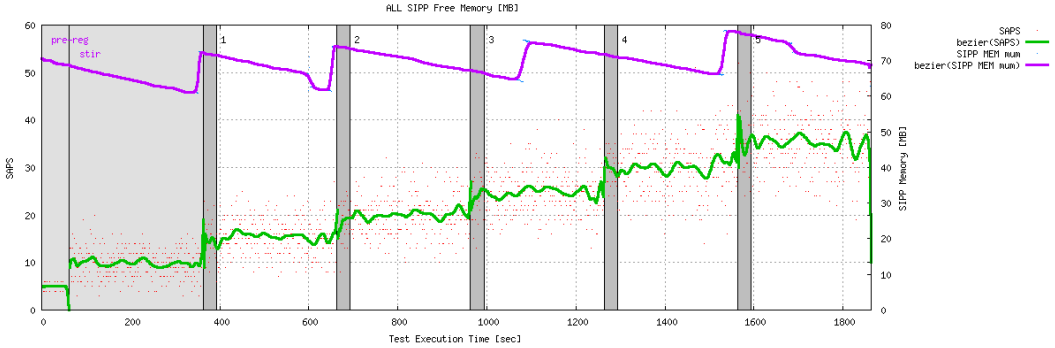
Moreover, the IHS percentage mentioned in Table 6.4 is not the IHS per seconds (average), but rather IHS is the number of failures for a step divided by the number of scenario attempts for this step.

6.2.4.1 Analysis of CPU Usage and Available Memory

Figure 6.17(a) depicts that the number of SAPS gradually increases from pre-registration phase to step-5 together with the CPU utilization of test system's VM. It can be seen that the test system (manager and SIPp instance) utilized approximately 25% of CPU on average during the benchmarking execution. The test system VM specification was described in Table 6.1. Figure 6.17(b) shows that the test system consumes varying amounts of VM memory during the execution of each phase of the benchmarking test.



(a) CPU utilization

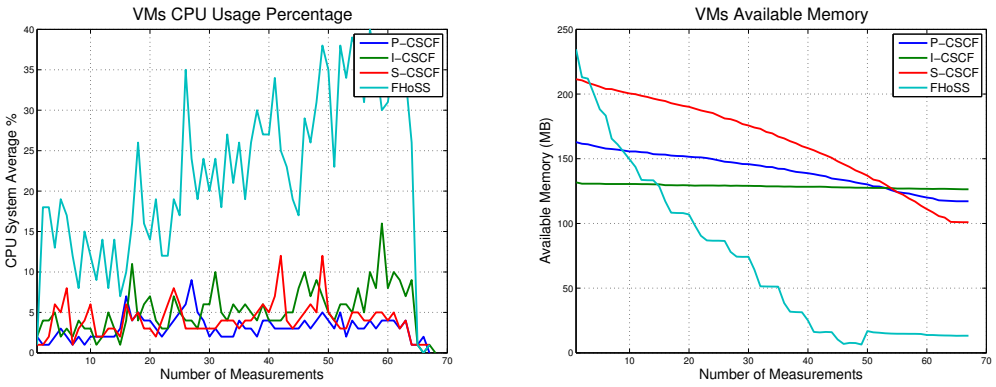


(b) Available memory

Figure 6.17. CPU utilization and available memory of the Test System's VM

The SUT VM's CPU utilization is shown in Figure 6.18(a). It can be perceived that the CPU usage of the FHoSS VM increases gradually and reached up to 40% of the CPU on average during the benchmarking. The average CPU usage of P-CSCF, I-CSCF, and S-CSCF VMs does not increase and their average CPU usage is less than 20%. Figure 6.18(b) illustrates that all the SUT VMs consumes memory

during the benchmarking except that the I-CSCF VM node has constant memory utilization. Moreover, we see that the amount of available memory in the case of the VM executing the FHoSS decreases abruptly and almost becomes constant because of the virtual memory usage of JAVA application (FHoSS) running was reached to the limit.



(a) CPU utilization (b) Available memory

Figure 6.18. CPU utilization and available memory of the SUT VMs

6.2.4.2 Analysis of Calling Scenario

IMS benchmark specification provides information about the calling scenario during the benchmark execution. Figure 6.19 shows the delay in milliseconds between the caller sending a SIP INVITE request and a callee receiving the corresponding ACK message. The x-axis represents the execution time (in seconds from the start) of benchmarking. The y-axis represents the number of SAPS on a linear scale (on the left vertical axis) and the delay on a logarithmic scale of 0.1 to 100 milliseconds (on the right vertical axis). It can be observed that the mean delay is in the range of 5 to 10 milliseconds during the entire duration of the benchmarking execution, thus we can conclude that the delay does not increase with an increase in number of SAPS during each phase of this benchmarking.

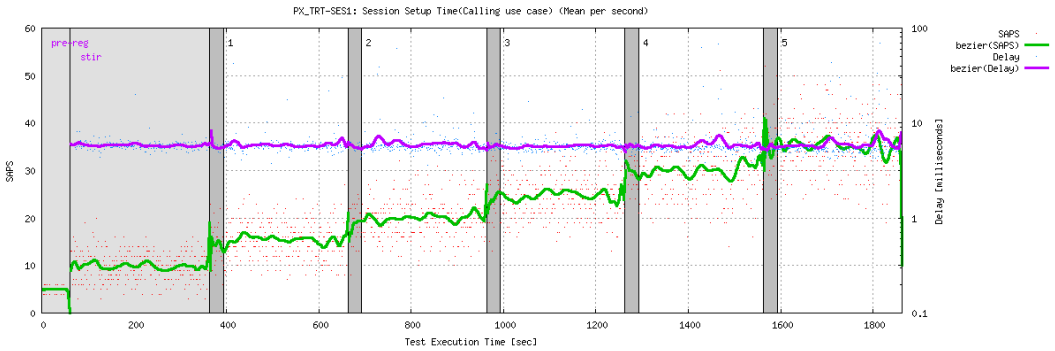


Figure 6.19. Session Setup Time

The delay in milliseconds between the caller sending a SIP INVITE request and the callee receiving a SIP INVITE request is shown in Figure 6.20. The x-axis represents the execution time (in seconds since the start) of benchmarking. The y-axis represents the number of SAPS on a linear scale (on the left vertical axis) and the delay on a logarithmic scale of 0.01 to 100 milliseconds (on the right vertical axis). It can be observed that the mean delay is in the range of 1 to 5 milliseconds during the entire duration of the benchmarking execution, thus we can conclude that the delay does not increase with an increase in number of SAPS during each phase of this benchmarking.

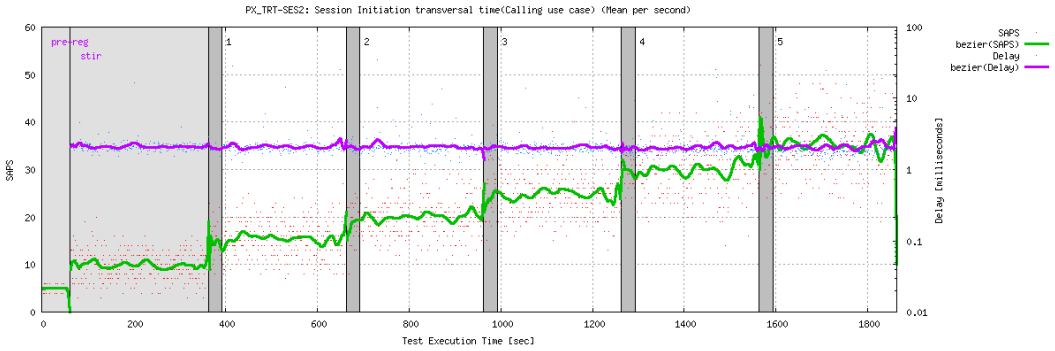


Figure 6.20. Session Initiation transversal time

The delay in milliseconds between the first BYE and the corresponding 200 OK is illustrated in Figure 6.21. The x-axis represents the execution time (in seconds since the start) of benchmarking. The y-axis represents the number of SAPS on a linear scale (on the left vertical axis) and the delay on a logarithmic scale of 0.1 to 1000 milliseconds (on the right vertical axis). It can be observed that the mean delay is in the range of 1 to 5 milliseconds during the entire duration of the benchmarking

execution, thus we can conclude that the delay does not increase with an increase in number of SAPS during each phase of this benchmarking.

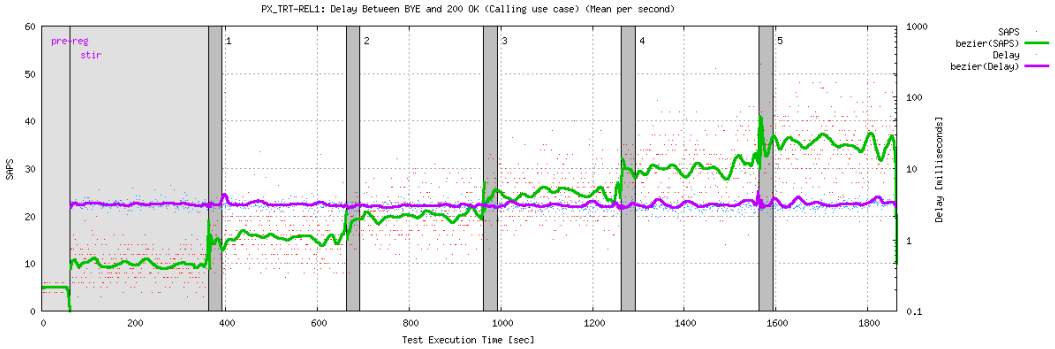


Figure 6.21. Delay Between BYE and 200 OK

6.2.4.3 Analysis of Message Scenario

IMS benchmark specification includes a message scenario. The delay in milliseconds between the sending of a message and the corresponding 200 OK is shown in Figure 6.22. The x-axis represents the execution time (in seconds since the start) of benchmarking test. The y-axis represents the number of SAPS on a linear scale (on the left vertical axis) and the delay on a logarithmic scale of 0.1 to 1000 milliseconds (on the right vertical axis). It can be observed that the mean delay is in the range of 1 to 5 milliseconds during the entire duration of the benchmarking execution, thus we can conclude that the delay does not increase with an increase in number of SAPS during each phase of this benchmarking.

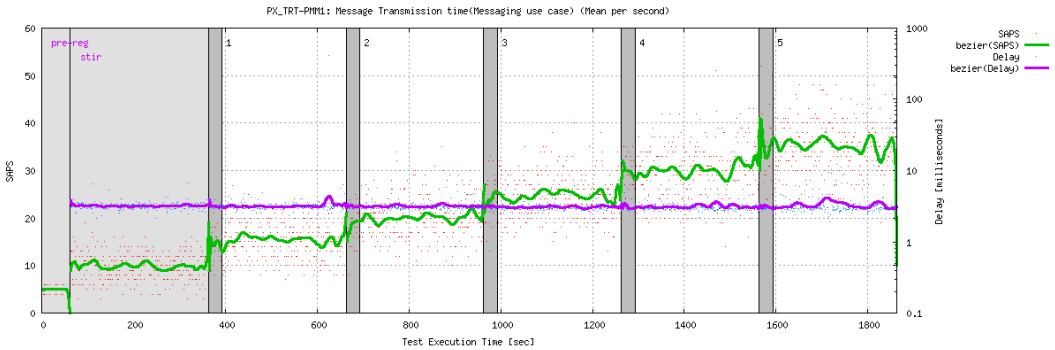


Figure 6.22. Message Transmission time

6.2.4.4 Analysis of Registration Scenario

IMS benchmark specification includes a registration scenario. The delay between the first SIP REGISTER request and the 401 Unauthorized response is shown for all the benchmark phases in the registration use case during the benchmark is shown in Figure 6.23. The x-axis represents the execution time (in seconds since the start) of benchmarking test. The y-axis represents the number of SAPS on a linear scale (on the left vertical axis) and the delay on a logarithmic scale (on the right vertical axis). It can be observed that the mean delay is in the range of 20 to 50 milliseconds during the entire duration of the benchmarking execution.

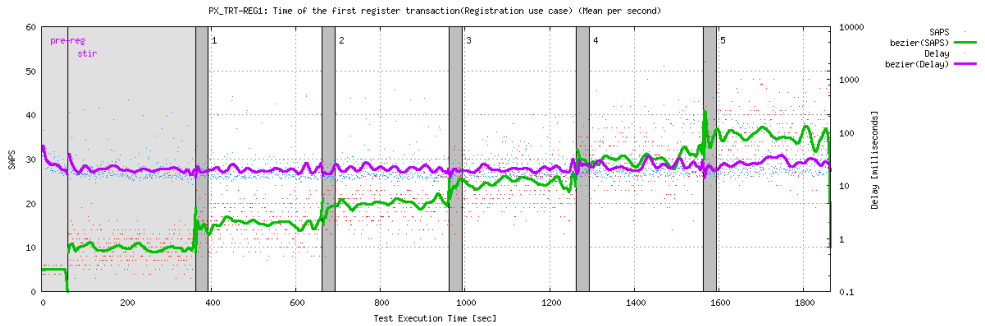


Figure 6.23. Time of the first register transaction

Figure 6.24 depicts the delay between the second REGISTER message and the corresponding 200 OK message in the registration scenario. The x-axis represents the execution time (in seconds since the start) of benchmarking test. The y-axis represents the number of SAPS on a linear scale (on the left vertical axis) and the delay on a logarithmic scale of 0.1 to 1000 milliseconds (on the right vertical axis). It can be observed that the mean delay is in the range of 20 to 40 milliseconds during the entire duration of the benchmarking execution.

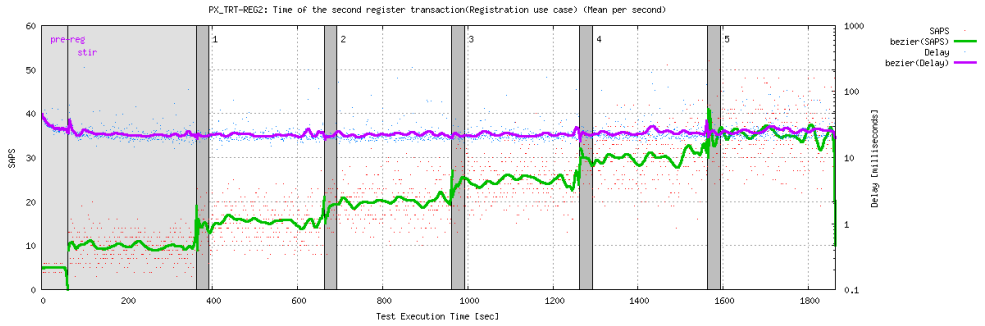


Figure 6.24. Time of the second register transaction

6.2.5 Analysis of second testbed scenario

The testbed setup for this second testbed scenario for benchmark testing is shown in Figure 6.25.

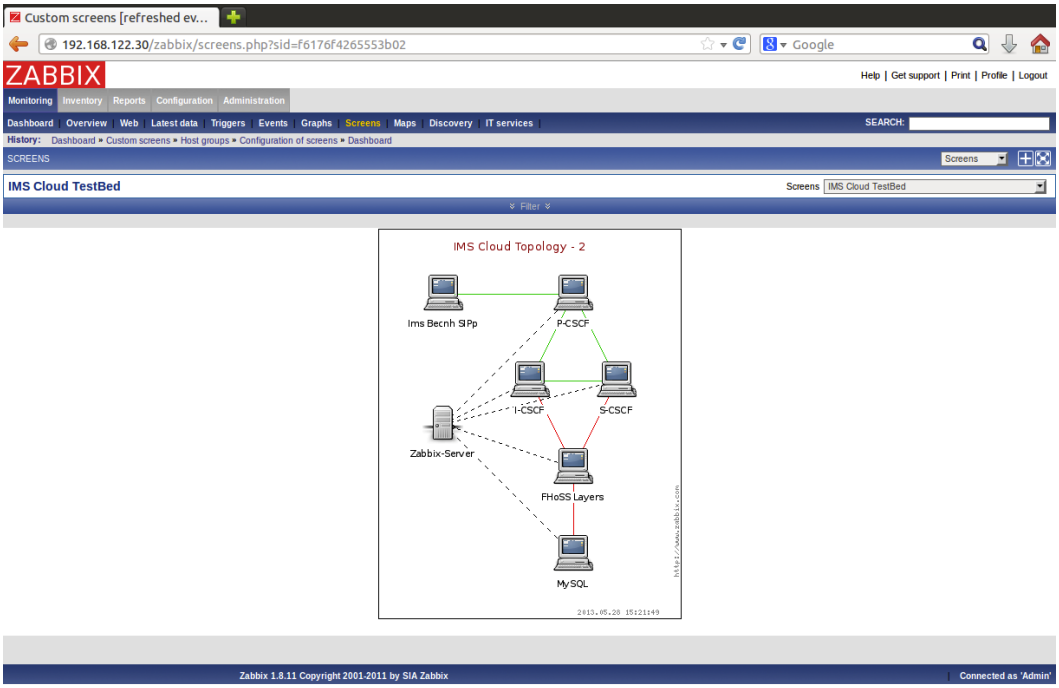


Figure 6.25. Second Testbed Topology

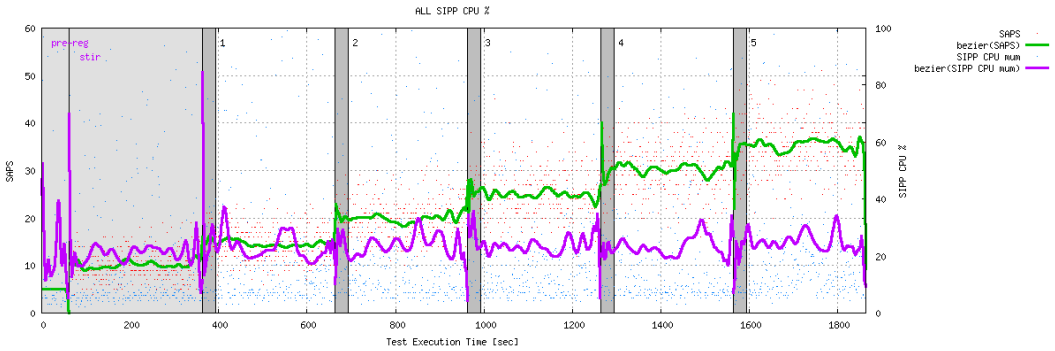
A summery of the benchmarking test results for second testbed topology is shown in Table 6.5. The terms of the table are explained in Section 6.2.4.

Table 6.5. Summary of Benchmarking Test

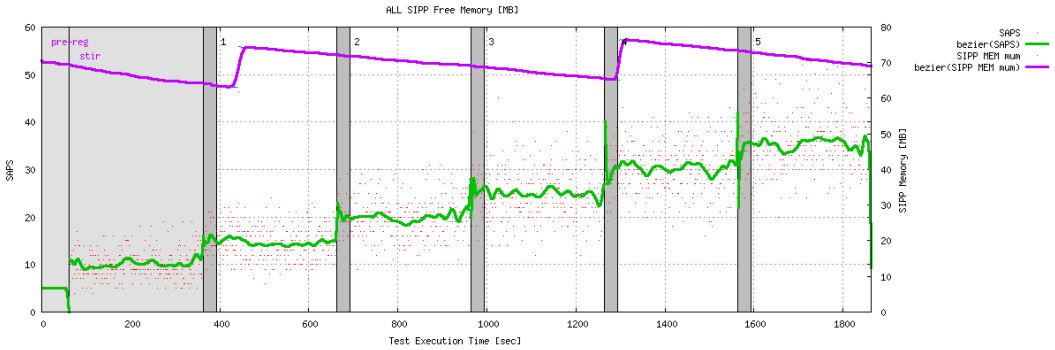
	Pre-reg	Step - 1	Step - 2	Step - 3	Step - 4	Step - 5
Requested Load	5	15	20	25	30	35
Effective Load	4.90	14.65	19.97	25.00	30.28	35.31
Ratio ims_reg %	100.00	29.25	30.06	30.04	29.31	29.87
Ratio ims_uac %	0.00	11.00	9.66	10.11	10.45	10.19
Ratio ims_dereg %	0.00	7.17	6.56	6.98	7.61	7.03
Ratio ims_msgc %	0.00	50.01	51.41	49.64	49.76	49.84
Ratio ims_rereg %	0.00	2.57	2.30	3.22	2.87	3.07
SIPP CPU mum	21.27	23.10	23.75	23.96	23.26	24.66
SIPP MEM mum	69.85	70.78	70.56	67.27	73.93	71.21
IHS ims_reg %	0.00	0.00	0.00	0.00	0.08	0.53
IHS ims_rereg %	0.00	0.00	0.00	0.00	0.00	0.34
global IHS %	0.00	0.00	0.00	0.00	0.02	0.17

6.2.5.1 Analysis of CPU Utilization and Available Memory

The CPU utilization of the test system is shown in Figure 6.26(a). The number of SAPS gradually increases during the benchmark. The test system's VM specification was described in Table 6.1. The test system's free memory during the benchmark's execution is shown in Figure 6.26(b).



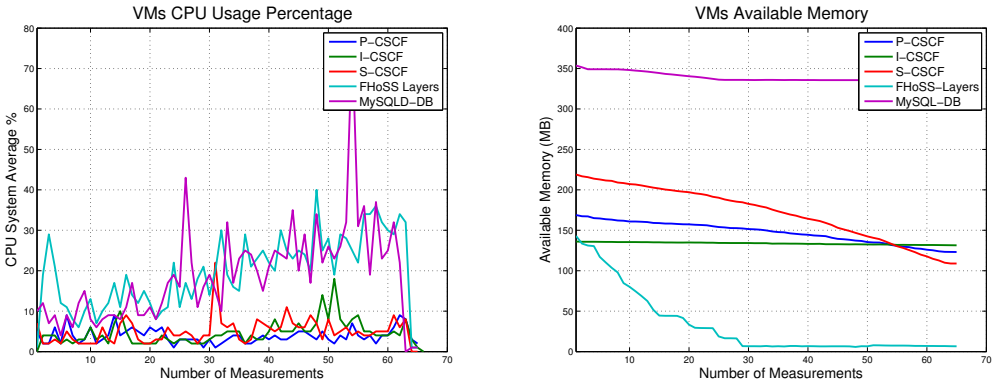
(a) CPU utilization



(b) Available memory

Figure 6.26. CPU utilization and available memory of the Test System's VM

Figure 6.27(a) depicts the CPU utilization of the SUT's VMs. It can be seen that the CPU usage of the FHoSS-Layers and MySQL-DB VM increase gradually and CPU usage is less than 40% during the benchmarking. MySQL-DB node has visible peaks of greater than 40% average CPU utilization. The CPU usage of P-CSCF, I-CSCF, and S-CSCF VMs CPU usage is less than 20% on average. The memory consumption of the SUT's VM during the benchmark's execution are shown in Figure 6.27(b). It can be observed that the FHoSS-Layer's VM consumes abruptly and almost becomes constant because of the virtual memory usage of JAVA application (FHoSS) running was reached to the limit.



(a) CPU utilization (b) Available memory

Figure 6.27. CPU utilization and available memory of SUT's VMs

6.2.5.2 Analysis of Calling Scenario

The delay in milliseconds between the caller sending a SIP INVITE request and the callee receiving the corresponding ACK message is depict in Figure 6.28. The x-axis represents the execution time (in seconds since the start) of the benchmarking test. The y-axis represents the number of SAPS on a linear scale (on the left vertical axis) and the delay on a logarithmic scale of 0.1 to 1000 milliseconds (on the right vertical axis). It can be observed that the mean delay is in the range of 5 to 10 milliseconds during the entire duration of the benchmarking execution.

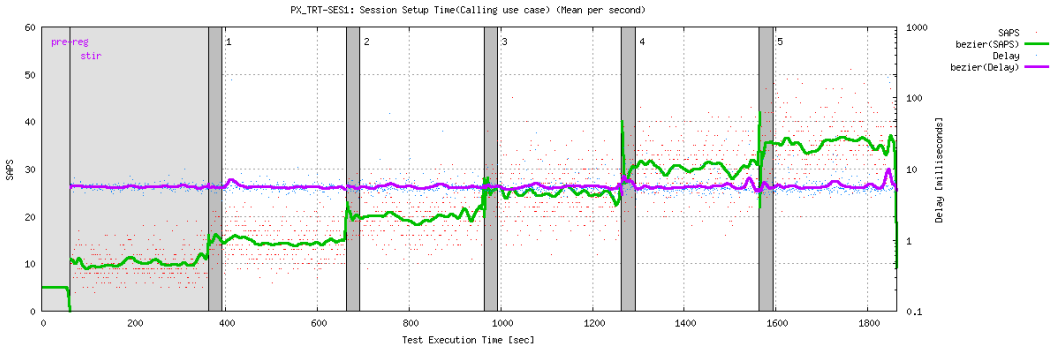


Figure 6.28. Session Setup Time

Figure 6.29 illustrates the delay in milliseconds between the caller sending a SIP INVITE request and the callee receiving the corresponding SIP INVITE

request. The x-axis represents the execution time (in seconds since the start) of the benchmarking test. The y-axis represents the number of SAPS on a linear scale (on the left vertical axis) and the delay on a logarithmic scale of 0.01 to 1000 milliseconds (on the right vertical axis). It can be observed that the mean delay is in the range of 1 to 5 milliseconds during the entire duration of the benchmarking execution.

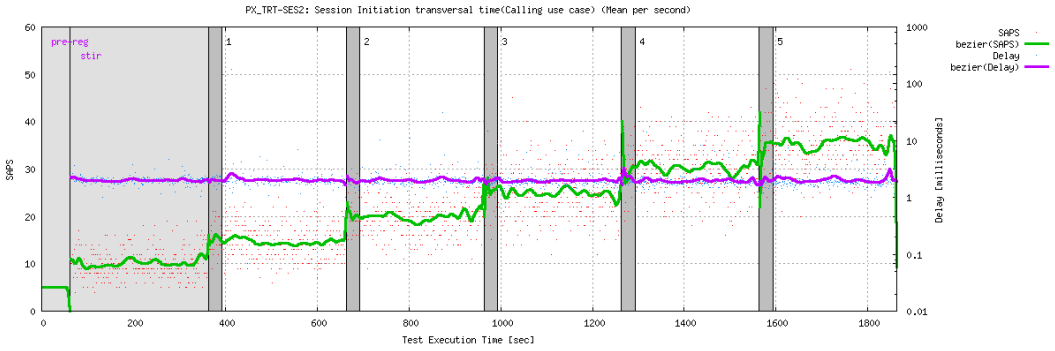


Figure 6.29. Session Initiation transversal time

Figure 6.30 depicts the delay between the first BYE and the corresponding 200 OK message. The x-axis represents the execution time (in seconds since the start) of benchmarking test. The y-axis represents the number of SAPS on a linear scale (on the left vertical axis) and the delay on a logarithmic scale of 0.1 to 100 milliseconds (on the right vertical axis). It can be observed that the mean delay is in the range of 1 to 5 milliseconds during the entire duration of the benchmarking execution.

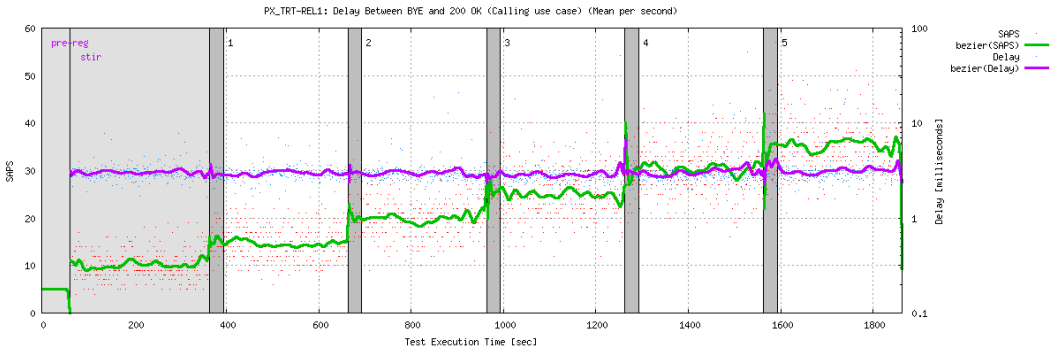


Figure 6.30. Delay Between BYE and 200 OK

6.2.5.3 Analysis of Message Scenario

The delay in milliseconds between the SIP Message and the corresponding 200 OK response is shown in Figure 6.31. The x-axis represents the execution time (in seconds since the start) of benchmarking test. The y-axis represents the number of SAPS on a linear scale (on the left vertical axis) and the delay on a logarithmic scale of 0.1 to 1000 milliseconds (on the right vertical axis). It can be observed that the mean delay is in the range of 1 to 5 milliseconds during the entire duration of the benchmarking execution, thus we can conclude that the delay does not increase with an increase in the number of SAPS during each phase of this benchmarking.

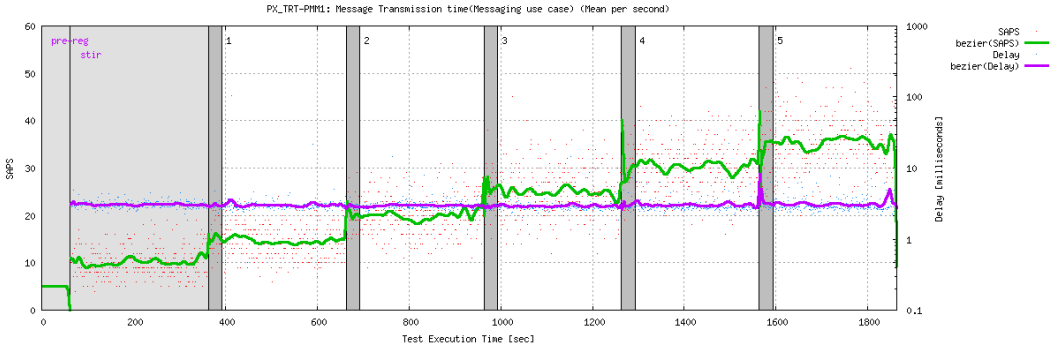


Figure 6.31. Message Transmission time

6.2.5.4 Analysis of Registration Scenario

The delay in milliseconds between the first SIP REGISTER request and the 401 Unauthorized message for all the benchmark phases in the registration use case during benchmark execution is shown in Figure 6.32. The x-axis represents the execution time (in seconds since the start) of benchmarking test. The y-axis represents the number of SAPS on a linear scale (on the left vertical axis) and the delay on a logarithmic scale of 0.1 to 10000 milliseconds (on the right vertical axis). It can be observed that the mean delay is in the range of 30 to 80 milliseconds during the entire duration of the benchmarking execution.

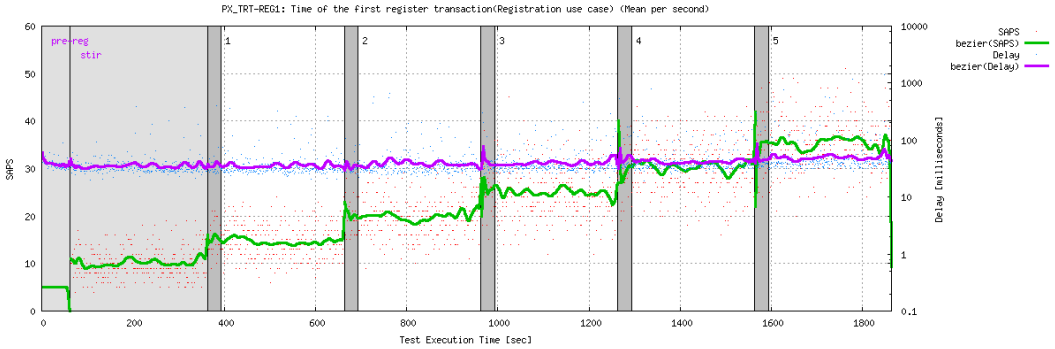


Figure 6.32. Time of the first register transaction

The delay in milliseconds between the second REGISTER and the 200 OK message in the registration scenario is shown in Figure 6.33. The x-axis represents the execution time (in seconds since the start) of benchmarking test. The y-axis represents the number of SAPS on a linear scale (on the left vertical axis) and the delay on a logarithmic scale of 0.1 to 1000 milliseconds (on the right vertical axis). It can be observed that the mean delay is in the range of 40 to 60 milliseconds during the entire duration of the benchmarking execution.

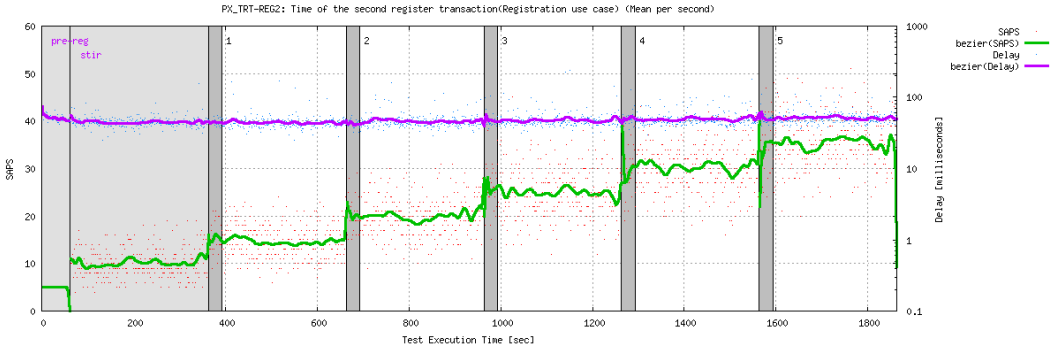


Figure 6.33. Time of the second register transaction

6.2.6 Analysis of third testbed scenario

The testbed setup for this third testbed scenario for benchmark testing is shown in Figure 6.34.

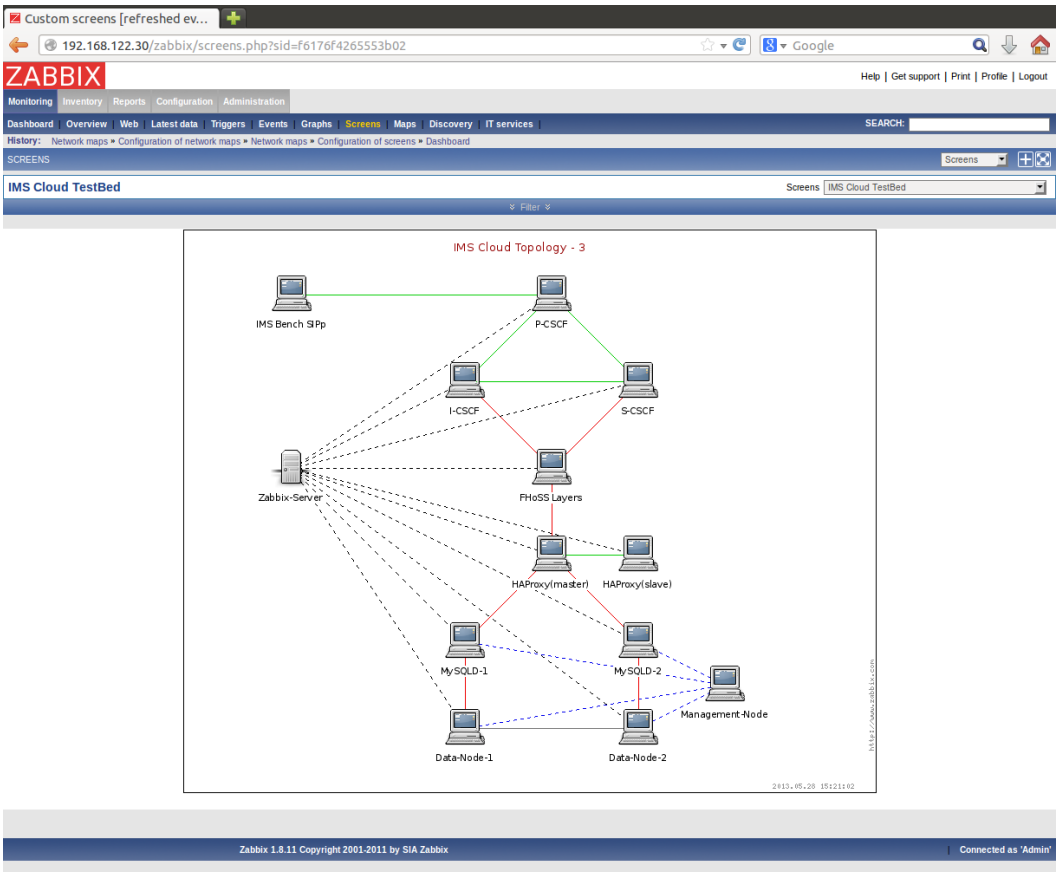


Figure 6.34. Third Testbed Topology

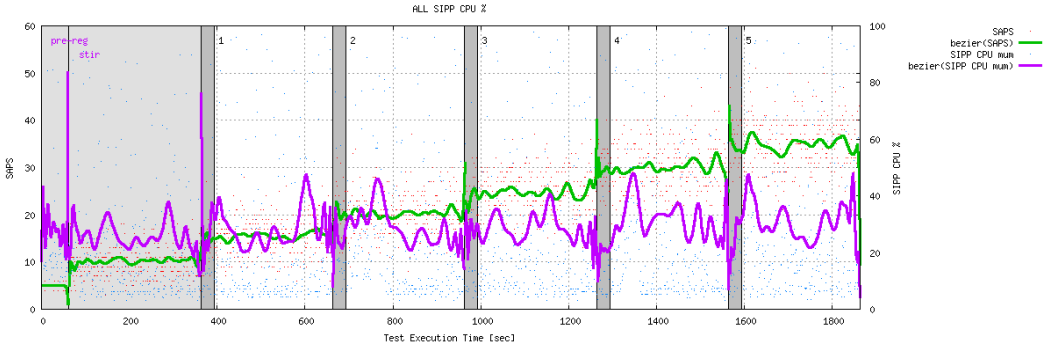
A summery of the benchmarking test for the third testbed topology is shown in Table 6.6. The terms in this table were explained in Section 6.2.4.

Table 6.6. Summary of Benchmarking Test

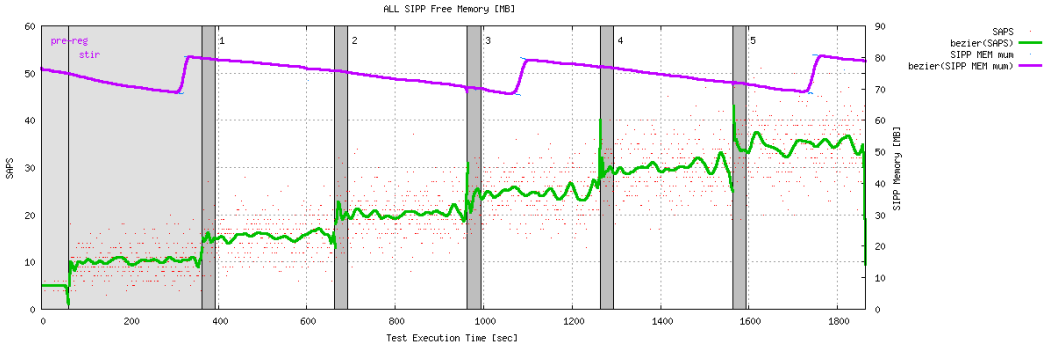
	Pre-reg	Step - 1	Step - 2	Step - 3	Step - 4	Step - 5
Requested Load	5	15	20	25	30	35
Effective Load	4.90	15.42	20.19	24.60	29.99	34.76
Ratio ims_reg %	100.00	29.89	30.59	29.57	29.62	29.70
Ratio ims_uac %	0.00	10.17	9.56	9.51	9.96	9.98
Ratio ims_dereg %	0.00	7.08	6.73	7.33	7.41	6.70
Ratio ims_msgc %	0.00	49.95	49.85	50.20	49.79	50.51
Ratio ims_rereg %	0.00	2.90	3.27	3.38	3.22	3.12
SIPP CPU mum	28.32	27.98	28.53	28.66	30.54	30.40
SIPP MEM mum	75.58	77.82	73.11	74.68	74.45	74.07
IHS ims_reg %	0.00	0.00	0.06	0.00	0.00	0.00
IHS ims_dereg %	0.00	0.34	0.54	0.00	0.00	0.48
IHS ims_msgc %	0.00	0.00	0.00	0.06	0.00	0.00
IHS ims_rereg %	0.00	0.00	0.00	0.44	0.00	0.00
global IHS %	0.00	0.02	0.06	0.05	0.00	0.03

6.2.6.1 Analysis of CPU Utilization and Available Memory

The CPU utilization of the test system is shown in Figure 6.35(a). The number of SAPS gradually increases during the benchmark. The test system's free memory decreases during the benchmark's execution is shown in Figures 6.35(b).



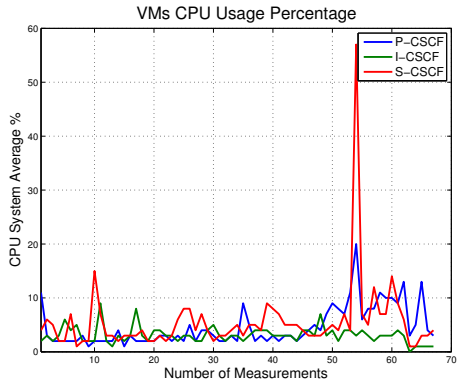
(a) CPU Utilization



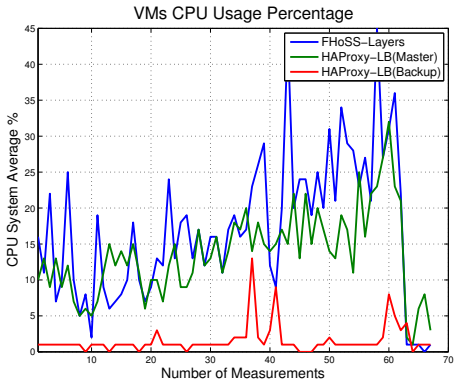
(b) Available Memory

Figure 6.35. CPU utilization and available memory of Test System's VM

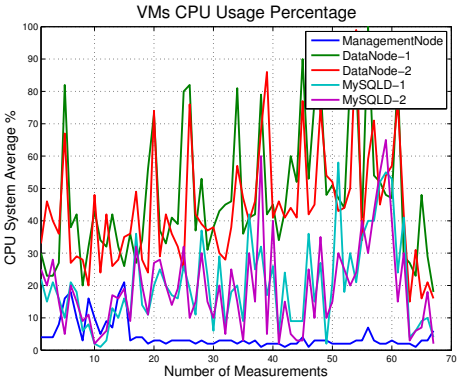
Figure 6.36(a) illustrates the CPU usage of P-CSCF, I-CSCF, and S-CSCF VMs. The CPU usage of the P-CSCF, I-CSCF, and S-CSCF is less than 20% on average except for S-CSCF which has one peak of greater than 50%. The CPU usage of the FHoSS Layers and HAProxy load balancer nodes VM is shown in Figure 6.36(b) and the average CPU usage is less than 40%. The HAProxy load balancer backup node has three peaks which indicates that some of the incoming requests are handled by the backup node. Figure 6.36(c) depicts the CPU utilization of the MySQL cluster nodes. Cluster front-end MySQL server nodes has an average CPU utilizaition of less than 60% and the average CPU usage of the management node is less than 20%. The data storage node's CPU utilization gradually increases, but remains less than 90% of CPU usage on average. The data nodes have high CPU utilizaition because of high number of input/output (I/O) waiting requests, which is shown in Figure 6.36(d). Additionaly, each data node have separate disk storage created by the hypervisor on a physical machine hard disk.



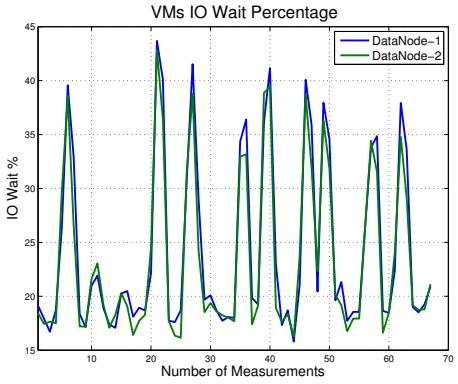
(a) CPU utilization of CSCFs



(b) CPU utilization of FHoSS Layers and LBs



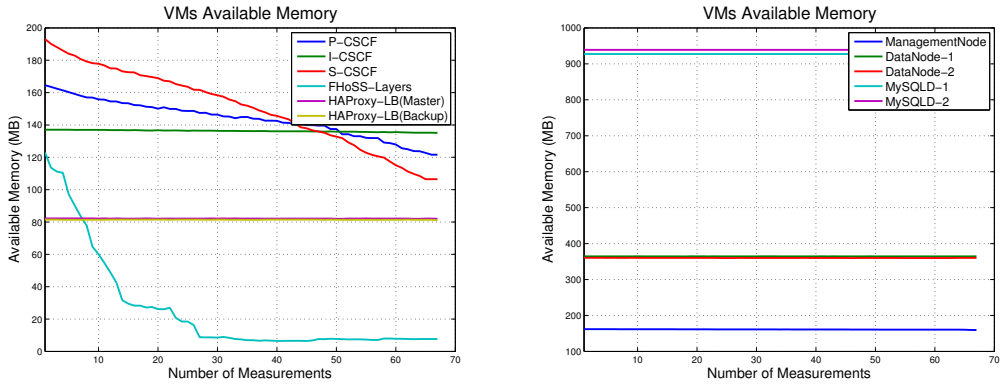
(c) CPU utilization of MySQL cluster nodes



(d) CPU I/O wait of MySQL cluster nodes

Figure 6.36. CPU utilization of SUT's VMs

Figure 6.37(a) illustrates the available memory of the CSCFs, FHoSS Layers, and HAProxy load balancer (LB) VMs. It can be noted that the load balancer VMs does not consume memory during the benchmarking execution. The available memory of the MySQL cluster nodes is shown in Figurer 6.37(b), which indicates that cluster nodes does not consumes memory during the benchmarking execution. However, MySQL cluster nodes (management, data storage, and MySQL server) VMs have the 512 MB, 2048 MB, and 1024 MB installed memory (respectively).



(a) Available memory of CSCFs, FHoSS Layers, and LBs

(b) Available memory of MySQL cluster nodes

Figure 6.37. Available memory of SUT VMs

6.2.6.2 Analysis of Calling Scenario

The delay in milliseconds between the caller sending a SIP INVITE request and callee receiving the corresponding ACK message is shown in Figure 6.38. The x-axis represents the execution time (in seconds since the start) of benchmarking test. The y-axis represents the number of SAPS on a linear scale (on the left vertical axis) and the delay on a logarithmic scale of 0.1 to 1000 milliseconds (on the right vertical axis). It can be observed that the mean delay is in the range of 5 to 60 milliseconds during the entire duration of the benchmarking execution.

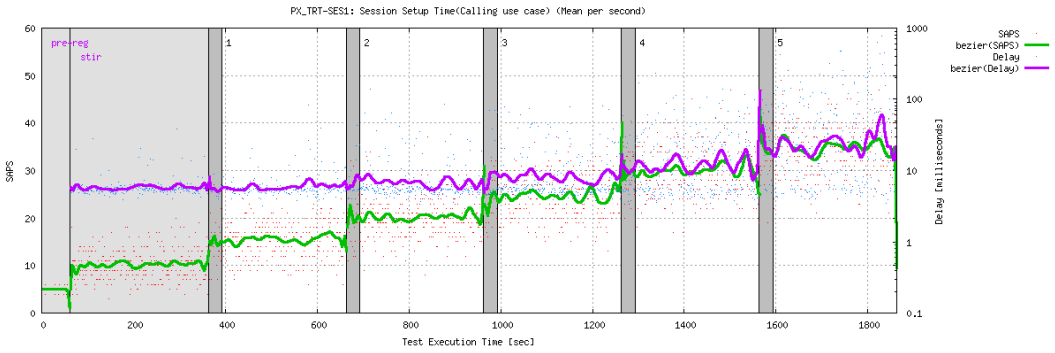


Figure 6.38. Session Setup Time

The delay in milliseconds between the caller sending a SIP INVITE request and the callee receiving the corresponding SIP INVITE request is shown in Figure 6.39. The x-axis represents the execution time (in seconds since the start) of benchmarking test. The y-axis represents the number of SAPS on a linear scale (on the left vertical axis) and the delay on a logarithmic scale of 0.01 to 1000 milliseconds (on the right vertical axis). It can be observed that the mean delay is in the range of 1 to 30 milliseconds during the entire duration of the benchmarking execution.

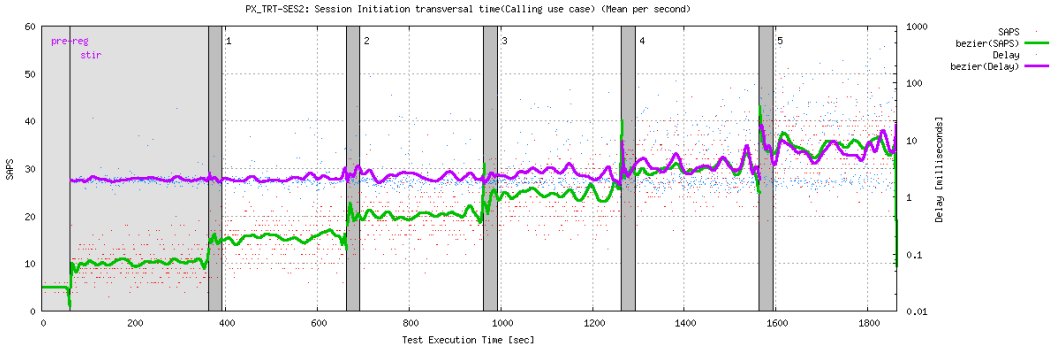


Figure 6.39. Session Initiation transversal time

Figure 6.40 illustrated the delay between the first BYE and the corresponding 200 OK message. The x-axis represents the execution time (in seconds since the start) of benchmarking test. The y-axis represents the number of SAPS on a linear scale (on the left vertical axis) and the delay on a logarithmic scale of 0.1 to 1000 milliseconds (on the right vertical axis). It can be observed that the mean delay is in the range of 1 to 30 milliseconds during the entire duration of the benchmarking execution. It can be seen that in step 5 bezier curve start declining, but the mean

and the standard deviation are higher in step 5 due to the spread out of data over large range of values. The statistical analysis is described in Section 6.2.8.3.

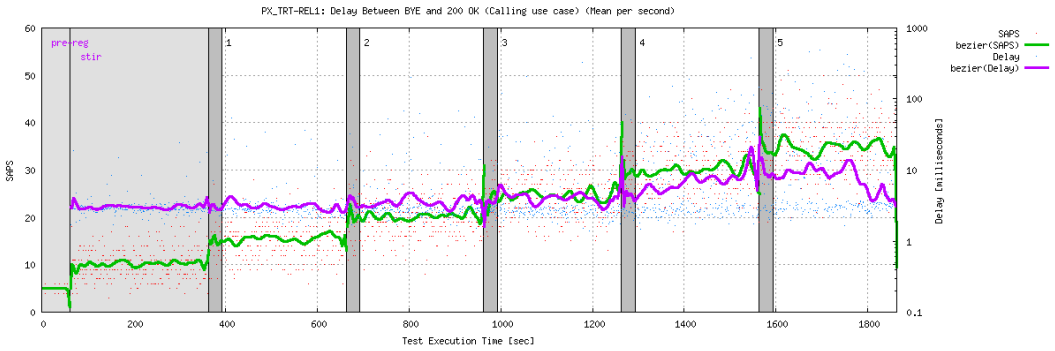


Figure 6.40. Delay Between BYE and 200 OK

6.2.6.3 Analysis of Message Scenario

The delay in milliseconds between the SIP Message and the 200 OK message is shown in Figure 6.41. The x-axis represents the execution time (in seconds since the start) of benchmarking test. The y-axis represents the number of SAPS on a linear scale (on the left vertical axis) and the delay on a logarithmic scale of 0.1 to 1000 milliseconds (on the right vertical axis). It can be observed that the mean delay is in the range of 1 to 40 milliseconds during the entire duration of the benchmarking execution.

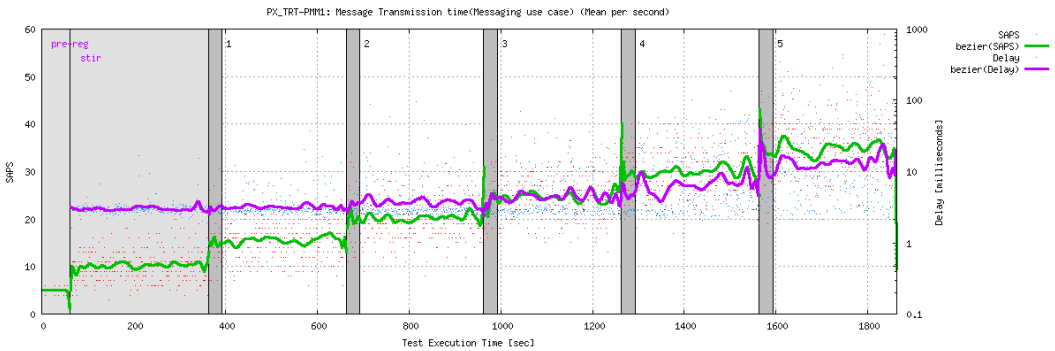


Figure 6.41. Message Transmission time

6.2.6.4 Analysis of Registration Scenario

The delay in milliseconds between the first SIP REGISTER request and the 401 Unauthorized message for all the benchmark phases in the registration use case during benchmark execution is shown in Figure 6.42. The x-axis represents the execution time (in seconds since the start) of benchmarking test. The y-axis represents the number of SAPS on a linear scale (on the left vertical axis) and the delay on a logarithmic scale of 0.1 to 1000 milliseconds (on the right vertical axis). It can be observed that the mean delay is in the range of 80 to 160 milliseconds during the entire duration of the benchmarking execution.

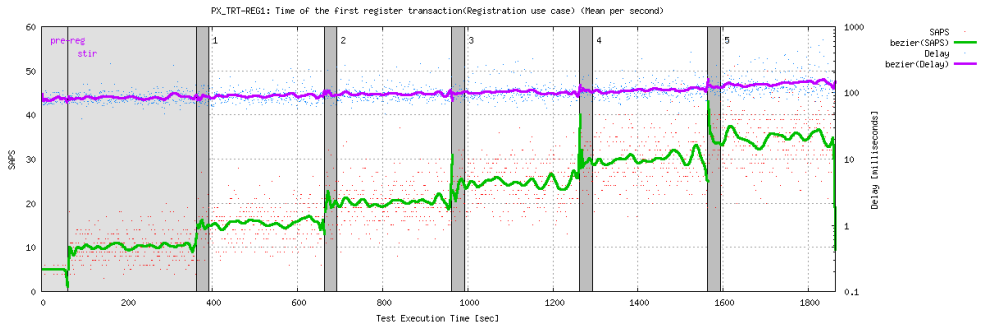


Figure 6.42. Time of the first register transaction

The delay in milliseconds between the second REGISTER and the corresponding 200 OK message in the registration scenario is shown in Figure 6.43. The x-axis represents the execution time (in seconds since the start) of benchmarking test. The y-axis represents the number of SAPS on a linear scale (on the left vertical axis) and the delay on a logarithmic scale of 0.1 to 1000 milliseconds (on the right vertical axis). It can be observed that the mean delay is in the range of 130 to 190 milliseconds during the entire duration of the benchmarking execution.

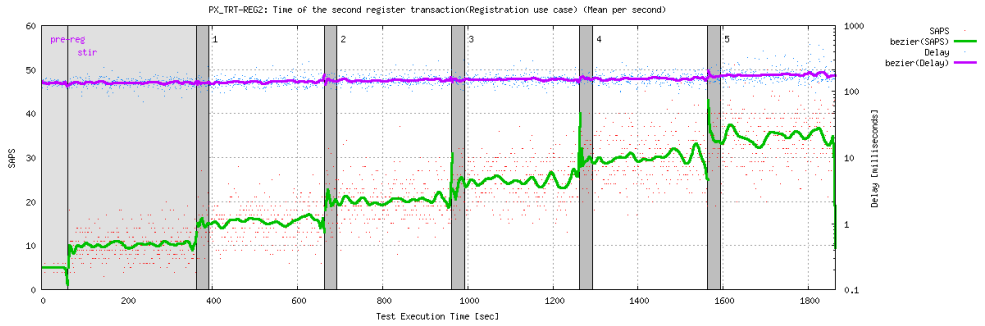


Figure 6.43. Time of the second register transaction

6.2.7 QoS Analysis of IMS Registration Request Delay (RRD)

The registration request delay (RRD) is the time interval from when the user agent originates the initial SIP REGISTER message to the intended registrar, until the corresponding 200 OK message is received, indicating a successful registration. RRD is expressed in units of milliseconds [63]. The contributions to RRD are illustrated in Figure 6.44. RRD is the sum of two time intervals t1 and t2, where t1 is the time interval from sending the first SIP REGISTER request until a 401 Unauthorized response is received and t2 is the time interval from sending the second SIP REGISTER request until the corresponding 200 OK message is received. In this thesis project, we employed the IMS Bench SIPp test system to measure the RRD.

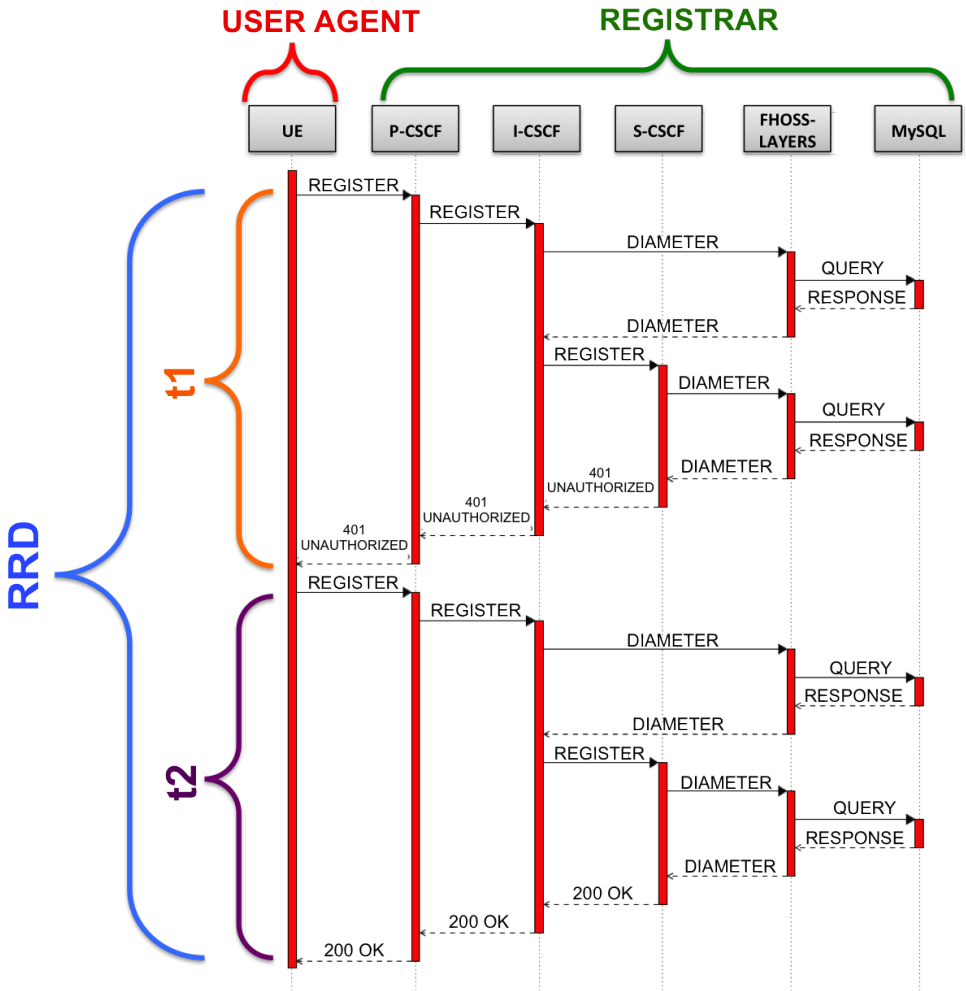


Figure 6.44. Registration request delay (Adapted from Section 4.1 Figure of [63])

The difference in RRD measurements of three testbed scenarios will be described with the help of Figures 6.45, 6.46 and 6.47. In the first testbed scenario, MySQL database resides at level zero as shown in Figure 6.45, while in second testbed scenario the MySQL database resides at level one as shown in Figure 6.46, which results in the RRD being higher than measured in first testbed scenario. Figure 6.44 showed an example of RRD for the second testbed case. The subscriber's data is at level three in the third testbed scenario as shown in Figure 6.47, which results in a RRD greater than the other two scenarios. The impact of higher RRD was visible in the analysis of the registration scenarios. Table 6.7 summarizing the mean RRD values that were measured for the three different testbeds. However, this increased delay of the registration request results from that system's improved ability to scale the FHoSS database instances horizontally.

Table 6.7. Summary of mean RRD measurements

Step	Requested Load	TestBed - 1 Mean RRD (msec)	TestBed - 2 Mean RRD (msec)	TestBed - 3 Mean RRD (msec)
Pre-reg	5	60.52	91.36	216.42
1	15	57.87	87.61	235.38
2	20	59.32	94.90	251.29
3	25	62.38	95.60	261.63
4	30	64.60	103.81	274.20
5	35	70.90	124.35	339.75

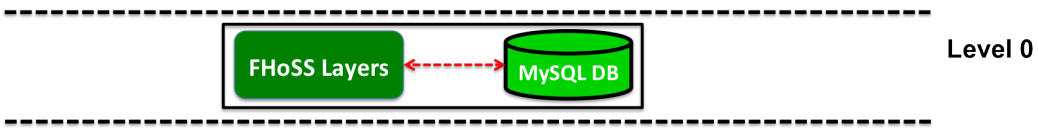


Figure 6.45. Subscriber's data level of first testbed

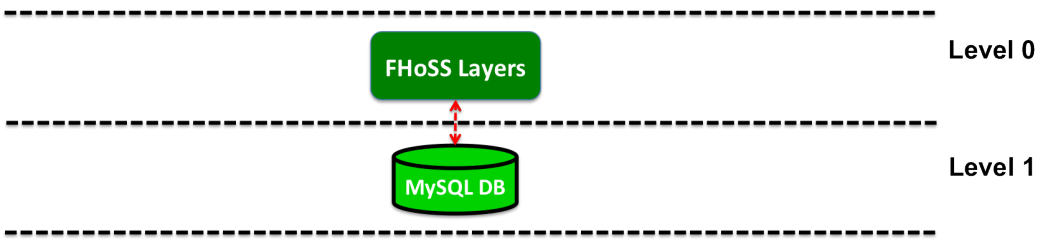


Figure 6.46. Subscriber's data level of second testbed

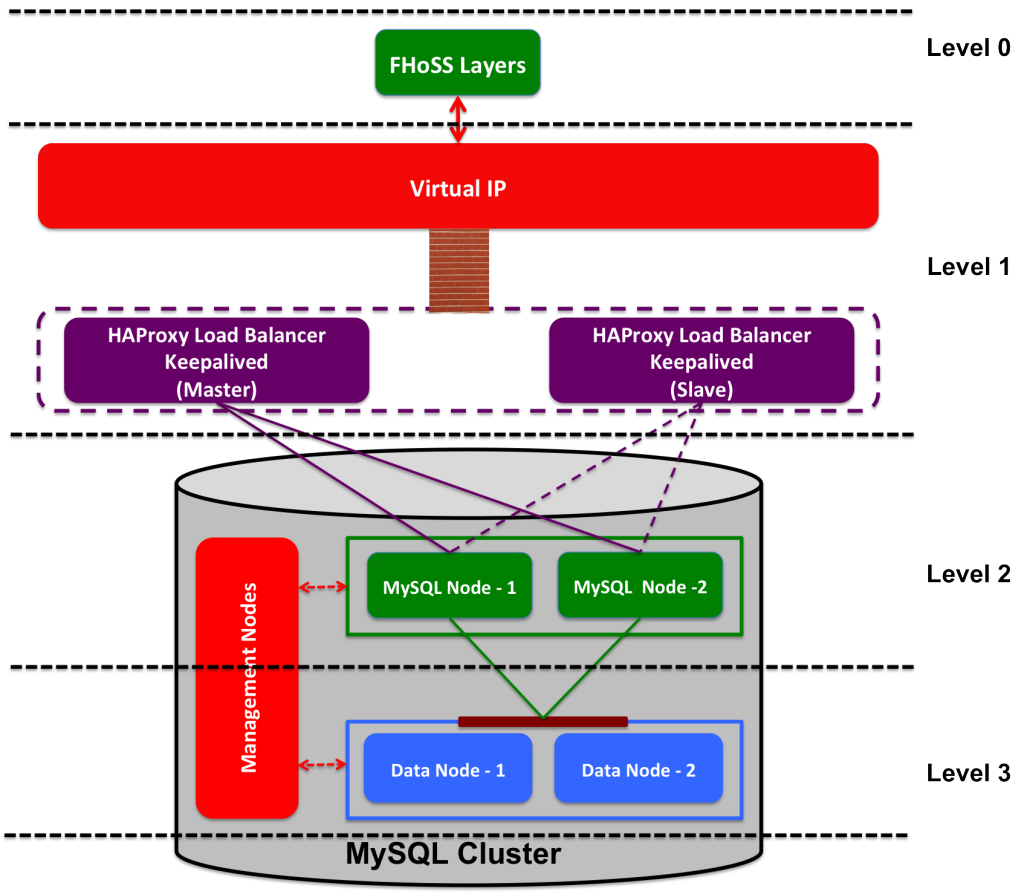


Figure 6.47. Subscriber's data level of third testbed

6.2.8 Statistical Analysis of Calling Scenario

This section described the statistical analysis of calling scenario of three different testbeds.

6.2.8.1 Statistical Analysis of Session Setup Time

The statistical analysis of the delay in milliseconds between the caller sending a SIP INVITE request and callee receiving the corresponding ACK message of three different testbeds are shown in Tables 6.8, 6.9, and 6.10 respectively.

Table 6.8. Session Setup Time of first testbed

Step	Requested Load	Mean	Standard Deviation	Minimun	Maximum
1	15	6.00	3.09	3.72	52.70
2	20	6.49	7.17	3.47	114.31
3	25	5.84	2.25	3.55	46.77
4	30	6.85	14.04	3.12	253.49
5	35	6.65	8.95	3.11	142.13

Table 6.9. Session Setup Time of second testbed

Step	Requested Load	Mean	Standard Deviation	Minimun	Maximum
1	15	5.93	8.80	3.20	181.92
2	20	6.16	8.57	3.19	185.02
3	25	5.87	4.10	3.40	85.95
4	30	6.20	5.59	3.22	94.54
5	35	7.01	16.85	2.92	336.33

Table 6.10. Session Setup Time of third testbed

Step	Requested Load	Mean	Standard Deviation	Minimun	Maximum
1	15	6.40	6.27	3.16	81.89
2	20	8.84	14.41	3.38	151.07
3	25	12.33	22.31	3.04	198.71
4	30	19.05	36.14	2.92	364.03
5	35	56.70	111.25	2.93	814.83

6.2.8.2 Statistical Analysis of Session Initiation Transversal Time

The statistical analysis of the delay in milliseconds between the caller sending a SIP INVITE request and callee receiving the corresponding SIP INVITE request of three different testbeds are shown in Tables 6.11, 6.12, and 6.13 respectively.

Table 6.11. Session Initiation Transversal Time of first testbed

Step	Requested Load	Mean	Standard Deviation	Minimun	Maximum
1	15	2.25	2.63	1.06	48.53
2	20	2.41	4.15	1.20	85.25
3	25	2.16	2.02	1.16	43.93
4	30	2.21	3.06	0.93	72.92
5	35	2.35	3.97	0.87	73.36

Table 6.12. Session Initiation Transversal Time of second testbed

Step	Requested Load	Mean	Standard Deviation	Minimun	Maximum
1	15	2.39	8.09	1.00	179.10
2	20	2.14	1.88	1.09	30.70
3	25	2.07	1.73	0.96	37.46
4	30	2.19	2.68	1.02	45.31
5	35	2.68	10.77	0.85	254.39

Table 6.13. Session Initiation Transversal Time of third testbed

Step	Requested Load	Mean	Standard Deviation	Minimun	Maximum
1	15	2.47	4.20	0.93	76.22
2	20	3.77	10.68	1.04	130.10
3	25	5.05	14.12	1.01	147.25
4	30	7.45	19.41	0.94	181.15
5	35	27.68	78.19	0.96	652.31

6.2.8.3 Statistical Analysis of Delay Between BYE and 200 OK

The statistical analysis of the delay in milliseconds between the first BYE and the corresponding 200 OK message of three different testbeds are shown in Tables 6.14, 6.15, and 6.16 respectively.

Table 6.14. Delay Between BYE and 200 OK of first testbed

Step	Requested Load	Mean	Standard Deviation	Minimun	Maximum
1	15	3.79	11.00	1.53	232.70
2	20	3.09	1.16	1.36	16.34
3	25	3.26	3.88	1.35	93.98
4	30	3.48	4.87	1.32	91.63
5	35	3.89	11.24	1.19	292.33

Table 6.15. Delay Between BYE and 200 OK of second testbed

Step	Requested Load	Mean	Standard Deviation	Minimun	Maximum
1	15	3.03	1.13	1.41	17.00
2	20	3.15	2.27	1.38	35.53
3	25	3.19	7.18	1.38	196.44
4	30	3.41	6.25	1.39	162.74
5	35	3.56	7.06	1.39	207.49

Table 6.16. Delay Between BYE and 200 OK of third testbed

Step	Requested Load	Mean	Standard Deviation	Minimun	Maximum
1	15	3.82	6.85	1.38	76.73
2	20	5.98	19.58	1.33	246.58
3	25	8.46	42.63	1.24	972.34
4	30	16.97	53.33	1.28	758.63
5	35	24.45	93.23	1.52	973.82

6.2.9 Statistical Analysis of Message Scenario

The statistical analysis of the delay in milliseconds between the SIP Message and the 200 OK message of three different testbeds are shown in Tables 6.17, 6.18, and 6.19 respectively.

Table 6.17. Message Transmission time of first testbed

Step	Requested Load	Mean	Standard Deviation	Minimun	Maximum
1	15	3.68	16.06	1.29	607.26
2	20	3.23	3.78	1.14	186.94
3	25	3.11	1.42	1.40	39.66
4	30	3.50	9.13	1.21	248.82
5	35	3.63	8.97	1.36	282.32

Table 6.18. Message Transmission time of second testbed

Step	Requested Load	Mean	Standard Deviation	Minimun	Maximum
1	15	3.42	11.55	1.34	361.96
2	20	3.00	2.05	1.15	82.54
3	25	2.98	2.22	1.24	57.44
4	30	3.18	3.97	1.14	95.39
5	35	3.51	9.36	1.22	361.15

Table 6.19. Message Transmission time of third testbed

Step	Requested Load	Mean	Standard Deviation	Minimun	Maximum
1	15	3.59	6.02	1.31	116.93
2	20	4.86	15.00	1.22	453.31
3	25	6.32	17.27	1.35	245.67
4	30	10.22	28.08	1.24	370.54
5	35	36.58	110.57	1.36	1041.87

6.2.10 Statistical Analysis of Registration Scenario

This section described the statistical analysis of registration scenario of three different testbeds.

6.2.10.1 Statistical Analysis of Time of the first register transaction

The statistical analysis of the delay in milliseconds between the first SIP REGISTER request and the 401 Unauthorized message of three different testbeds are shown in Tables 6.20, 6.21, and 6.22 respectively.

Table 6.20. Time of the first register transaction of first testbed

Step	Requested Load	Mean	Standard Deviation	Minimun	Maximum
Pre-reg	5	29.20	23.64	12.49	336.83
1	15	34.07	40.87	7.82	680.29
2	20	31.10	42.31	7.96	834.17
3	25	32.78	39.26	7.58	674.00
4	30	38.23	80.82	7.57	1212.49
5	35	43.05	70.62	7.50	609.80

Table 6.21. Time of the first register transaction of second testbed

Step	Requested Load	Mean	Standard Deviation	Minimun	Maximum
Pre-reg	5	39.52	19.72	19.81	171.04
1	15	41.65	58.99	11.54	837.69
2	20	47.58	95.09	13.32	1225.46
3	25	46.54	64.81	12.48	874.32
4	30	52.50	192.12	12.92	7535.91
5	35	71.39	329.99	11.41	7841.55

Table 6.22. Time of the first register transaction of third testbed

Step	Requested Load	Mean	Standard Deviation	Minimun	Maximum
Pre-reg	5	81.68	26.57	35.04	237.88
1	15	93.09	41.50	32.26	428.31
2	20	100.57	52.10	34.30	617.31
3	25	106.78	60.56	35.47	556.75
4	30	114.33	61.98	31.11	841.24
5	35	151.18	109.30	33.30	978.65

6.2.10.2 Statistical Analysis of Time of the second register transaction

The statistical analysis of the delay in milliseconds between the second REGISTER and the corresponding 200 OK message of three different testbeds are shown in Tables 6.23, 6.24, and 6.25 respectively.

Table 6.23. Time of the second register transaction of first testbed

Step	Requested Load	Mean	Standard Deviation	Minimun	Maximum
Pre-reg	5	31.32	14.77	17.13	165.99
1	15	23.80	21.02	11.05	385.22
2	20	28.22	29.02	10.97	651.40
3	25	29.60	23.82	11.04	759.73
4	30	26.37	32.00	11.17	655.29
5	35	27.85	32.96	10.87	607.15

Table 6.24. Time of the second register transaction of second testbed

Step	Requested Load	Mean	Standard Deviation	Minimun	Maximum
Pre-reg	5	51.84	9.08	28.17	104.10
1	15	45.96	33.63	18.29	892.87
2	20	47.32	24.04	18.51	480.69
3	25	49.06	32.59	17.82	842.86
4	30	51.31	33.73	17.94	583.50
5	35	52.96	31.75	17.94	634.33

Table 6.25. Time of the second register transaction of third testbed

Step	Requested Load	Mean	Standard Deviation	Minimun	Maximum
Pre-reg	5	134.74	23.71	57.37	243.48
1	15	142.29	41.99	48.34	567.60
2	20	150.72	45.27	60.74	553.33
3	25	154.85	51.86	57.66	606.63
4	30	159.87	52.14	50.97	781.67
5	35	188.57	91.10	63.42	1025.61

6.2.11 Statistical Analysis of Registration Scenario Retransmissions

Retransmissions occur when the corresponding response of an SIP request is not received within a defined period. The SIP standard timeout defines that the first retransmission takes place when a SIP request has been sent and the response is not received in 500 milliseconds [64]. The number of retransmissions per second for the registration scenario for all the three testbeds is illustrated in Tables 6.26, 6.27, and 6.28 respectively.

Table 6.26. Registration scenario retransmissions of first testbed

Step	Requested Load	Mean	Standard Deviation	Minimun	Maximum
Pre-reg	5	0.00	0.00	0.00	0.00
1	15	0.07	0.51	0.00	6.00
2	20	0.07	0.56	0.00	6.00
3	25	0.01	0.18	0.00	3.00
4	30	0.18	0.89	0.00	11.00
5	35	0.17	0.55	0.00	4.00

Table 6.27. Registration scenario retransmissions of second testbed

Step	Requested Load	Mean	Standard Deviation	Minimun	Maximum
Pre-reg	5	0.00	0.00	0.00	0.00
1	15	0.02	0.15	0.00	2.00
2	20	0.06	0.52	0.00	6.00
3	25	0.04	0.31	0.00	3.00
4	30	0.10	0.47	0.00	5.00
5	35	0.41	1.04	0.00	7.00

Table 6.28. Registration scenario retransmissions of third testbed

Step	Requested Load	Mean	Standard Deviation	Minimun	Maximum
Pre-reg	5	0.00	0.00	0.00	0.00
1	15	0.01	0.13	0.00	2.00
2	20	0.01	0.11	0.00	1.00
3	25	0.06	0.46	0.00	5.00
4	30	0.05	0.37	0.00	5.00
5	35	0.48	1.75	0.00	15.00

6.2.12 Statistical Analysis of Calling Scenario Retransmissions

The number of retransmissions per second for the calling scenario for all the three testbeds is illustrated in Tables 6.29, 6.30, and 6.31 respectively.

Table 6.29. Calling scenario retransmissions of first testbed

Step	Requested Load	Mean	Standard Deviation	Minimun	Maximum
1	15	0.00	0.00	0.00	0.00
2	20	0.00	0.00	0.00	0.00
3	25	0.00	0.00	0.00	0.00
4	30	0.00	0.00	0.00	0.00
5	35	0.00	0.00	0.00	0.00

Table 6.30. Calling scenario retransmissions of second testbed

Step	Requested Load	Mean	Standard Deviation	Minimun	Maximum
1	15	0.00	0.00	0.00	0.00
2	20	0.00	0.00	0.00	0.00
3	25	0.00	0.00	0.00	0.00
4	30	0.00	0.00	0.00	0.00
5	35	0.00	0.06	0.00	1.00

Table 6.31. Calling scenario retransmissions of third testbed

Step	Requested Load	Mean	Standard Deviation	Minimun	Maximum
1	15	0.00	0.00	0.00	0.00
2	20	0.00	0.00	0.00	0.00
3	25	0.00	0.00	0.00	0.00
4	30	0.00	0.06	0.00	1.00
5	35	0.13	0.82	0.00	9.00

6.2.13 Statistical Analysis of IHS percentage

The percentage of IHS for all three testbed cases are shown in tables 6.32, 6.33, and 6.34 respectively. It can be observed that the mean percentage of IHS does not reached the defined threshold of 1% for each steps of all three testbed cases.

Table 6.32. IHS per use_case percentage of first testbed

Step	Requested Load	Mean	Standard Deviation	Minimun	Maximum
Pre-reg	5	0.00	0.00	0.00	0.00
1	15	0.00	0.00	0.00	0.00
2	20	0.00	0.00	0.00	0.00
3	25	0.00	0.00	0.00	0.00
4	30	0.05	0.39	0.00	3.70
5	35	0.07	0.47	0.00	5.41

Table 6.33. IHS per use_case percentage of second testbed

Step	Requested Load	Mean	Standard Deviation	Minimun	Maximum
Pre-reg	5	0.00	0.00	0.00	0.00
1	15	0.00	0.00	0.00	0.00
2	20	0.00	0.00	0.00	0.00
3	25	0.00	0.00	0.00	0.00
4	30	0.03	0.33	0.00	3.45
5	35	0.15	1.02	0.00	12.50

Table 6.34. IHS per use_case percentage of third testbed

Step	Requested Load	Mean	Standard Deviation	Minimun	Maximum
Pre-reg	5	0.00	0.00	0.00	0.00
1	15	0.06	0.64	0.00	8.33
2	20	0.04	0.44	0.00	5.56
3	25	0.05	0.45	0.00	4.55
4	30	0.00	0.00	0.00	0.00
5	35	0.03	0.26	0.00	2.94

Chapter 7

Conclusions and Future Work

This discusses all the results obtained in this master's thesis project. Additionally, some suggested future work to build upon this master thesis project is suggested in order to continue to enhance the system. Moreover, some required reflections are presented in Section 7.3.

7.1 Conclusions

The test infrastructure has been designed to analyzed the impact of virtualization of an IMS core network. The FHoSS database infrastructure has been enhanced in order to offer high availability and high scalability. Additionally, a solution has been proposed to synchronize the MySQL cluster by integrating a HAProxy load balancer in a passive architecture in order to meet the requirements of high availability and high scalability.

Functional testing has been performed after successfully deploying and configuring a OpenIMS core with an integrated MySQL cluster acting as a FHoSS database. All of the virtual machines have been placed on top of a hypervisor on a single physical machine. A SIP based soft phone "myMonster" was used for functional testing. Performance testing of virtualized IMS testbed has been conducted using an open source implementation of ETSI's IMS/NGN Performance Benchmark specification (i.e. the IMS Bench SIPp test system). The results of the performance testing would be better, if the testing shall be performed by deploying all the testbed components on separate computing resources connected with high speed network.

When deploying and configuring the testbed, most of the problems were related to the configuration of the OpenIMS core components, IMS Bench SIPp, and Zabbix configuration. The FHoSS was written entirely in JAVA with a focus on conformance rather than performance as stated in [41]. An open questions is what would the performance be for an implementation written with a focus on

performance.

7.2 Future Work

This master's thesis project analyzed the virtualization of OpenIMS core, integrated the FHoSS MySQL database with MySQL cluster technology (in order to synchronize a horizontally scalable database), and the system performance was evaluated. Future work should examine a IMSaaS solution. Figure 7.1 shows the high level IMS network cloud architecture. The following specific areas for enhancement are suggested:

- Investigate and implement a load balancing mechanism for the IMS network. In [65], they proposed a SIP based load balancing solution between the P-CSCF and UE. In this approach the load balancer acts as a SIP proxy for the IMS network. The load balancing provides the scalability of P-CSCF IMS core component.

- Investigate and implement a load balancing solution for I-CSCF and S-CSCF core components of IMS. It might be useful to use a software defined network (SDN) approach to load balancing for I-CSCF and S-CSCF. In [66] and [67], they proposed OpenFlow based load balancing for different services. OpenFlow is an open source implementation of SDN. A similar approach could be used to implement a load balancing mechanism to improve the scalability of I-CSCF and S-CSCF components.

- Investigate and implement a Cx diameter interface load balancing mechanism in order to improve the HSS front-end diameter layer scalability. There are many proposed solutions [68, 69, 70] related to the HSS front-end diameter interface scalability.

- Perform testing of a complete IMS network scalability realized over a large scale cloud infrastructure, such as BoneFIRE (`http://www.bonfire-project.eu`).

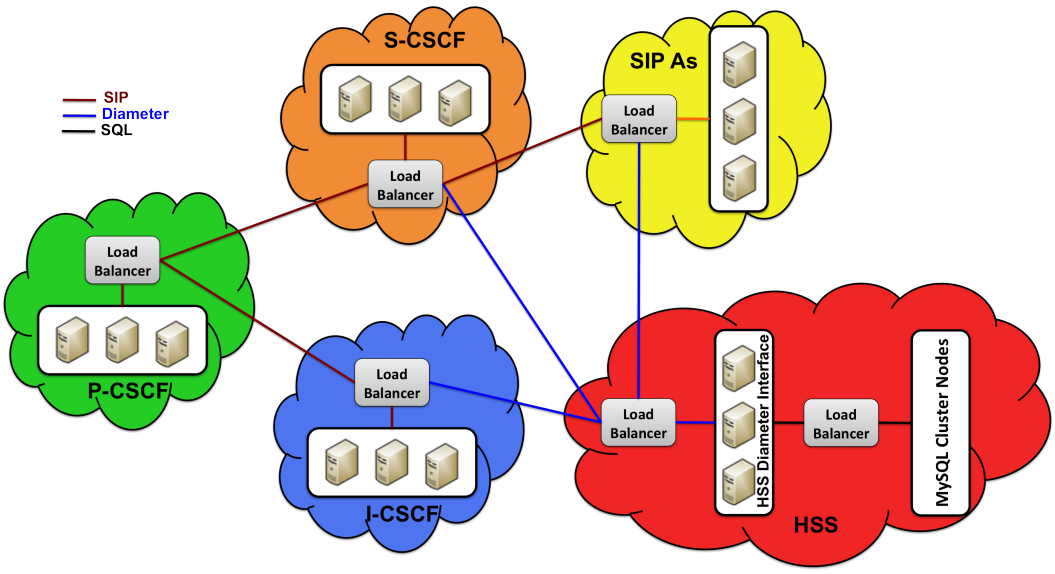


Figure 7.1. High level Cloud realization of an IMS network architecture

7.3 Required Reflections

This master's thesis project enhanced my knowledge and skills regarding scientific report writing with appropriate references and citations, and I learned about a number of open source technologies which were used to setup the testbed environment. This thesis project presents an solution for database scalability using MySQL cluster technology, hence the FHoSS has the potential to be deployed over a large scale IMS cloud network. The proposed solution provides a greater number of potential benefits in terms of economic aspect. The project testbed was deployed using an open source technologies that can help reduce the deployment cost of future IMS realizations. This could have large economic impact by enabling the deployment of an open source technologies based solution. Considering the limitations of the thesis work, the proposed future work motivates the further enhancement of the system.

Bibliography

[1] I. Albarrán Munoz and M. Parras Ruiz De Azua, "Telecommunication Services′ Migration to the Cloud : Network Performance analysis," Master's thesis, KTH, Communication Systems, CoS, 2012. [Online]. Available: http://urn.kb.se/resolve?urn=urn:nbn:se:kth:diva-93841, [Accessed: April 30, 2013].

[2] X. Lei, X. Zhe, M. Shaowu, and T. Xiongyan, "Cloud computing and services platform construction of telecom operator," in *Broadband Network Multimedia Technology, 2009. IC-BNMT '09. 2nd IEEE International Conference on*, pp. 864–867, 2009. DOI: 10.1109/ICBNMT.2009.5347793.

[3] A. Roozbeh, "Resource monitoring in a Network Embedded Cloud : An extension to OSPF-TE," Master's thesis, KTH, Radio Systems Laboratory (RS Lab), 2013. [Online]. Available: http://urn.kb.se/resolve?urn=urn:nbn:se:kth:diva-124367 , [Accessed: June 26th, 2013].

[4] 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*, pp. 1–6, 2009. DO1: 10.1109/ICIN.2009.5357099.

[5] J.-L. Chen, S.-L. Wuy, Y. Larosa, P.-J. Yang, and Y.-F. Li, "IMS cloud computing architecture for high-quality multimedia applications," in *Wireless Communications and Mobile Computing Conference (IWCMC), 2011 7th International*, pp. 1463–1468, 2011. DOI: 10.1109/IWCMC.2011.5982754.

[6] Christian Makaya, Ashutosh Dutta, Subir Das, Dana Chee, F. Joe Lin, Satoshi Komorita, and Hidetoshi Yokota, "Service continuity support in self-organizing IMS networks," pp. 1–5, IEEE, Feb. 2011. DOI: 10.1109/WIRELESSVITAE.2011.5940890.

[7] P. Bellavista, G. Carella, L. Foschini, T. Magedanz, F. Schreiner, and K. Campowsky, "QoS-aware elastic cloud brokering for IMS infrastructures," in *Computers and Communications (ISCC), 2012 IEEE Symposium on*, pp. 157–160, July. DOI: 10.1109/ISCC.2012.6249285.

[8] X. Zhiqun, C. Duan, H. Zhiyuan, and S. Qunying, "Emerging of Telco Cloud," *Communications, China*, vol. 10, no. 6, pp. 79–85, 2013. DOI: 10.1109/CC.2013.6549261.

[9] T. V. Ganesh, "Architecting a cloud based IP Multimedia System (IMS)," 21 Feb. 2013. [Online]. Available: http://gigadom.wordpress.com/2013/02/21/architecting-a-cloud-based-ip-multimedia-system-ims/ , [Accessed: March 10, 2013].

[10] T. Yang, X. Wen, Y. Sun, Z. Zhao, and Y. Wang, "A new architecture of HSS based on cloud computing," in *Communication Technology (ICCT), 2011 IEEE 13th International Conference on*, pp. 526–530, 2011. DOI: 10.1109/ICCT.2011.6157931.

[11] G. Ziegler, "Runtime test configurations for load testing. Lecture notes. Ericsson Hungary Ltd.," May 2007. [Online]. Available: http://www.ttcn-3.org/TTCN3UC2007/Presentations/Fri/Session%204/ziegler-Runtime_test_configurations.pdf , [Accessed: May 02, 2013].

[12] G. Din, "An IMS Performance Benchmark Implementation based on the TTCN-3 Language," *Int. J. Softw. Tools Technol. Transf.*, vol. 10, pp. 359–370, July 2008. DOI : 10.1007/s10009-008-0078-x.

[13] N. Kalaichelvan, "Distributed Traffic Load Scheduler based on TITANSim for System Test of a Home Subscriber Server (HSS)," Master's thesis, KTH, School of Information and Communication Technology (ICT), 2011. Online]. Available: http://urn.kb.se/resolve?urn=urn:nbn:se:kth:diva-50066, [Accessed: May 2nd, 2013].

[14] D. Thiben, J. M. E. Carlín, and R. Herpertz, "Evaluating the Performance of an IMS/NGN Deployment," in *GI Jahrestagung'09*, pp. 2561–2573, 2009. Online]. Available: http://subs.emis.de/LNI/Proceedings/Proceedings154/article5154.html, [Accessed: May 5th, 2013].

[15] Adel Al-Hezmi, Christian Riede, Oliver Friedrich, Stefan Arbanowski, and Thomas Magedanz, "Cross-fertilization of IMS and IPTV services over NGN," (Geneva), pp. 153–160, IEEE, May 2008. DOI: 10.1109/KINGN.2008.4542261.

[16] J.-L. Chen, S.-L. Wuy, Y. Larosa, P.-J. Yang, and Y.-F. Li, "IMS cloud computing architecture for high-quality multimedia applications," in *Wireless Communications and Mobile Computing Conference (IWCMC), 2011 7th International*, pp. 1463–1468, July. DOI: 10.1109/IWCMC.2011.5982754.

[17] ITU-T, "Global Information Infrastructure, Internet Protocol Aspects and Next Generation Networks, Frameworks and functional architecture models, general overview of NGN," Tech. Rep. ITU-T Recommendation Y.2001, Dec. 2004. [Accessed: January 26th, 2013].

[18] S. Esaki, A. Kurokawa, and K. Matsumoto, "NTT Technical Review - Overview of Next Generation Network," Tech. Rep. Vol. 5, NTT, June 2007. [Accessed: January 27th, 2013].

[19] ITU-T, "Series Y: Global Information Infrastructure, Internet Protocol Aspects and Next - Generation Networks Next Generation Networks - Frameworks and functional architecture models," Tech. Rep. ITU-T Recommendation Y.2012, Sep. 2006. [Accessed: January 27th, 2013].

[20] ETSI, "Telecommunications and Internet converged Services and Protocols for Advanced Networking (TISPAN); IP Multimedia Subsystem (IMS); Stage 2 description," Tech. Rep. ETSI TS 182 006 V1.1.1, March 2006. [Accessed: January 27th, 2013].

[21] S. Dixit, "On Fixed-Mobile Network Convergence," *Wireless Personal Communications*, vol. 38, pp. 55–65, June 2006. DOI: 10.1007/s11277-006-9042-9 [Accessed: January 24th, 2013].

[22] K.-D. Chang, C.-Y. Chen, J.-L. Chen, and H.-C. Chao, "Challenges to Next Generation Services in IP Multimedia Subsystem," *Journal of Information Processing Systems*, vol. 6, pp. 129–146, June 2010. DOI: 10.3745/JIPS.2010.6.2.129 [Accessed: January 30th, 2013].

[23] 3GPP, "IP Multimedia Subsystem (IMS)," Tech. Rep. 3GPP TS 23.228 V12.0.0 , Stage 2 Release 12, March 2013. [Accessed: March 30th, 2013].

[24] ETSI, "Digital cellular telecommunications system (Phase 2+); Universal Mobile Telecommunications System (UMTS); LTE; Network architecture (3GPP TS 23.002 version 9.3.0 Release 9)," Tech. Rep. ETSI TS 123 002 V9.3.0, June 2010. [Accessed: March 30th, 2013].

[25] J. Rosenberg, H. Schulzrinne, G. Camarillo, A. Johnston, J. Peterson, R. Sparks, M. Handley, and E. Schooler, "SIP: session initiation protocol." Internet Request for Comments, RFC 3261 (Proposed Standard), June 2002. [Online]. Available: http://www.ietf.org/rfc/rfc3261.txt , [Accessed: January 31th, 2013].

[26] B. Aboba, M. A. Beadles, J. Arkko, and P. Eronen, "The Network Access Identifier." Internet Request for Comments, RFC 4282 (Proposed Standard), Dec. 2005. [Online]. Available: http://www.rfc-editor.org/rfc/rfc4282.txt , [Accessed: February 2nd, 2013].

[27] R. Buyya, C. S. Yeo, S. Venugopal, J. Broberg, and I. Brandic, "Cloud computing and emerging IT platforms: Vision, hype, and reality for delivering computing as the 5th utility," *Future Generation Computer Systems*, vol. 25, no. 6, pp. 599 – 616, 2009. DOI: 10.1016/j.future.2008.12.001.

[28] L. M. Vaquero, L. Rodero-Merino, J. Caceres, and M. Lindner, "A break in the clouds: towards a cloud definition," *SIGCOMM Comput. Commun. Rev.*, vol. 39, pp. 50–55, Dec 2008.

[29] J. Geelan, "Twenty-One Experts Define Cloud Computing," *Cloud Computing Journal*, Jan. 2009. [Online]. Available http://virtualization.sys-con.com/node/612375. [Accessed: February 9th, 2013].

[30] P. Mell and T. Grance, "The NIST Definition of Cloud Computing," *NIST Special Publication 800-145*, p. 7, sep. 2011. [Online]. Available: http://csrc.nist.gov/publications/PubsSPs.html#800-145, [Accessed: February 9th, 2013].

[31] G. Pallis, "Cloud Computing: The New Frontier of Internet Computing," *Internet Computing, IEEE*, vol. 14, pp. 70 –73, Sept.-Oct. 2010.

[32] J. Peng, X. Zhang, Z. Lei, B. Zhang, W. Zhang, and Q. Li, "Comparison of Several Cloud Computing Platforms," in *Information Science and Engineering (ISISE), 2009 Second International Symposium on*, pp. 23 –27, Dec. 2009.

[33] X. Xu, "From cloud computing to cloud manufacturing," *Robotics and Computer-Integrated Manufacturing*, vol. 28, pp. 75 – 86, Feb. 2012. DOI: 10.1016/j.rcim.2011.07.002.

[34] X. Chen, G. Wills, L. Gilbert, and D. Bacigalupo, "Using Cloud for Research: A Technical Review," June 2010. [Online]. Available: http://eprints.soton.ac.uk/271273/, [Accessed: February 17th, 2013].

[35] M. Mollah, K. Islam, and S. Islam, "Next generation of computing through cloud computing technology," in *Electrical Computer Engineering (CCECE), 2012 25th IEEE Canadian Conference on*, pp. 1 –6, may 2012.

[36] J. Varia, "Amazon Web Services - Architecting for the cloud: Best practices," Jan. 2011. [Online]. Available: http://media.amazonwebservices.com/AWS_Cloud_Best_Practices.pdf, [Accessed: February 23, 2013].

[37] M. Armbrust, A. Fox, R. Griffith, A. D. Joseph, R. Katz, A. Konwinski, G. Lee, D. Patterson, A. Rabkin, I. Stoica, and M. Zaharia, "A View of Cloud Computing," *Commun. ACM*, vol. 53, pp. 50–58, Apr. 2010. DOI: 10.1145/1721654.1721672.

[38] N. Kshetri, "Cloud Computing in Developing Economies," *Computer*, vol. 43, pp. 47 –55, oct. 2010.

[39] "VMM - Virtual Machine Manager," March 2013. [Online]. Available: http://virt-manager.org/ , [Accessed: March 23rd, 2013].

[40] M. T. Jones, "Managing VMs with the Virtual Machine Manager," Oct. 2012. IBM Corporation, [Online]. Available: http://www.ibm.com/developerworks/cloud/library/cl-managingvms/cl-managingvms-pdf.pdf , [Accessed: June 5th, 2013].

[41] Fraunhofer FOKUS NGNI, "FOKUS Home Subscriber Server (FHoSS) - Open Testbed for IMS Technologies." [Online]. Available: http://www.fokus.fraunhofer.de/en/fokus_testbeds/open_ims_playground/components/osims/fhoss/index.html , [Accessed: April 30th, 2013].

[42] Fraunhofer FOKUS NGNI, "OSIMS - The FOKUS Open Source IMS Core." [Online]. Available: http://www.fokus.fraunhofer.de/en/fokus_testbeds/open_ims_playground/components/osims/index.html , [Accessed: March 2nd, 2013], Nov. 2006.

[43] Fraunhofer FOKUS NGNI, "OpenIMSCore.org - The Open Source IMS Core Project." [Online]. Available: http://www.openimscore.org/ , [Accessed: March 2nd, 2013].

[44] IPtel, "SIP Express Router," Nov. 2008. [Online]. Available: http://www.iptel.org/ser/, [Accessed: March 3rd, 2013].

[45] "HAProxy - The Reliable, High Performance TCP/HTTP Load Balancer." [Online]. Available: http://haproxy.1wt.eu , [Accessed: April 17th, 2013].

[46] "Guide to Scaling Web Databases with MySQL Cluster, Accelerating Innovation on the Web," tech. rep., A MySQL White Paper, Oracle, May 2012. [Online]. Available: [http://www.mysql.com/why-mysql/white-papers/guide-to-scaling-web-databases-with-mysql-cluster/ , Accessed: April 17th, 2013].

[47] ETSI, "IMS/NGN Performance Benchmark specification ETSI TS 186 008-1, Part 1: Core Concepts," Tech. Rep. TS 186 008-1, ETSI, Oct. 2007. [Accessed: March 3rd, 2013].

[48] ETSI, "IMS/NGN Performance Benchmark specification ETSI TS 186 008-2, Part 2: Subsystem Configuraiton and Benchmarks," Tech. Rep. TS 186 008-1, ETSI, Oct. 2007. [Accessed: March 3rd, 2013].

[49] ETSI, "IMS/NGN Performance Benchmark specification ETSI TS 186 008-3, Part 3: Traffic Sets and Traffic Profiles," Tech. Rep. TS 186 008-1, ETSI, Oct. 2007. [Accessed: March 3rd, 2013].

[50] Intel, "Understanding the New Performance Benchmark for IP Multimedia Subsystem (IMS) Architecture to Enable Next-Generation Networking (NGN)." White Paper IMS/NGN Performance Benchmark, [Online]. Available: http://download.intel.com/design/telecom/papers/319469.pdf, [Accessed: March 15th, 2013].

[51] Intel, "IMS Bench SIPp," Oct 2010. Published : Oct 25th, 2010, [Online]. Available: http://sipp.sourceforge.net/ims_bench/intro.html, [Accessed: March 3rd, 2013].

[52] Zabbix, "Zabbix - The Enterprise-class Monitoring Solution for Everyone." Founded in 2001, [Online]. Available: http://www.zabbix.com/, [Accessed: March 19th, 2013].

[53] S. Galiano Molina, "Leaf Project - A book streaming platform : Infrastructure," Master's thesis, KTH, School of Information and Communication Technology (ICT). TRITA-ICT-EX-2011:177, August 2011. [Online]. Available: http://kth.diva-portal.org/smash/record.jsf?pid=diva2:460936 , [Accessed: April 29th, 2013].

[54] "LVS - Linux Virtual Server." [Online]. Available: http://www.linuxvirtualserver.org/HighAvailability.html, [Accessed: April 29th, 2013].

[55] MySQL, "MySQL and DRBD High Availability Architectures," May 2007. A MySQL Technical White Paper, [Online]. Available: http://www.oszone.co.kr/sites/default/files/mysql_wp_drbd.pdf, [Accessed: April 29th, 2013].

[56] Oracle, "High Availability Concepts and Best Practices." Oracle9i Real Application Clusters Concepts, Release 1 (9.0.1), Part Number A89867-02, [Online]. Available: http://docs.oracle.com/cd/A91202_01/901_doc/rac.901/a89867/pshavdtl.htm , [Accessed: April 29th, 2013].

[57] "Keepalived - Load balancing and High Availability ." [Online]. Available: http://www.keepalived.org , [Accessed: April 13th, 2013].

[58] A. Cassen, "Keepalived for LVS," 2002. [Online]. Available: http://www.keepalived.org/pdf/UserGuide.pdf , [Accessed: April 30th, 2013].

[59] D. Vingarzan, P. Weik, and T. Magedanz, "Design and Implementation of an Open IMS Core," in *Mobility Aware Technologies and Applications* (T. Magedanz, A. Karmouch, S. Pierre, and I. Venieris, eds.), vol. 3744 of *Lecture Notes in Computer Science*, pp. 284–293, Springer Berlin Heidelberg, 2005. ISBN: 978-3-540-29410-8.

[60] "myMonster - Telco Communicator Suite." [Online]. Available: http://www.monster-the-client.org/overview/overview/index.html , [Accessed: April 10th, 2013].

[61] A. Johnston, S. Donovan, R. Sparks, C. Cunningham, and K. Summers, "Session Initiation Protocol (SIP) Basic Call Flow Examples." Internet Request for Comments, RFC 3665 (Best Current Practice), Dec. 2003. [Online]. Available: http://www.rfc-editor.org/rfc/rfc3665.txt, [Accessed: March 28th, 2013].

[62] "BoneFIRE Documentation - Exporting Monitoring Data to CSV." [Online]. Available: http://doc.bonfire-project.eu/R3.1/monitoring/getting-data/export-csv.html, [Accessed: June 3rd, 2013].

[63] D. Malas and A. Morton, "Basic Telephony SIP End-to-End Performance Metrics." Internet Engineering Task Force (IETF), RFC 6076 (Standards Track), ISSN: 2070-1721, January. 2011. [Online]. Available: http://tools.ietf.org/html/rfc6076, [Accessed: June 13th, 2013].

[64] M. Voznak and J. Rozhon, "SIP Registration Stress Test," in *Proceedings of the 6th international conference on Communications and Information Technology, and Proceedings of the 3rd World conference on Education and Educational Technologies*, WORLD-EDU'12/CIT'12, (Stevens Point, Wisconsin, USA), pp. 101–105, World Scientific and Engineering Academy and Society (WSEAS), 2012. ISBN: 978-1-61804-077-0.

[65] C. Makaya, A. Dutta, S. Das, D. Chee, F. Lin, S. Komorita, H. Yokota, and H. Schulzrinne, "Service continuity support in self-organizing ims networks," in *Wireless Communication, Vehicular Technology, Information Theory and Aerospace Electronic Systems Technology (Wireless VITAE), 2011 2nd International Conference on*, pp. 1–5, 2011. DOI: 10.1109/WIRELESSVITAE.2011.5940890.

[66] H. Uppal and D. Brandon, "OpenFlow Based Load Balancing," Project Report CSE561: Networking, University of Washington. [Online]. Available: http://people.cs.umass.edu/~hardeep/cse561_openflow_project_report.pdf, [Accessed: June 20th, 2013].

[67] M. Koerner and O. Kao, "Multiple service load-balancing with OpenFlow," in *High Performance Switching and Routing (HPSR), 2012 IEEE 13th International Conference on*, pp. 210–214, 2012. DOI: 10.1109/HPSR.2012.6260852.

[68] Acme Packet Inc., "Net-Net Diameter Director," December 2012. Acme Packet, Inc., Bedford, MA, USA, Data sheet, [Online]. Available: http://www.acmepacket.com/collateral/acm/datasheet/APKT_DS_NetNet DD.pdf , [Accessed: June 20th, 2013].

[69] Elitecore Technologies Pvt. Ltd, "ELITEDSC Usecase - HSS Mapping." [Online]. Available: http://www.elitecore.com/telecompractices/HSS-Mapping.html , [Accessed: June 20th, 2013].

[70] P. Bhadrapur, "HSS Front-End implementation for a large scale common HLR/HSS," Master's thesis, Uppsala University, Department of Information Technology, 2012. Series: IT 12 048, [Online]. Available: http://urn.kb.se/resolve?urn=urn:nbn:se:uu:diva-182248 , [Accessed: June 20th, 2013].

[71] V. Cackovic and Z. Popovic, "The use of IMS for cloud based services control and management," in *MIPRO, 2012 Proceedings of the 35th International Convention*, pp. 501–505, 2012. Print ISBN: 978-1-4673-2577-6.

[72] M. Gutierrez and N. Ventura, "Mobile Cloud Computing based on service oriented architecture: Embracing network as a service for 3^{RD} party application service providers," in *Kaleidoscope 2011: The Fully Networked Human? - Innovations for Future Networks and Services (K-2011), Proceedings of ITU*, pp. 1–7, 2011. Print ISBN: 978-1-4577-1935-6.

Appendix A

Installation and Configuration of an OpenIMS Core

A.1 Prerequisites

First prepare the environment by installing the following packages.

sudo apt-get install debhelper cdbs lintian build-essential fakeroot devscripts pbuilder dh-make debootstrap dpatch flex libxml2-dev libmysqlclient15-dev sun-java6-jdk ant docbook-to-man

A.2 Installation and Configuration of CSCFs

A.2.1 Get the Source Code

Create a new directory "ser_ims" at a default location "/opt/OpenIMSCore". Afterwords, get the source code which is available at `http://svn.berlios.de/svnroot/repos/openimscore/`.

cd /opt/OpenIMSCore
mkdir ser_ims
svn checkout http://svn.berlios.de/svnroot/repos/openimscore/ser_ims/trunk ser_ims

A.2.2 Compile

Go inside the directory "ser_ims" and compile the source code by executing the make libs install command.

cd ser_ims
make install-libs all

A.2.3 Configure

First copy all the *.xml, *.cfg and *.sh files from "ser_ims/cfg/" to "/opt/OpenIMSCore". Afterwords, use the "configurator.sh" to configure with specific IP address and Domain name. By default all the components are configured with "127.0.0.1" and "open-ims.test".

```
cp ser_ims/cfg/*.cfg .
cp ser_ims/cfg/*.xml .
cp ser_ims/cfg/*.sh .
```

A.3 Installation and Configuration of HSS

A.3.1 Get the Source Code

First create a new directory "FHoSS" at a default location "/opt/OpenIMSCore". Afterwords, get the source code which is available at `http://svn.berlios.de/svnroot/repos/openimscore/`.

```
cd /opt/OpenIMSCore
mkdir FHoSS
svn checkout http://svn.berlios.de/svnroot/repos/openimscore/FHoSS/trunk FHoSS
```

A.3.2 Compile

First set the environment variable "JAVA_HOMA" and go inside the directory "FHoSS". Afterwords, build the binaries from source by executing the following command.

```
cd FHoSS
ant compile deploy
```

A.3.3 Configure

For configuration of Home Subscriber Server (HSS), first create a MySQL database using the following scripts and populate it with the default configuration.

```
pwd
/opt/OpenIMSCore
mysql -u root -p < FHoSS/scripts/hss_db.sql
mysql -u root -p < FHoSS/scripts/userdata.sql
mysql -u root -p < ser_ims/cfg/icscf.sql
```

At this point, MySQL database is configured and working properly. Now take a look into the HSS configuration files which exists inside the directory

"FHoSS/deploy/".

DiameterPeerHSS.xml : It provides a peer configuration parameters such as FQDN, Realm, Acceptor Port or Authorized identifiers.

hibernate.properties : It provides a hibernate configuration parameters; by default MySQL server was configured on a local host (127.0.0.1:3306) and hibernate connect to MySQL server via JDBC connector.

hss.properties : It provides a configuration parameter that is relevant to the FHoSS web interface.

log4j.properties : It provides a logging configuration.

A.4 DNS Server Configuration

Modify the DNS server zone file according to the IP addresses of CSCFs and HSS, by default all the components are configured with localhost. First copy the zone file "open-ims.dnszone" into "/etc/bind/" directory.

```
cp ser_ims/cfg/open-ims.dnszone /etc/bind/
```

Add the following piece of lines into the file "/etc/bind/named.local"

```
zone "open-ims.test" IN {
type master;
file "/etc/bind/open-ims.dnszone";
notify no;
};
```

Afterwords, run the DNS server by executing the following command.

```
/etc/init.d/bind9 start
```

Furthermore, add the following lines into a file "/etc/resolv.conf" to ensure DNS server working properly,

```
search open-ims.test
domain open-ims.test
```

A.5 Run the OpenIMS Core

After successful installation and configuration of an OpenIMS core, start the OpenIMS core components by exciting the following scripts.

```
pwd
/opt/OpenIMSCore
./pcscf.sh
./icscf.sh
./scscf.sh
FHoSS/deploy/startup.sh
```

Appendix B

Installation and Configuration of MySQL Cluster

B.1 Download MySQL Cluster Software

MySQL cluster is an Open Source software and can be downloaded the latest version of MySQL cluster software form a site http://www.mysql.com/downloads/cluster/ according the operating system.

B.2 Installation and Configuration of Management Node

This step described the installation and configuration of Management node of a MySQL cluster.

B.2.1 Installation of Management Node

Locate the MYSQL cluster software that you have downloaded, and step up the management node environment.

```
pwd
/net/u/mum/Downloads
mv mysql-cluster-gpl-7.2.12-linux2.6-x86_64.tar.gz /usr/local
adduser mysql
addgroup mysql
cd /usr/local
tar xvfz mysql-cluster-gpl-7.2.12-linux2.6-x86_64.tar.gz
ln -s mysql-cluster-gpl-7.2.12-linux2.6-x86_64 mysql-cluster
cd mysql-cluster
chown -R mysql:mysql /usr/local/mysql-cluster/
```

At this point, you have downloaded and step up the management node, the configuration of management node described in next Section.

B.2.2 Configuration of Management Node

The management node stores and manage the global configuration file named as "config.ini" of a MySQL cluster, which describes all the information of the MYSQL cluster about the number of management nodes, MySQL server nodes and data nodes. The example of configuration file is shown in Figure B.1. In this example, one management node, two data nodes and 2 MySQL nodes are configured for a cluster. Create a "config" directory and a config.ini file using a editor.

pwd
/usr/local/mysql-cluster
mkdir config
cd config
vim config.ini

Management Node

```
#####################################
#       GLOBAL CONFIGURATION        #
#####################################

##############Management Node Parameter###############
[ndb_mgmd default]
DataDir=/usr/local/mysql-cluster/data/m_data
PortNumber=1186

[ndb_mgmd]
NodeId = 1
HostName = mgmd



#############Data Storage Node Parameters#############
[ndbd default]
NoOfReplicas = 2
DataDir= /usr/local/mysql-cluster/data/m_data

[ndbd]
NodeId = 2
HostName = ndbd1

[ndbd]
NodeId = 3
HostName = ndbd2


#################Application Node Parameters###########
[mysqld]
NodeId = 4
[mysqld]
NodeId = 5
                                                    1,1           All
```

Figure B.1. Example of an Global Configuration of a MySQL Cluster

B.3 Installation and Configuration of Data Node

The data node installation configuration steps are described in this section.

B.3.1 Installation of Data Node

Download the MySQL cluster software if you are using separate virtual machine.

```
pwd
/net/u/mum/Downloads
mv mysql-cluster-gpl-7.2.12-linux2.6-x86_64.tar.gz /usr/local
adduser mysql
addgroup mysql
cd /usr/local
tar xvfz mysql-cluster-gpl-7.2.12-linux2.6-x86_64.tar.gz
ln -s mysql-cluster-gpl-7.2.12-linux2.6-x86_64 mysql-cluster
cd mysql-cluster
chown -R mysql:mysql /usr/local/mysql-cluster/
sudo apt-get install libaio1
```

B.3.2 Configuration of Data Node

The MySQL cluster data nodes doesn't have a separate configuration file, but you have to install the database tables by exacting the following command for only first time.

```
scripts/mysql_install_db –user=mysql Ðdatadir=/usr/local/mysql-cluster/data/m_data
```

It is important that datadir has the same path, which is specified in the "config.ini" file of the management node.

B.4 Installation and Configuration of Application Node

The installation and configuration of application nodes are described in this section.

B.4.1 Installation of Application Node

Download the MySQL cluster software if you are using separate virtual machine.

```
pwd
/net/u/mum/Downloads
mv mysql-cluster-gpl-7.2.12-linux2.6-x86_64.tar.gz /usr/local
adduser mysql
addgroup mysql
cd /usr/local
```

tar xvfz mysql-cluster-gpl-7.2.12-linux2.6-x86_64.tar.gz
ln -s mysql-cluster-gpl-7.2.12-linux2.6-x86_64 mysql-cluster
cd mysql-cluster
chown -R mysql:mysql /usr/local/mysql-cluster/
sudo apt-get install libaio1

B.4.2 Configuration of Application Node

The application nodes of MySQL manages a local configuration file named as "my.cnf", which contains the MySQL cluster parameters and mainly the management node address in order to figured out number of data nodes are participating in the MySQL cluster. The Figure B.2 shows an example of local configuration of a MySQL cluster. Create a directory and a "my.cnf" file using a editor.

pwd
/usr/local/mysql-cluster/data
mkdir s_data
scripts/mysql_install_db –user=mysql Ðdatadir=/usr/local/mysql-cluster/data/s_data
mkdir config
cd config
vim my.cnf

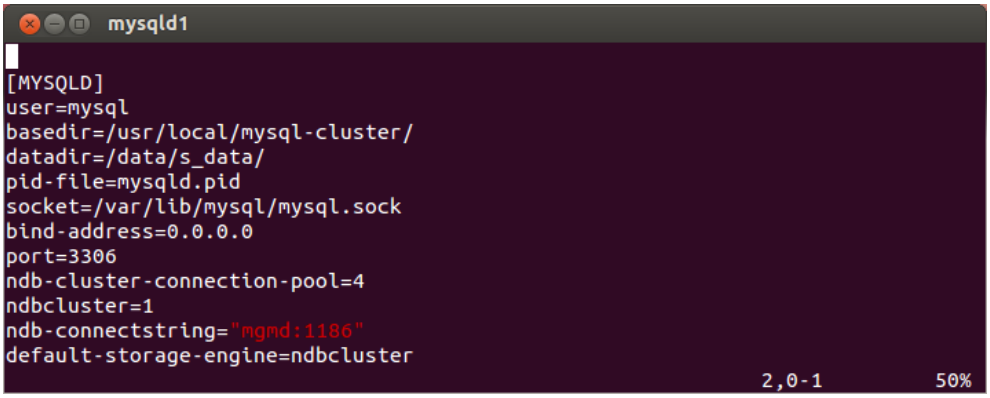


Figure B.2. Example of an Local Configuration of a MySQL Cluster

B.5 Starting of an MYSQL Cluster

MySQL Cluster nodes are started in the following order. The following step applies to all the cluster nodes. First, login into the Virtual machine where cluster nodes are installed using the ssh command.

ssh VM_IP_Address

After login into the Virtual machine. Go inside the directory. It is important to start the cluster nodes as a **"root"**.

cd /usr/local/mysql-cluster

B.5.1 Starting of Management Server and Client

Management server is started by executing the following command;

bin/ndb_mgmd –initial -f /usr/local/mysql-cluster/config/config.ini
–configdir=/usr/local/mysql-cluster/config

After executing the above command, the following message will displayed on the terminal window.

MySQL Cluster Management Server mysql-5.5.28 ndb-7.2.12

After words, start the management client by executing the following command;

bin/ndb_mgm –ndb-connectstring=mgmd:1186

It will start the Management client and enter into the management console, where you can manage all the MySQL cluster nodes including management server.

B.5.2 Starting of Data Node

Data nodes are started by executing the following command;

bin/ndbd –ndb-connectstring=mgmd:1186

After started, the following messages will displayed on the terminal window.

2013-04-24 12:17:04 [ndbd] INFO – Angel connected to 'mgmd:1186'
2013-04-24 12:17:04 [ndbd] INFO – Angel allocated nodeid: 2

B.5.3 Starting of MYSQL Server Node

MySQL daemon is started by executing the following command;

bin/mysqld_safe –defaults-extra-file=/usr/local/mysql-cluster/config/my.cnf
–datadir=/usr/local/mysql-cluster/data/s_data/ –basedir=/usr/local/mysql-cluster
–lc-messages-dir="/usr/local/mysql-cluster/share/english/" &

B.6 MySQL Client User Privileges Configuration

Login into MySQL server node on another terminal using ssh. Run the MySQL client locally by executing the following command and set the user privileges for the remote access.

```
pwd
/usr/local/mysql-cluster/bin/mysql
```

It will open the mysql console. Add the root users by exciting the following commands,

```
mysql> GRANT USAGE ON *.* TO root@'%' IDENTIFIED BY 'password';
mysql> GRANT USAGE ON *.* TO root@'IP_Address_MySQL_Server_Node_1'
IDENTIFIED BY 'password';
mysql> GRANT USAGE ON *.* TO root@'IP_Address_MySQL_Server_Node_2'
IDENTIFIED BY 'password';
```

It is important to specify the MySQL Server Nodes IP address and the "password" that needs to be used for MySQL client.

```
mysql> GRANT ALL PRIVILEGES ON *.* TO root@'%';
mysql> GRANT ALL PRIVILEGES ON *.* TO root@'IP_Address_MySQL_Server_1'
mysql> GRANT ALL PRIVILEGES ON *.* TO root@'IP_Address_MySQL_Server_2'
```

Also, add a haproxy user by executing the following command, if in case you are using HAProxy load balancer.

```
mysql> GRANT USAGE ON *.* TO haproxy@'%';
```

Appendix C

Installation and Configuration of IMS Bench SIPp

C.1 Pre-requisites

You have to install the following necessary packages:

- Make sue you have installed gcc-4.4 C++ compiler.
- Install the GSL library for the random number generation for the statistical distributions. GSL library can be download from `http://www.gnu.org/software/gsl` and compiled from sources by executing the following commands:

  pwd
  /root/gsl-1.9/
  ./configure
  make
  make install
  echo /usr/local/lib/ »/etc/ld.so.conf

- Install the Perl XML and Gnuplot for to able to use benchmark configuration using the menu-driven tool and the report generation tool, execute the following commands for installation:

  Perl XML::Simple module can be downloaded from
  http://search.cpan.org/dist/XML-Simple/
  pwd
  /root/XML-Simple/
  perl -MCPAN -e shell
  install XML::Simple

```
make install
quit
```

```
Gnuplot 4.2 can be downloaded from http://gnuplot.sourceforge.net/
pwd
/root/gnuplot-4.2.0/
./configure –without-x
make
make install
```

C.2 Download the IMS Bench SIPp source

IMS Bench SIPp is an open source software released under the GNU GPL license and can be downloaded from the subversion repository at:

```
svn co https://sipp.svn.sourceforge.net/svnroot/sipp/sipp/branches/ims_bench ims_bench
```

C.3 Build IMS Bench SIPp source tree

Execute the following commands in order to build the manager and SIPp:

```
pwd
/root/ims_bench/
make rmtl
make ossl
make mgr
```

C.4 Configuration of IMS Bench SIPp

The test system for a benchmark run can be configured using an menu driven user interface and automatically generate all the necessary execution scripts and configuration files. Run the following built-in perl script:

```
pwd
/root/ims_bench/
./scripts/ims_bench.pl
```

After executing the above command, You have see the IMS Benchmark configuration main menu as shown in Figure C.1

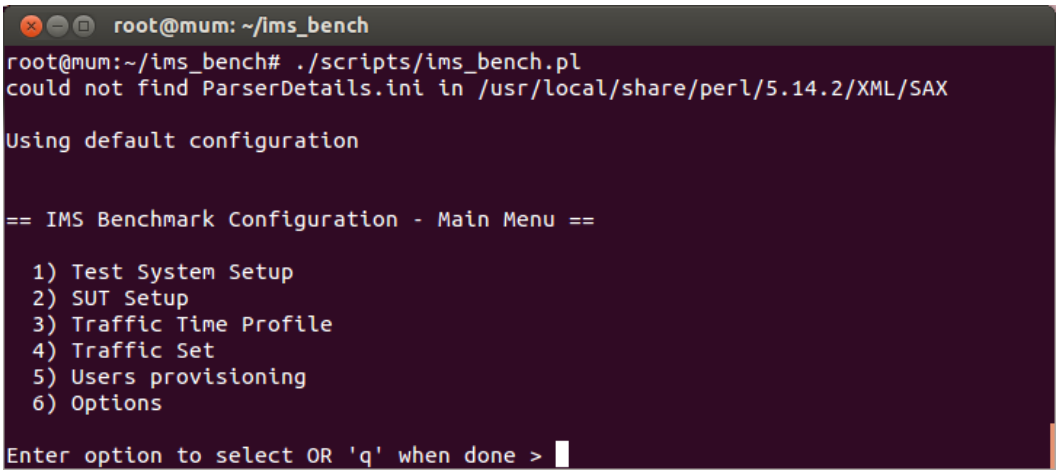


Figure C.1. IMS benchmark configuration menu

Press '1' to go inside the test system menu and configured the manager and SIPp TS instance IP addresses. Figure C.2 shows an example of the test system configuration. Press 'q' to return to main menu.

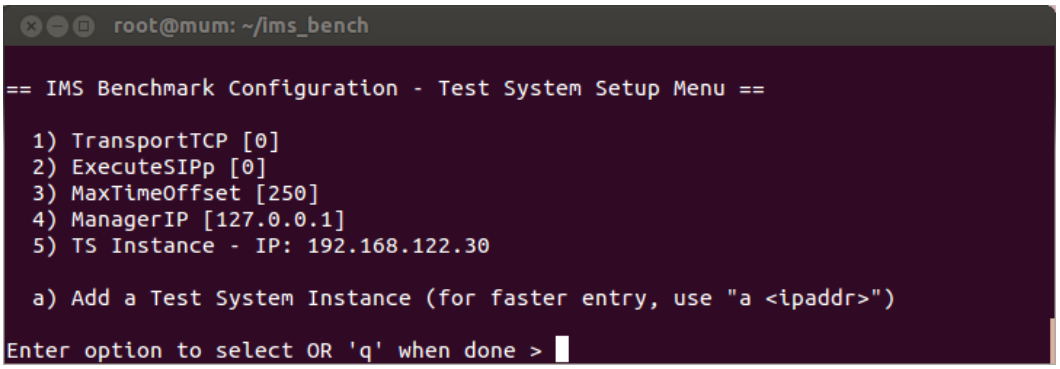


Figure C.2. Test system configuration menu

Press '2' to go inside the SUT setup menu and configured the P-CSCF VM IP address. Figure C.3 shows an example of the SUT configuration. Press 'q' to return to main menu.

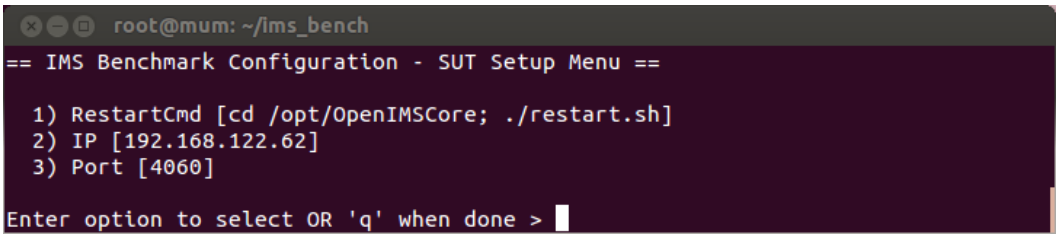


Figure C.3. SUT configuration menu

Press '5' to go inside the user provisioning menu and configured the public and private user identities. Figure C.4 shows an example of the user provisioning configuration. Press 'q' to return to main menu.

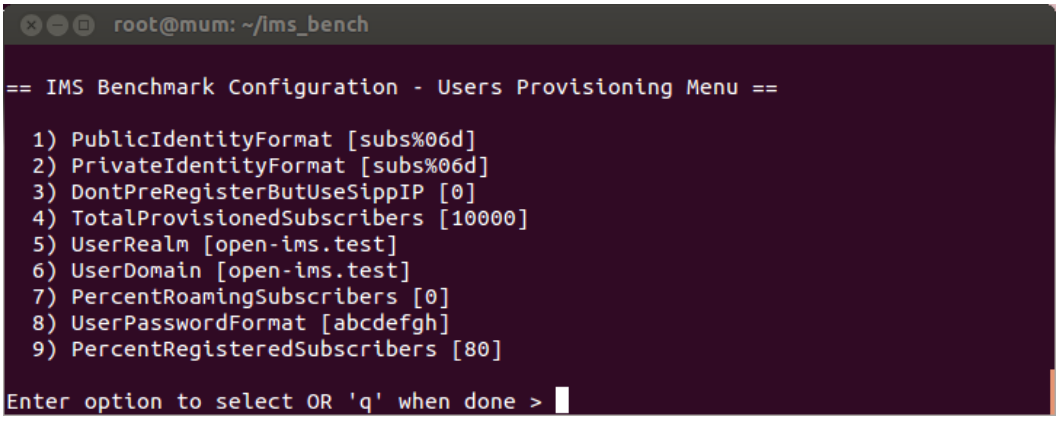


Figure C.4. User provisioning menu

Press 'q' to generate the configuration files and execution scripts in the directory named "ims_bench_7".

C.5 Run Benchmark Test

All the benchmarking configuration files and execution scripts are exist inside the directory named 'ims_bench_7". Go inside to the directory, edit the manager configuration "manager.xml" file and run the manager by executing the following commands:

pwd
/root/ims_bench/Ims_becnh_7
../manager -f manager.xml

Above command runs the manager, open the new terminal and execute the following commands to run the SIPp instance.

pwd
/root/ims_bench/Ims_becnh_7
./run_1.sh

After running manger and SIPp client, click on the manager terminal and press 'e' to start the benchmarking test.

Appendix D

Installation and Configuration of Zabbix

D.1 Installation of Zabbix Agent

The Zabbix agent can be installed on a Ubuntu machine by executing the following command.

```
sudo apt-get install zabbix-agent
```

D.2 Configuration of Zabbix Agent

Edit the Zabbix agent configuration "/etc/zabbix/zabbix_agentd.conf". Modify "Server" with the IP address of the Zabbix server machine and specify the hostname of the Zabbix agent machine in the "Hostname" line and then restart the Zabbix agent.

```
sudo vim /etc/zabbix/zabbix_agentd.conf
Server=<Zabbix server ip address>
Hostname=zabbix-agent-hostname
sudo vim /etc/init.d/zabbix_agentd.conf restart
```

D.3 Installation of Zabbix server

The Zabbix server can be installed on a Ubuntu virtual machine by executing the following command.

```
sudo apt-get install zabbix-server-mysql
sudo apt-get install zabbix-frontend-php
```

D.4 Access to Zabbix web console

Zabbix web console can be accessed using a favoite browser by entering the zabbix server IP address in the following url. Log in with default setting, the username "Admin" and the password "zabbix".

http://<zabbix server ip address>/zabbix

D.5 Configuration of auto-registration

Create an action for auto-registration of the hosts. Go to the following tabs from with zabbix web console:

Configuration > Actions > Create action

Afterwords, enter the action name like "registration", select the event source to "auto registration" and attached the action operation with "Template_Linux" for the default items. Figure D.1 illustrates an example of Auto-registration of hosts.

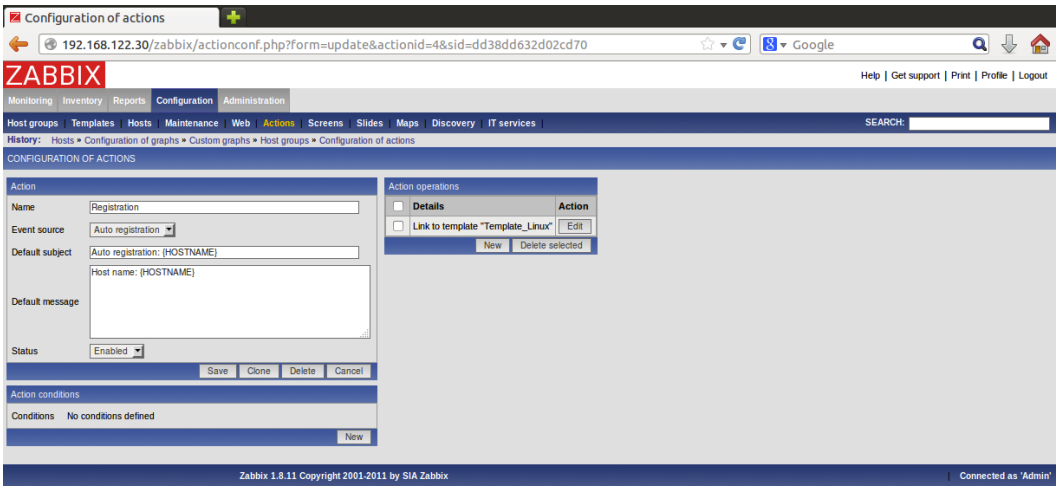


Figure D.1. Example of an auto-registration of hosts

After auto-registration, click on the hosts tab which indicates the registered hosts. Figure D.2 illustrates an example of an automatic registered hosts.

Configuration > Hosts

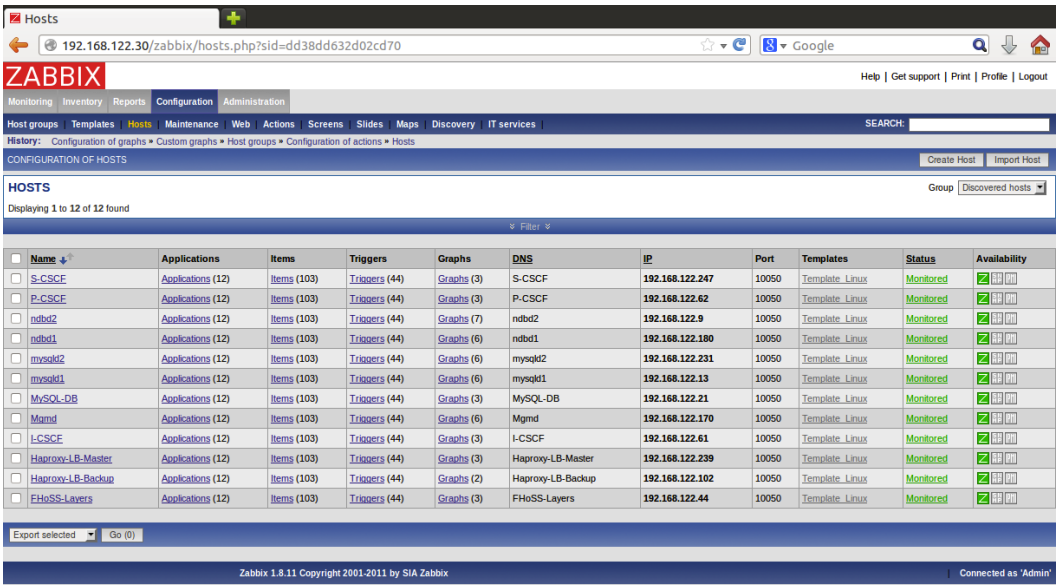


Figure D.2. Example of an automatic registered hosts

Appendix E

Installation and Configuration of HAProxy

E.1 Installation of HAProxy

The HAProxy can be installed on a Ubuntu machine by executing the following command:

```
sudo apt-get install haproxy
```

E.2 Configuration of HAProxy

Edit the HAProxy configuration "/etc/haproxy/haproxy.cfg" based upon the requirement. Example of an HAProxy configuration for our project is shown in Figure E.1.

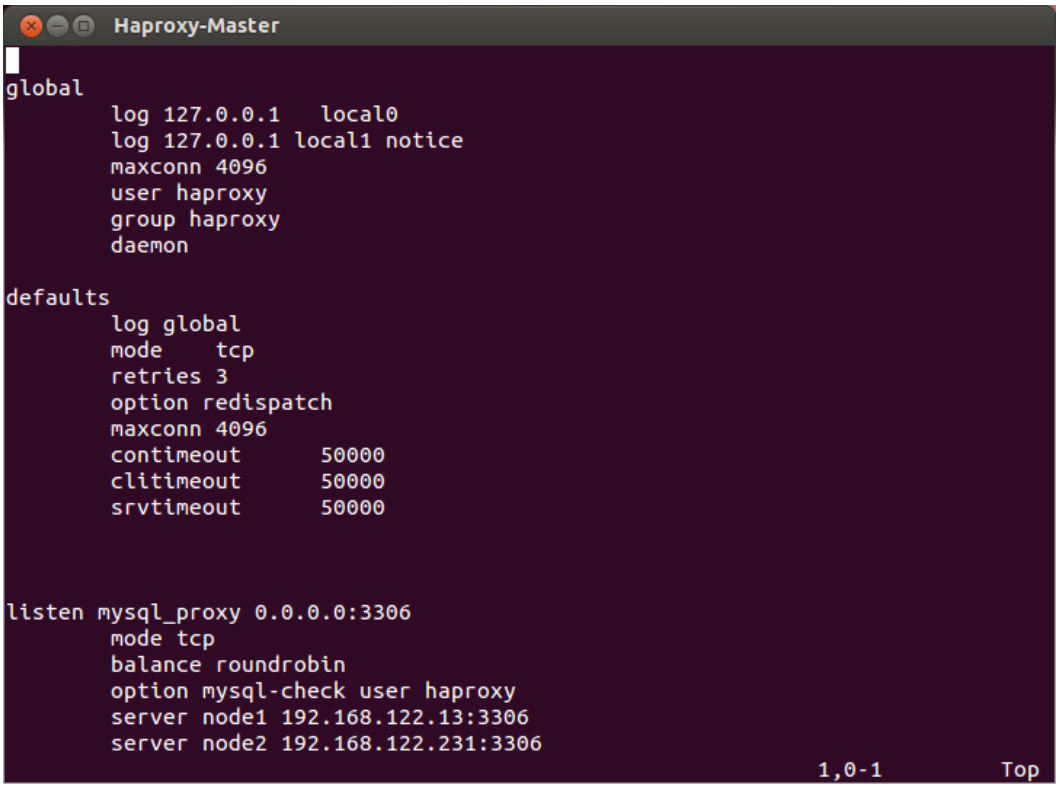


Figure E.1. Example of an HAProxy configuration

Appendix F

Installation and Configuration of Keepalived

F.1 Installation of Keepalived

The Keepalive can be installed on a Ubuntu machine by executing the following command.

```
sudo apt-get install -y keepalived
```

F.2 Configuration of Keepalived

This section described the configuration setting for the Keeplaived.

F.2.1 Kernel Binding

Execute the following command on the terminal, which tells the kernel to allow non-local IP address binding into the hosts. The following kernel binding configuration step should be performed on both load balancner machines.

```
echo "net.ipv4.ip_nonlocal_bind = 1" >> /etc/sysctl.conf
```

F.2.2 Keepalived Configuration

The example of an Keepalived configuration "/etc/keepalived/keepalived.conf" is shown in Figure **??**. In this example the virtual IP is configured with "192.168.122.122". The virtual IP must be within the same network of the load balancer virtual machine.

Haproxy-Master

```
vrrp_script chk_haproxy {
      script "killall -0 haproxy"     # verify the pid is exist or not
      interval 2                      # check every 2 seconds
      weight 2                        # add 2 points of prio if OK
}

vrrp_instance VI_1 {
      interface eth0                  # interface to monitor
      state MASTER
      virtual_router_id 51            # Assign one ID for this route
      priority 101                    # 101 on master, 100 on backup
      virtual_ipaddress {
            192.168.122.122                   # the virtual IP
      }
      track_script {
            chk_haproxy
      }


                                                    1,0-1         All
```

(a) Keepalived master node configuration

Haproxy-Backup

```
vrrp_script chk_haproxy {
      script "killall -0 haproxy"     # verify the pid is exist or not
      interval 2                      # check every 2 seconds
      weight 2                        # add 2 points of prio if OK
}

vrrp_instance VI_1 {
      interface eth0                  # interface to monitor
      state MASTER
      virtual_router_id 51            # Assign one ID for this route
      priority 100                    # 101 on master, 100 on backup
      virtual_ipaddress {
            192.168.122.122                   # the virtual IP
      }
      track_script {
            chk_haproxy
      }


                                                    1,0-1         Top
```

(b) Keealived backup node configuration

Figure F.1. Example of an Keepalived Configuration

F.2.3 Verification of Keepalived

Verify the status of the Keepalived by executing the following command on both machines.

```
ip a | grep -e inet.*eth0
```

Appendix G

Miscellaneous

G.1 myMonster Preference Setting

The preference setting of myMONSTER client for creating user profile is shown in Figure G.1.

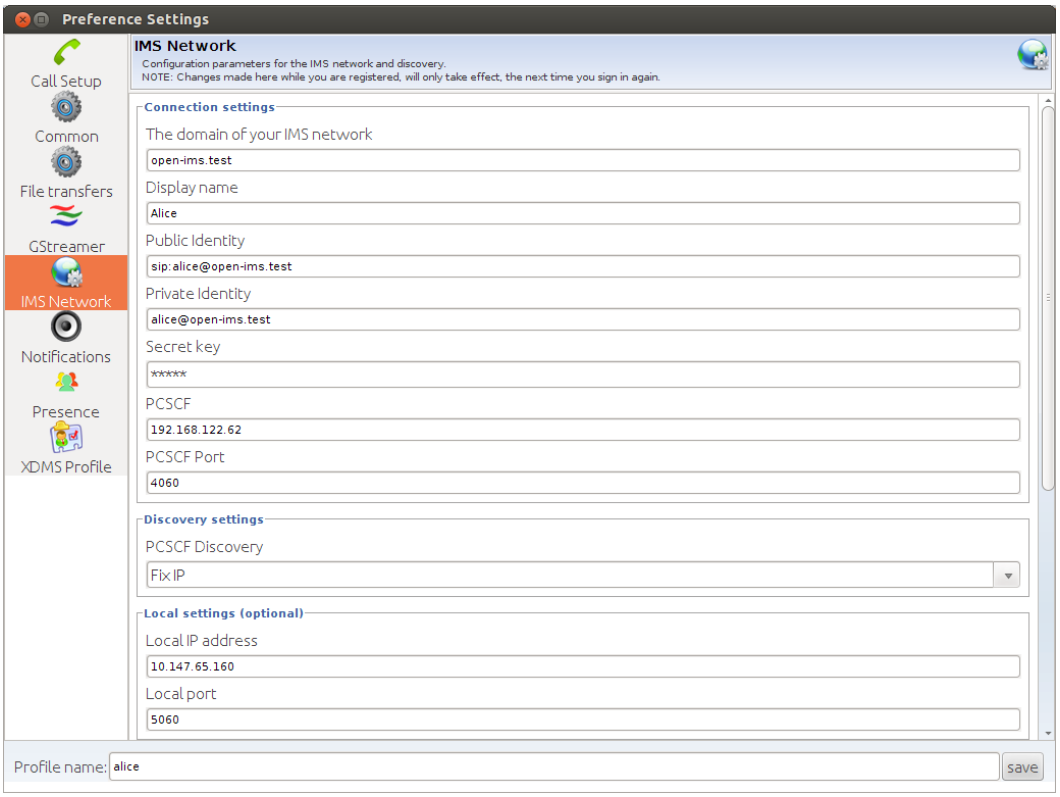


Figure G.1. Preference Setting

TRITA-ICT-EX-2013:178

www.kth.se