ISTANBUL TECHNICAL UNIVERSITY ★ GRADUATE SCHOOL OF SCIENCE ENGINEERING AND TECHNOLOGY

DYNAMIC LOAD MANAGEMENT FOR IMS NETWORKS USING NETWORK FUNCTION VIRTUALIZATION

M.Sc. THESIS

Kaan DANDİN

Department of Electronics and Communications Engineering

Telecommunications Engineering Programme

JANUARY 2015

ISTANBUL TECHNICAL UNIVERSITY ★ GRADUATE SCHOOL OF SCIENCE ENGINEERING AND TECHNOLOGY

DYNAMIC LOAD MANAGEMENT FOR IMS NETWORKS USING NETWORK FUNCTION VIRTUALIZATION

M.Sc. THESIS

Kaan DANDİN
(504931010)

Department of Electronics and Communications Engineering

Telecommunications Engineering Programme

Thesis Advisor: Associate Prof. Dr. Güneş KARABULUT KURT
Co Advisor: Dr.İbrahim HÖKELEK

JANUARY 2015

İSTANBUL TEKNİK ÜNİVERSİTESİ ★ FEN BİLİMLERİ ENSTİTÜSÜ

ŞEBEKE FONKSİYONLARI SANALLAŞTIRMA KULLANARAK IMS ŞEBEKELERİNDE DİNAMİK YÜK PAYLAŞIMI

YÜKSEK LİSANS TEZİ

Kaan DANDİN
(504931010)

Elektronik ve Haberleşme Mühendisliği Anabilim Dalı

Telekomünikasyon Mühendisliği Programı

Tez Danışmanı: Doç. Dr. Güneş KARABULUT KURT
Eş Danışman: Dr.İbrahim HÖKELEK

OCAK 2015

Kaan DANDİN, a **M.Sc.** student of ITU **Institute of / Graduate School of Engineering.** student ID **504931010**, successfully defended the **thesis** entitled "**DYNAMIC LOAD MANAGEMENT FOR IMS NETWORKS USING NETWORK FUNCTION VIRTUALIZATION**", which he prepared after fulfilling the requirements specified in the associated legislations, before the jury whose signatures are below.

Thesis Advisor :	**Assoc. Prof. Dr. Güneş KARABULUT KURT** İstanbul Technical University	....
Co-advisor :	**Dr. İbrahim HÖKELEK** İstanbul Technical University	....
Jury Members :	**Prof. Dr. Hakan Ali Çırpan** İstanbul Technnical University	....
	Prof. Dr. Mehmet Ertuğrul Çelebi İstanbul Technical University	....
	Asst.Prof. Dr. Ali Özer ERCAN Özyeğin University	....

Date of Submission : 12 December 2014
Date of Defense : 14 January 2015

To my spouse and daugters,

FOREWORD

I would like to express my deepest gratitude to Associate Prof. Güneş Karabulut Kurt and Dr.İbrahim Hökelek accepting me as a M.Sc.thesis student and closely following my achievements, giving guidance and motivation with their valuable feedbacks. They have also helped me to arrange necessary tools and working environment in the University Lab.

I would like to also thank my company Vodafone-Oksijen for HW support and patent appliance related with the thesis topic.

Finally , I would like to thank my family, wife and daughters for their support and patience during this period.

December 2014

Kaan DANDİN
(Senior Solution Architect)

TABLE OF CONTENTS

Page

ABBREVIATIONS

AS	: Application Server
ATCA	: Advanced Telecommunications Network Infrastructure
CAPEX	: Capital Expenditure
COTS	: Commercial Off The Shelf
CPU	: Central Processing Unit
CSCF	: Call Session Control Function
CSR	: Call Success Rate
DNS	: Domain Name Services
ETSI	: Europen Telecommmunication Standards Institute
HSS	: Home Subcriber Server
IAAS	: Infrastructure as a Service
I-CSCF	: Interrogating-Call Session Control Function
IHS	: Inadequately Handled Scenarios
IMS	: IP Multimedia Subsystem
IMX	: IP Multimedia eXchange
IOT	: Internet of Things
ISG	: Industry Specification Group
ISUP	: Integrated Services Digital Network User Part
IT	: Information Technologies
LAVM	: List of Active VMs
LB	: Load Balancer
LRUP	: List of Active RUPs for active VMs
LTE	: Long Term Evolution
NFV	: Network Function Virtualization
NSP	: Network Service Provider
OPEX	: Operation Expenditure
OPNFV	: Open Platform for NFV
PAAS	: Platform as a Service
P-CSCF	: Proxy-Call Session Control Function
RUP	: Resource Utilization Percentage
SAAS	: Software as a Service
S-CSCF	: Serving-Call Session Control Function
SDN	: Software Defined Networking
SER	: System Express Router
SIP	: Session Initiation Protocol
SLA	: Service Level Aggrement
SUT	: System Under Test
UAC	: User Agent Client
UAS	: User Agent Server
UE	: User Equipment
QOS	: Quality Of Service
VM	: Virtual Machine
VPN	: Virtual Private Tunnel

LIST OF FIGURES

Page

DYNAMIC LOAD MANAGEMENT FOR IMS NETWORKS USING NETWORK FUNCTION VIRTUALIZATION

SUMMARY

Long Term Evolution (LTE) access technology has become a de facto standard to meet the unprecedented mobile traffic increase. LTE, which has an all IP based architecture, uses IP Multimedia Subsystem (IMS) at the core network for delivering and managing IP multimedia services. IMS systems has several functions like P-CSCF, I-CSCF, S-CSCF and HSS. On the other hand cloud technologies and commercial off the shelf(COTS) HW are evolved in recent years which gives also the possibility to virtualize network functions. Virtualization at telecom networks is also necessary to overcome high capex investment costs coming with high data traffic due to smartphones, Internet of Things(IOT) devices and decreasing revenues. Network Function Virtualization (NFV) is utilized to simplfy the deployment and management of constantly evolving and highly complex networking services. NFV also helps to dynamically scale network resources when capacity is really needed. Using COTS HW will also reduce opex costs since the HW used for IT and networks services will be similar. On the other hand NFV gives the possibility to decrease used power resources by allowing dynamic scaling down the capacity when the network traffic is low. This also makes the green network in reality. In this thesis, we propose a dynamic load management framework for IMS networks using NFV. In the proposed framework, IMS functions are created within a single virtual machine (VM) instance and moved to the cloud environment hosted by a general purpose hardware system. Requests coming from the IMS clients are first processed by a load balancer module to efficiently distribute the incoming load over a pool of multiple IMS VMs. The decision of switching an IMS VM instance on or off is performed by the VM provisioning service using the resource utilization information periodically received from the IMS VMs. We utilized open source tools to implement the dynamic load management framework. The proof-of-concept experiments using a real testbed environment demonstrate that the proposed framework significantly increases the scalability of the IMS networks for the highly loaded traffic scenarios.

In thesis we have build a real testbed environment with open source tools. Open IMS is used for IMS functions like P-CSCF, I-CSCF, S-SCSF and HSS. Open SIPS is used as load balancer for balancing Sesssion Initiation Protocol (SIP) traffic towards IMS. IMS Bench SIPp used for SIP traffic generation. Zabbix monitoring tool used for monitoring IMS instances and starting new instances when needed because of high CPU load in active IMS nodes. The advantage of our testbed is showing real results since it is build completely by real products other than simulation tools like NS2 or Omnet+. These open source products installed on VMWare Workstation as VMs and necessary configurations done for these tools. Some configuration and source code change was also necessary at Open IMS, Open SIPS side for the integration of these tools. Scripting at Open SIPS load balancer and IMS Bench SIPp was necessary for the test scenarios like IMS calls , IMS registration and distributing the SIP traffic to IMS nodes effectively. To obtain the test results we run scenarios with LB and without

LB for several times and used average values in our graphics. Results obtained shows that putting load balancer is increasing CSRs (Call Success Rate) significantly when compared with the scenario without load balancer.

ŞEBEKE FONKSİYONLARI SANALLAŞTIRMA KULLANARAK IMS ŞEBEKELERİNDE DİNAMİK YÜK PAYLAŞIMI

ÖZET

LTE erişim teknolojisi, artan mobil trafiği karşılamak için standart hale geldi. LTE, tamamen IP tabanlı bir erişim teknolojisi kullanmaktadır. LTE şebekesi ana şebeke elemanı olarak IMS kullanmaktadır. IMS kendi içinde P-CSCF, I-CSCF , S-CSCF ve HSS elemanlarından oluşmaktadır. Bu elemanlar sayesinde IP servisleri verilebilmektedir. Öte yandan son yıllarda bulut şebekelerinde ve donanımlardaki gelişmeler ile şebeke fonksiyonlarının sanallaştırması mümkün olmaktadır. Şebeke fonksiyonlarındaki sanallaştırma ile telekom şebekelerinde akıllı telefonların ve internet cihazlarının artışının getirdiği veri trafiği nedeni ile oluşan yüksek yatırım maliyetlerinin önüne geçmek mümkün olmaktadır. Artan şebeke maliyetleri ve düşen gelirler nedeni ile NFV kullanımı gerekli hale gelmiştir. NFV ile sürekli gelişen kompleks şebeke servisleri de verilebilmektedir. IT altyapısında kullanılan donanımların , telekom servisleri için de kullanılması ile devreye alma ve işletme aşamaları da kolaylaşmaktadır. Öte yandan NFV dinamik olarak şebeke elemanlarının boyutlandırmasını da sağlamaktadır, böylece kapasite ihtiyacı oldukça yeni şebeke elemanları başlatılmakta, az trafik olduğu zamanlarda ise bu elemanlar kapatılmaktadır. Ani trafik ihtiyaçları sağlanmakta ve gereksiz durumlarda bazı şebeke elemanlarının kapatılması ile şebekelerin aynı zamanda daha az enerji tüketen yeşil şebekeler olması sağlannaktadır. Bu tezde, NFV kullanarak IMS şebekeleri için dinamik yük paylaşımını sağlayan bir yapı önerdik. Bu yapıda IMS şebeke fonksiyonları tek bir sanal makina olarak yaratılıp genel amaçlı bir donanım üzerinde çalışan bir bulut ortamına taşındı. IMS cihazlarından gelen talepler ilk olarak bir yük dağıtım elemanı tarafından işlenip IMS sanal makinalarına dağıtılmaktadır. Bir IMS sanal makinasını açma kapama kararı, VM provizyon servisi tarafından IMS sanal makinalarından gelen kullanım bilgisine göre yapılmaktadır. Dinamik yük paylaşım yapısını oluşturmak ve göstermek amacı ile tamamen açık kaynak kodlu yazılımlar kullandık. Bu açık kaynak kodlu yazılımları kullanarak gerçek bir test ortamı oluşturduk. Test ortamında yaptığımız deneyler dinamik yük paylaşımının yüksek trafik oluşturan durumlarda IMS şebekesindeki arama başarı oranını ölçülebilir şekilde arttırdığını gördük.

NFV konusu günümüz telekom dünyasında artan data trafiği ve IOT cihazları nedeni ile artık bir gereklilik haline geldi. Bu gerekliliği gören telekom dünyası, özellikle operatörler bu konuda çeşitli çalışma grupları oluşturdu. Buna örnek olarak ETSI bünyesinde oluşturulan ETSI NFV ISG çalışma grubunu verebiliriz. Bu gruba tüm dünyadan çeşitli telekom operatörleri ve üretici firmaları katılmaktadır. Bu grubun temel amacı NFV konusundaki mimari yapıyı oluşturmak, endüstriyi ürün geliştirme konusunda bilgilendirmek , standartları oluşturmak ve açık kaynak kodlu çalışmaları başlatmak ve teşvik etmek olarak söylenebilir. Açık kaynak kodlu NFV çalışmalarına “Open Platform for NFV” (OPNFV) örnek olarak verilebilir.

OPNFV grubunun amacı, açık kaynak kodlu entegre NFV fonksiyonlarını verebileceğimiz bir platform geliştirmektir. Bu gruba dahil olan üretici ve kullanıcılar ihtiyaçları tam olarak bildiği için tutarlılık, performans ve açık kaynak kodlu elemanlar arası iletişimin standart hale gelmesini sağlayabilecektir.

Çalışmamızda önerdiğimiz dinamik yük paylaşımının faydasını göstermek için tamamen açık kaynak kodlu yazılımlar kullandık. Bu açık kaynak kodlu yazılımlar ticari olarak da firmalar tarafından kullanılan yazılımlar olduğundan ortaya gerçek bir şebeke test ortamı çıktı. Tezimizde, NS2, OmNet gibi simülasyon yazılımları kullanmak yerine tamamen açık kaynak kodlu yazılım projeleri kullanmayı tercih etmemizin sebebi, ortaya daha gerçekçi bir test ortamı çıkarmaktı.IMS şebeke elemanı olarak Open IMS yazılımı kullandık, bu yazılım Fraunhofer Enstitüsü tarafından geliştirilen halen internetde kendi komünitesi olan bir yazılımdır. Burada IMS şebeke elemamının alt fonksiyonlarını verelebilecek P-CSCF, I-CSCF, S-CSCF ve HSS yazılımlarının tamamı mevcutdur. Bu yazılımları Open IMS internet sitesinden indirip derleyerek kullanmak mümkündür. Ek A'da bu kurulumun detayları geniş olarak anlatılmaktadır. Kurulumda P-CSCF, I-CSCF, S-CSCF ve HSS fonksiyonları aynı bilgisayar üzerine kurulduğu gibi farklı bilgisayarlara da kurulabilmektedir. Biz testleriminiz sırasında tüm IMS fonksiyonlarını aynı sanal makina üzerine kurarak gelen trafik miktarina gore IMS sanal makinalarının sayısını arttırmayı tercih ettik. Bu yöntem P-CSCF, I-CSCF, S-CSCF ve HSS fonksiyonları arasındaki sinyalleşme trafiğinin aynı makina üzerinde kalmasını sağlayıp, şebeke trafiğini de azalttığından tercih edilen bir yöntem olmaya başlamıştır.

Tüm test ortamı için birden fazla sanal makina kurulmasına izin verdiği için VMware Workstation'ı kullanmayı tercih ettik. Windows makinalar üzerine kurulan VMware Workstation ile sanallaştırma işlemini gerçekledik. VMware Workstation üzerine ihtiyacımız olan sanal makinaları kurarak, bunlara farklı ağ adresleri verdik. Böylece tamamen bulut ortamında IMS network fonksiyonlarını sanallaştırmış olduk. IMS sanal makinalarında farklı ağ adresleri kullanabilmek için konfigürasyon dosyalarında çeşitli değişiklikler yapmak gerekiyor. Buna ait detayları, Ek A'da paylaştık.

IMS terminallerinden gelen trafiği simüle etmek üzere endüstri de yaygın olarak kullanılan SIPp'nin bir versiyonu olan IMS Bench SIPp kullandık. Yine bu yazılım da tamamen açık kaynak kodludur. Ek B'de IMS Bench SIPp 'in nasıl yüklendiği ve nasıl konfigüre edildiğine dair tüm detaylar bulunmaktadır. IMS Bench SIPp, özellikle IMS testlerinde kullanmak üzere gerekli değişiklikler yapılmış bir versiyondur. Bu yazılım ile şebekeye kayıt olma, arama, şebekeye tekrar kayıt olma ve mesajlaşma gibi senaryolar test edilebilmektedir. Saniye başına düşen arama sayısı ve senaryonun ne kadar süreceği bilgisi test ayarlarında verilebilmektedir. Biz de yaptığımız testlerde bu ayarları kullanarak oluşturduğumuz yapının artan trafik ortamında performansa olan katkısını gösterdik. IMS Bench SIPp oluşturduğu log dosyalarından daha sonra senaryo ile ilgili tüm detay bilgilere ulaşmak mümkün olmaktadır. IMS Bench SIPp için farklı programlar çalıştırılmaktadır. Manager programı ile çalıştırılacak senaryolar yüklenmektedir ve daha sonra SIPp programlarından manager programına olan bağlantı sağlanarak senaryo başlatılmaktadır. Daha fazla trafik simüle edebilmek için birden fazla SIPp çalıştırmak mümkün olmaktadır. Aynı zamanda test edilecek sistemlerde cpum denilen bir program çalıştırılarak manager programı ile senkronizasyon işlemi sağlanmaktadır.

Dinamik yük paylaşımı yapılan yer ise Open SIPS yük paylaşım programıdır. Bu program da açık kaynak kodlu olarak erişilebilmekte ve kullanılabilmektedir.

Testlerimizde yük paylaşımını bu programın lb ve dispatcher denilen farklı modülleri ile yaptık. Dispatcher modülü farklı algoritmalara göre yük paylaşımı yapabilmektedir.

Testlerde kullanılmak üzere bazı kabuk programları yazılarak test işlemleri ve sonuçların alınması otomatize edildi. Farklı yük değerleri için dinamik yük paylaşımı yaparak ve yapmadan benzer testler farklı trafik modelleri ile çalıştırıldı. Artan yüklerde dinamik yük paylaşımının daha iyi çalıştığı test sonuçları ile gösterildi.

Ayrıca yine açık kaynak kodlu Zabbix gözlem programı ile IMS sanal makinalarının mevcut yüklerine bakarak yeni sanal makinaları başlatmak veya mevcut sanal makinaları kapatmak mümkün oldu.

Bu çalışma ile başlangıçta ulaşmak istediğimiz şebeke fonksiyonlarının sanallaştırmasını, IMS fonksiyonları için tamamen açık kaynak kodlu gerçek ürünler ile oluşturmuş olduk. Yine NFV için en kritik olan noktalardan biri olan dinamik yük paylaşımı ve gerekli durumlarda eleman sayısını arttırmayı yine tamamen açık kaynak kodlu ürünler ile gerçekleştirmiş olduk. Burada NFV'nin ve dinamik yük paylaşımının faydası gerçek bir ortamda gösterilmiş oldu.

Bundan sonraki çalışmalarda yine bu yapı üzerinde başka yük paylaşım algoritmaları denemek, LB'nin getirdiği ek maliyeti incelemek de mümkün olabilecektir. Aynı zamanda Openstack, Open NFV gibi yeni ürünlerinin oluşturduğumuz yapıya entegre edilmesi ve bu gerçek test ortamında denenmesi de mümkün olabilecektir.

1. INTRODUCTION

Cloud computing provides a convenient on-demand access mechanism to a shared pool of configurable resources including hardware or software. Enabled by increased data rates of wired and wireless network infrastructures, cloud computing has the potential to enable substantial costs reduction without compromising from the end users' quality of experience requirements due to increased capabilities provided by the cloud model.

The cost reduction is possible due to the dynamic configuration of the shared network resources according to the instantaneous processing requirements in fine granularity. This provides the basis for network function virtualization (NFV). Through NFV, multiple instances of network functions can be co-located in the same commercial off-the-shelf (COTS) hardware by the correspondingly configured virtual machines (VMs). This enables decoupling of hardware and software resources hence providing an easily scalable framework that can dynamically configured according to the access demands.

NFV offers a substantial cost saving means for network service providers. Traditionally, network service provides (NSPs) operate their systems through specialized hardwares, resulting in high capital expenditure for large-scale nation-wide deployments. Due to reduced average revenue per transmitted bit, NSPs have major concerns about sustaining a large-scale service deployment in a profitable operation. Furthermore, noting the ever increasing number of connected devices through the Internet of Things (IoT) paradigm, it is clear that the traditional methods of NSPs are not sustainable. Providing easy scalability using COTS hardware without any extensive modification to the existing infrastructure, the use of NFV is vital for the profitability of NSPs in the near future. Due to its high potential to impact the NSP infrastructures, ETSI recently initiated standardization efforts for NFV through an Industry Specification Group (ISG) [8].

NFV in a NSP can implemented through the IP multimedia subsystem (IMS) which provides a control plane for provisioning of multimedia services in the core network. Currently supporting Rel. 13 [1], IMS standardization activities started with the 3GPP Rel. 5, as part of the core network evolution from circuit-switching to packet-switching in the all-IP infrastructure [1]. IMS constitutes a collection of software functionalities for the core network of a telecommunication network. Although an IMS solution composed of hardware and software components can be obtained from a vendor, it is also possible to abstract the IMS functionalities from hardware through the use of NFV techniques. In this approach, the logical functionalities of IMS platform and the hardware are separated through the use of virtual machines (VMs) in the telecommunication network cloud. ETSI ISG on NFV is also addressing virtualized IMS infrastructures [8].

The rest of thesis organized as follows. In Chapter 1.2, we provide an overview of the existing literature. In Chapter 2, we introduce the main concepts used in the NFV. The proposed dynamic load-balancing framework and its implementation presented in Chater 2.6. and 3, respectively. The test model description and test results given in Chapter 4. We conclude the thesis in Chapter 5.

1.1 Purpose of Thesis

The purpose of this thesis is to show the benefits of NFV and how dynamic load balancing increasing overall performance and efficiency.

With the goal of creating a scalable, low cost infrastructure for NSPs through cloud computing, we propose a robust and scalable framework for integration of the virtual IMS for telecommunication network cloud as shown in Figure 1.1. This framework can also enable energy consumption reduction as the network resources are switched on by the load balancer according to their loads, avoiding any idle run-times. Through VMs and a load balancing control function, we determine the number of functional IMS components. The robustness stems from the fact the number of VMs system model that includes a load balancer for IMS networks using NFV.

1.2 Literature Review

Recently, there have been many research efforts towards migrating telecommunication networks over the cloud using the Network Function Virtualization concept [9]. For example, the authors in [9] presents a set of three software architectures for the virtualisation of IMS on top of a cloud-based infrastructure. In the merge-IMS architecture, four IMS functional entities (P-CSCF, S-CSCF, I-CSCF and HSS) are merged together within a single virtual machine called IMS VM and a pool of IMS VMs is used as an elastic infrastructure which can dynamically increase or decrease the number of active IMS VMs according to the system loading. The IMSLocator entity acts as a simple proxy for assigning the incoming IMS requests to a specific IMS VM instance during the registration phase, and locating the corresponding IMS VM instance during the call setup phase. Our proposed framework uses the merge-IMS architecture and a pool of IMS VMs similar to [9] and has additional functionalities such as dynamic load balancing and feedback based VM provisioning. In addition, our experiments performed using a real virtualized IMS testbed demonstrated the quantitative benefits of having additional IMS VMs.

In a recent study [20], the scalability of IMS networks is improved using the load balancing concept in which the HSS database is instantiated on multiple MySQL instances, where each instance is running on a physically seperated VM. This thesis study suggested that the scalability can be further improved by instantiating P-CSCF, I-CSCF and S-CSCF functions on multiple VMs and performing a load balancing among functionally equivalent CSCF instances. Our study presents a dynamic load management framework by creating multiple CSCF instances and improves the scalability of IMS networks by distributing the IMS signalling load on multiple IMS VM instances. In [4], [5] and [6] a similar load balancing concept is utilized but the purpose of using redundant CSCF instances is to maintain service continuity user transparently when one of the CSCF node fails for Self-Organizing IMS Networks.

In another study [10], the authors uses the fact that INVITE and BYE transactions can incur significantly different processing overheads: INVITE transactions are about 75% more expensive than BYE transactions. Three load balancing algorithms such as Call-Join-Shortest-Queue, Transaction-Join-Shortest-Queue, and Transaction-Least-Work-Left are designed by taking this fact into account. These algorithms used to make more intelligent load-balancing decisions to improve both the end-to-end response delay and the system scalability. Our proposed framework monitors the loading information on each IMS VM instance and makes a load balancing decision using the feedback information instead of relying on message processing overhead assumptions.

The author in [14] proposes a load management mechanism for IMS networks. The authors assume that application servers (ASs) are the bottleneck nodes and the resource utilization levels of ASs should be kept under control to gurantee a certain quality of service (QoS). The system load on each AS is estimated and this information is sent back to the load balancer which is located between S-CSCF and ASs. Using the shortest positioning strategy for the load balancing decision, messages coming from S-CSCF are allocated to the appropriate ASs. In our work, the load balancer is located between IMS clients and P-CSCFs and messages coming from IMS clients are first processed by the load balancer. Apart from the location of the load balancer, the dynamic provisioning of IMS VMs and the implementation of the proposed load balancer in the real testbed are two main differences of our work compared to [14].

The various aspects of the dynamic provisioning of VM resources in the NFV has been studied in [28]. For example, [21] proposes a dynamic and real-time load balancing mechanism to orchestrate the usage of cloud resources. The proposed system provides elastic resource scaling such that new nodes are created if more resources are needed and the operational nodes are shut down if their utilization levels are low. This system together with its load balancing feature ensures that the cloud resources are efficiently used and the customer satisfaction is provided with minimal cost. Our work is similar to this study in terms of the performance objective and the load balancing architecture; however, we apply the dynamic load balancing mechanism specifically for IMS networks and use a real testbed for the demonstration of the proposed technology.

The IP Multimedia eXchange (IMX) concept [13], which integrates P-CSCF, I-CSCF, S-CSCF and AS functions into the single border element node, is proposed to keep a significant portion of the SIP signaling within the intra-cluster domain. The resulting disributed architecture consists of equal general-purpose nodes which realize the IMS functions at the edge. However, due to unpredicted traffic patterns, flash crowd effects, value-added services such as televoting, or platform failures, it is likely that some of the distributed nodes can get overloaded while others are under-utilized. The IMX concept is used by our framework as well by instantiating P-CSCF, I-CSCF, S-

CSCF and HSS functions within a single IMS VM node and a dynamic load management mechanism is used to distribute the processing efforts among multiple IMS VMs within a distributed IMS architecture.

The author in [18] proposes de-registration based load-balancing for S-CSCF in requesting re-association between subscriber and S-CSCF pair, and therefore facilitate in re-directing consequent traffic from over-utilized S-CSCF to the others.

In study [12], a new S-CSCF assignment scheme which takes current S-CSCF load is proposed.

In another study [22], different S-CSCF selection schemes (uniform random allocation, round robin, shortest expected delay and least response time) are considered and diffences investigated.

ETSI NFV workgroup has been working on outlining the benefits, enablers and challenges for [8] and providing a common framework towards accelerating the development and deployment of interoperable solutions based on high volume industry standard servers [8].

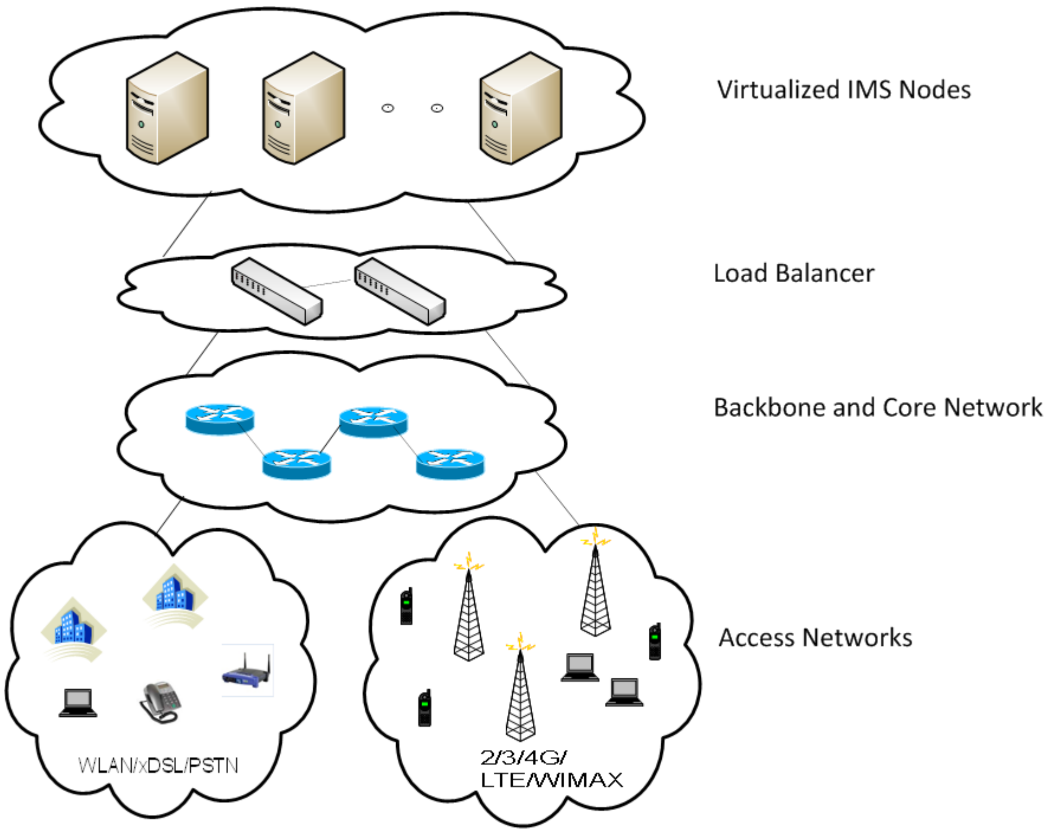


Figure 1.1 : Dynamic load management using Network Function Virtualization for IMS networks

1.3 Hypothesis

With the development of COTS HW and cloud technologies, it is possible to virtualize network functions. NFV will decrease the costs of capex and opex for network infrastructure, which is necessary with growing capacity need of increasing data usage.

Load balancing is necessary when multiple network nodes are used. We proposed a dynamic load balancing method, which increase the overall performance and efficiency. Our load balancing method is making the traffic distiribution according to the end nodes performance and start new nodes when it is necessary from traffic point of view. Therefore, it also improves the energy efficiency.

2. BASICS OF CLOUD COMPUTING FOR NETWORK SERVICE PROVIDERS

2.1 Cloud Computing and Network Function Virtualization

Cloud computing provides a convenient on-demand access mechanism to a shared pool of configurable resources including hardware or software. Enabled by increased data rates of wired and wireless network infrastructures, cloud computing has the potential to enable substantial costs reduction without compromising from the end users' quality of experience requirements due to increased capabilities provided by the cloud model.

The cost reduction is possible due to the dynamic configuration of the shared network resources according to the instantaneous processing requirements in fine granularity. This provides the basis for NFV. Through NFV, multiple instances of network functions can be co-located in the same COTS hardware by the correspondingly configured VMs. This enables decoupling of hardware and software resources hence providing an easily scalable framework that can dynamically configured according to the access demands.

Cloud computing can also act as an efficient cost saving tool for NSPs in service design, enabling them to allocate network resources in a dynamic fashion according to the user demand. Such a network has some differences from a conventional cloud computing infrastructure as carrier-grade availability, scalability and reliability has to be constantly supported. In this chapter, we describe the main components of cloud computing systems and highlight the differences between classical cloud computing systems and the cloud based all IP telecommunication networks. We then provide an overview of NFV. We also briefly describe IMS infrastructure along with virtual IMS that enables the use of NFV for telecommunication service delivery through VMs.

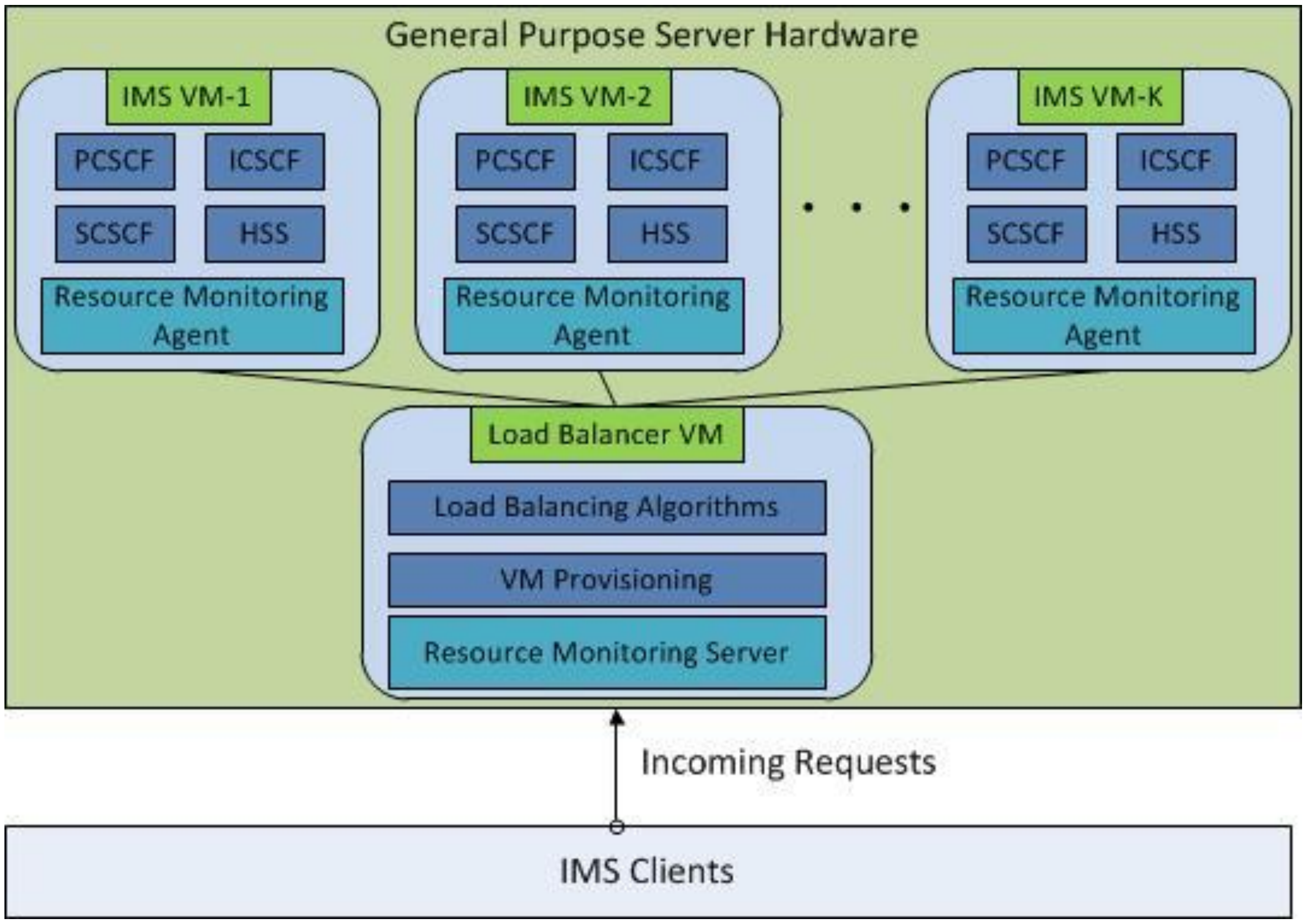

General Purpose Server Hardware
IMS VM-1
PCSCF
ICSCF
SCSCF
HSS
Resource Monitoring Agent
IMS VM-2
PCSCF
ICSCF
SCSCF
HSS
Resource Monitoring Agent
IMS VM-K
PCSCF
ICSCF
SCSCF
HSS
Resource Monitoring Agent
Load Balancer VM
Load Balancing Algorithms
VM Provisioning
Resource Monitoring Server
Incoming Requests
IMS Clients

Figure 2.1 : Increasing the number of IMS Nodes when network traffic increases.

2.2 Cloud based all IP Telecommunication Networks

Ever increasing data rates provided through both wired and wireless networks can connect end users with a diverse set of services through a network. With this seamless access capability, services can be offered from remote locations. Such a service deployment model can also be used to distribute a pool of computing resources to end users according to their temporal and spatial demands [27]. This enables the service provider to allocate resources with a fine granularity, offering substantial cost savings and increased flexibility. There are three commonly accepted service models for cloud computing; software as a service (SaaS), platform as a service (PaaS) and infrastructure as a service (IaaS). In SaaS, an enterprise application is delivered and managed as a service to simultaneously meet the requirements of a customer base. Of course customer demand can show temporal and spatial variations. In PAAS, a cluster of hardware and software can be rented from the service provider for a specific purpose. As an example, VMs can be rented by the end-users for their computational requirements. Finally, IaaS service model, end-users can provision fundamental computing resources to deploy and run software(s). Note that the IaaS model can accommodate both SaaS and PaaS models.

Based on the deployment strategy, cloud computing systems can be classified into three categories, public cloud, private cloud and hybrid cloud. Public clouds constitute the basis of the cloud ecosystem, with access from a diverse set of end-users. Public clouds are used within the users of a specific enterprise. Hybrid clouds are cloud systems where both public and private clouds are used that are connected possibly through a VPN tunnel.

As mentioned above, it is clear that NSPs can greatly benefit from the cloud computing architectures, to enable this however, strict controls of the network services are compulsory to enable carrier grade telecommunication services to the end user. From this perspective, the cloud computing systems for NSPs should be evaluated in a different category. Cloud based all IP telecommunication networks should support high availability probability, low latency and also should enable service level agreements (SLAs). To enable such a deployment, NFV can be used as a tool, as will be detailed in the following subchapter. For the deployment timeline, it is expected that the core network will transition to cloud based architecture at the preliminary stages. Later on, cloud concepts will possibly appear in the complete network, including the radio access networks.

We should also note the security aspects associated with cloud based all IP telecommunication networks. Obviously, it is vital for an NSP to address all security threats in the network. As the classical voice network evolve to all-IP networks, the vendor specific infrastructure is replaced with COTS equipment. Such a transition increases, increasing the attacker profile diversity with a relatively low technical barriers to entry through the use of sophisticated attack tools. In order to address such threats, several different approaches have been proposed such as mapping virtual network components onto physical nodes and links [19].

2.3 Network Function Virtualization

Traditionally, NSPs have a diverse set of hardware equipments possibly from distinct vendors to provide services to the end users. NFV decouples the network functionalities from the hardware. Hence, with the use of NFV, NSPs can consolidate the functionalities of these equipments by using COTS hardware, in a scalable and robust manner. To achieve this goal VMs are used in a dynamic manner according to the demand, controlled through a hypervisor that manages the available resources by the virtualization layer. Such an architecture is almost independent from the selected hardware resources, enabling the use of COTS hardware.

Another option for hardware selection is the so-called advanced telecommunications computing architecture (ATCA) compliant hardware. However, use of ATCA reduces implementation flexibility as may not be interchangeable among applications from different vendors [16]. Using NFV, the dependency of NSPs to a specific from hardware platform can be eliminated.

NFV can also used to increase the resiliency of the NSP services against any faults while maintaining satisfactory user experience both in the hardware and software context. In software context, the VMs can be switched on or off depending on the network status. When considering hardware faults, it is reasonable to deploy back-up devices due to the reasonable costs of COTS devices. Note that the same reasons also enable the scalability of the delivered services by the NSPs.

Frequently, NFV is jointly considered with software defined networking (SDN) concept [17]. SDN implies separation of the control plane from the data plane. It is worth noting that although SDN and NFV can benefit from joint usage, they are complementary technologies that can be utilized independently.

2.4 IMS Networks

Here, we provide the basic IMS components that are used to control voice calls. As one of the main differences of IMS with the circuit-switched infrastructure the communication protocol is changes to session initiation protocol (SIP) from integrated services digital network user part (ISUP) [1]. The main controller in the classical core network, the mobile switching centre is replaced by the x-call Session control function blocks. The three types are the proxy call session control function (P-CSCF), the serving call session control function (S-CSCF), and the interrogating call session control function (I-CSCF). P-CSCF directly communicates with the user equipment (UE) and functions as a SIP proxy server. The S-CSCF functions as the central node of the signalling plane executing session control as a key element in the IMS roaming methodology. It enables requests to be routed to the correct Serving Call State Control Function. As there may be several S-CSCFs either within a network, or if a roaming user requests access. The user information database is kept at the home subscriber server (HSS), replacing the conventional home location register database.The I-CSCF interrogates the HSS to obtain the address of the relevant S-CSCF to process the SIP initiation requests, mainly managing roaming functionalities.

Conventional telecommunication networks that are based on circuit switching have been transitioning to all-IP networks supporting circuit switching. The control plane functionalities can be performed through the IMS infrastructure. Such an infrastructure also provides the opportunity to provide services to the NSPs with the associated functionalities though NFV.

2.5 Dynamic Load Management Framework for IMS Networks

In this chapter, the proposed dynamic load management framework as depicted in Figure 2.1 is presented. The corresponding software implementation architecture is given in Figure.3.1. As explained in the previous chapters, with the recent developments in cloud technologies and COTS hardware, it becomes feasible to virtualize several network functions in Telco over Cloud Environment. In our framework, IMS functions such as P-CSCF, I-CSCF, S-CSCF and HSS are created within a single VM instance and moved to the cloud environment hosted by a general purpose hardware system. Moving these functions to the cloud environment is an example realization of Network Function Virtualization for telecommunication networks. To handle a highly variable and dynamic IMS signalling traffic, functionally equivalent multiple VM instances can be turned on and off on demand within a general purpose hardware system. For example, when the amount of traffic is significantly low during the night times, only IMS VM-1 may handle all the incoming requests and all the other IMS VM instances may be turned off to save the energy consumption. As the amount of traffic keeps increasing during the day times, new IMS VM instances can be instantiated on demand.

An explosive growth in the number of users necessitates higher capacities in the telecommunication network infrastructure and emerging real-time services such as VoIP and Video on Demand have stringent delay requirements. To meet the unprecedented growth in traffic demand and support the QoS requirements of real-time services, we propose to use a dynamic load management mechanism using a pool of IMS nodes instantiated as independent virtual machines. In this framework, requests coming from the IMS clients are first processed by a load balancer VM to efficiently distribute the incoming load over multiple IMS VMs. Load balancing decisions in typical Open SIPs implementation are performed by utilizing the following parameters: hash over callid, hash over from uri, hash over to uri, hash over request-uri, round-robin, hash over authorization-username, and random.

Load balancing algorithms mentioned above do not consider the amount of load on each active IMS VM instance which provides practically important information for efficient load distribution. In our proposed framework, resource monitoring agent running in each IMS VM instance monitors the resource utilizations such as CPU load, free memory, swap space, and energy consumption, and periodically reports this information to the resource monitoring server running on Load Balancer VM. The decision of instantiating a new IMS VM instance or turning the active IMS VM off is performed by the VM Provisioning process using this feedback information regarding the resource utilizations. Dynamically changing the number of active IMS VM instances will significantly reduce the energy consumption and opex of the network. This valuable feedback information can also be used to implement more intelligent load balancing algorithms to decide which IMS VM instance to send a new incoming request. We conjecture that distribution of the active load according to the near-past and current resource utilizations of IMS VMs is expected to increase the overall network performance and the end-to-end service quality of IMS clients.

Figure 2.2. shows the pseudo code of an example dynamic load management algorithm. We assume that there are k active VMs and *LAVM* represents the list of currently operational VMs. Resource Utilization Percentage (*RUP*) represents the level of resource utilization for each VM and is a function of VM status, CPU usage, free memory, swap space, and energy consumption for an underlying VM. If a VM is not operational, its VM status is off and hence its *RUP* value is 0%. *LRUP* represents the list of RUPs for active VMs (e.g., RUP-1, RUP-2, ..., RUP-*k*). *RUP_min*, *RUP_min2*, and *RUP_avg* corresponds to the smallest, the second smallest, and the average RUP values for LRUP, respectively, and *RUP_thr-min* and *RUP_thr-max* are the minimum and maximum RUP thresholds used for shutting down an operational

VM if the average system load is low and instantiating a new VM if the average system load is high, respectively.

The algorithm in Figure 2.2. is executed for every incoming new request from IMS clients. First, *RUP_min* and *RUP_min2* are found in *LRUP*, their corresponding VMs are recorded and *RUP_avg* is calculated. If *RUP_avg* is lower than *RUP_thr-min*, it indicates that the average resource utilization of *LAVM* is lower than the administratively specified minimum RUP value and one of the active VMs can be shutdown to save the energy consumption. In this case, an incoming new request is sent to the VM with the second smallest RUP value in *LRUP* since the VM with smallest RUP value is a potential candidate for the shut down operation. Otherwise, an incoming new request is sent to the VM with the smallest RUP value in *LRUP*. If *RUP_avg* is higher than *RUP_thr-max*, it indicates that the average resource utilization of *LAVM* is higher than the administratively specified maximum RUP value and a new VM needs to be instantiated to maintain the administratively specified quality of service. Note that this is an example algorithm which utilizes the RUP values of the active VMs for the load management purpose and one can develop more intelligent load management algorithms using our proposed framework to improve the overall system performance.

```
LAVM: List of Active VMs {VM-1, VM-2, ..., VM-k
RUP: Resource Utilization Percentage
LRUP: List of RUPs for active VMs {RUP-1, RUP-2, ..., RUP-k
RUP_min: the smallest RUP value in LRUP
RUP_min2: the second smallest RUP value in LRUP
RUP_avg: the average RUP value for LRUP
RUP_thr-min: minimum RUP threshold for shutting down a VM
RUP_thr-max: maximum RUP threshold for instantiating a new VM

The following algorithm is executed upon receipt of an incoming new
request to select its corresponding VM and update LAVM
Update RUP values in  LRUP
Find RUP_min and its VM index i
Find RUP_min2 and its VM index j
Calculate RUP_avg
if (RUP_avg<RUP_thr-min){
send incoming new request to VM-j
         if no active call in VM-i{
            shut down VM-i
            remove VM-i from LAVM
         }
}
else{
        send incoming new request to VM-i
        if (RUP_avg>RUP_thr-max){
               instantiate a new VM and add it to LAVM
        }
}
```

Figure 2.2 : The Pseudo code of a sample dynamic load management algorithm using resource utilization information received from the active VMs.

3. IMPLEMENTATION

In this Chapter, we will first describe the testbed environment used for the realization of the proposed dynamic load management mechanism as depicted in Figure 3.1.

We will explain the details of the tools used for implementation and then give the details of our realization of the test environment.

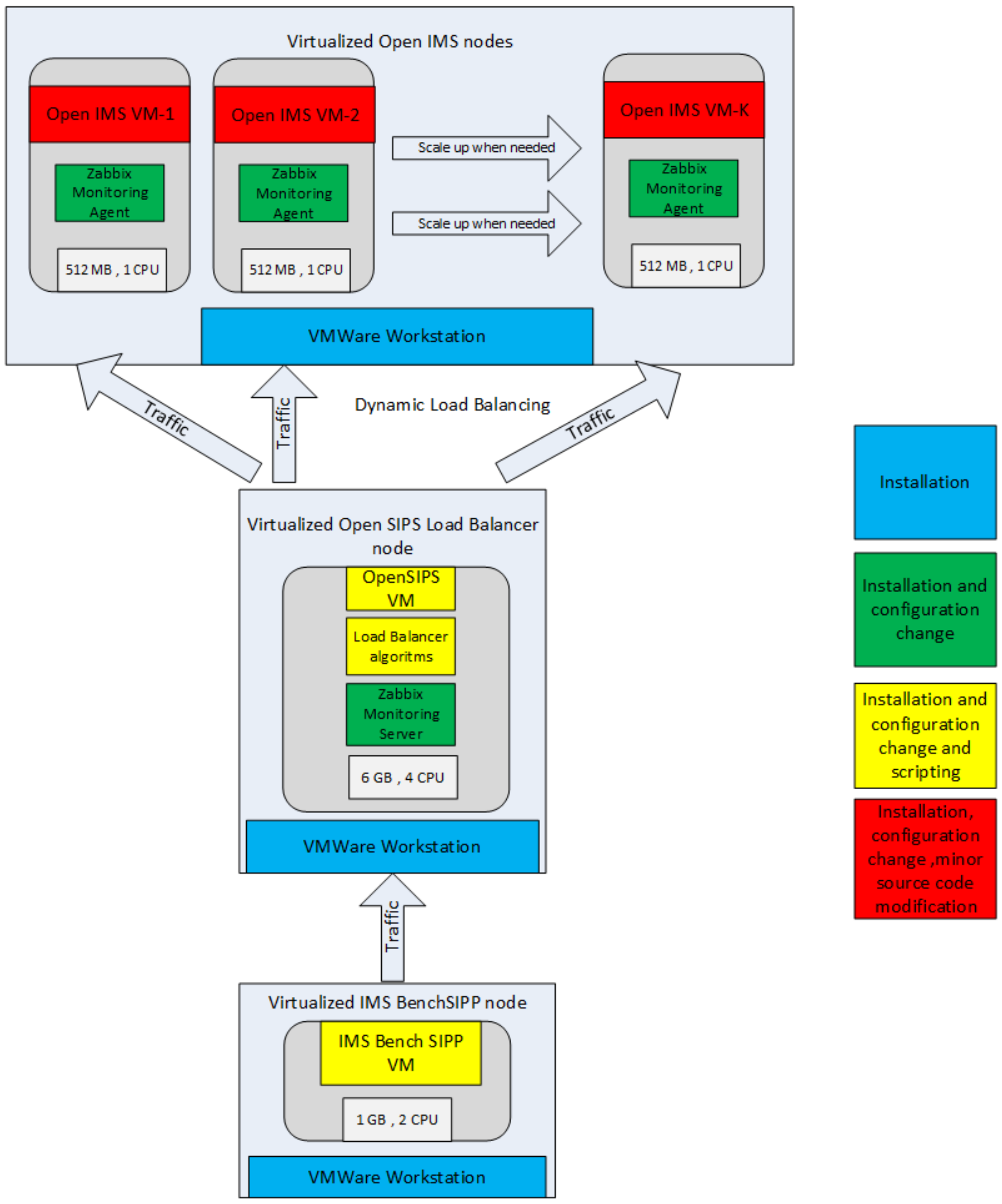


Figure 3.1 : Software implementation of testbed.

3.1 OpenIMS Core

For our implementation of IMS nodes, we have decided to use Open IMS, which is an open source implementation used in industry and academic researchs. Source code of Open IMS components are reachable and it is possible to make some modifications, which needed when used for specific purposes. We have also made some changes to make the necessary integration between Open SIPS load balancer and Open IMS.

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 [1]. OpenIMS Core was developed by Fraunhofer Institute FOKUS [26].

The high level architecture of OpenIMS Core is shown in Figure 3.2. The OpenIMS Core's CSCFs are based on a SIP Express Router (SER) [33]. 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 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. We used all of these IMS functions in one IMS VM running in VMWare workstation.

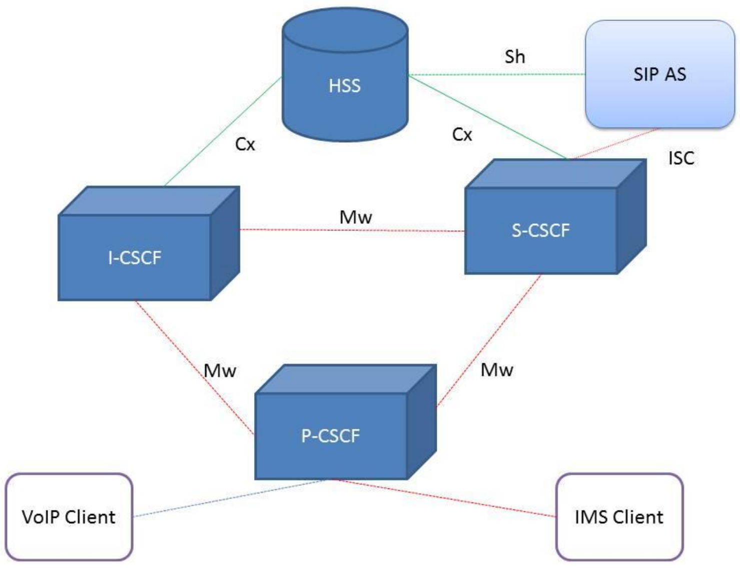
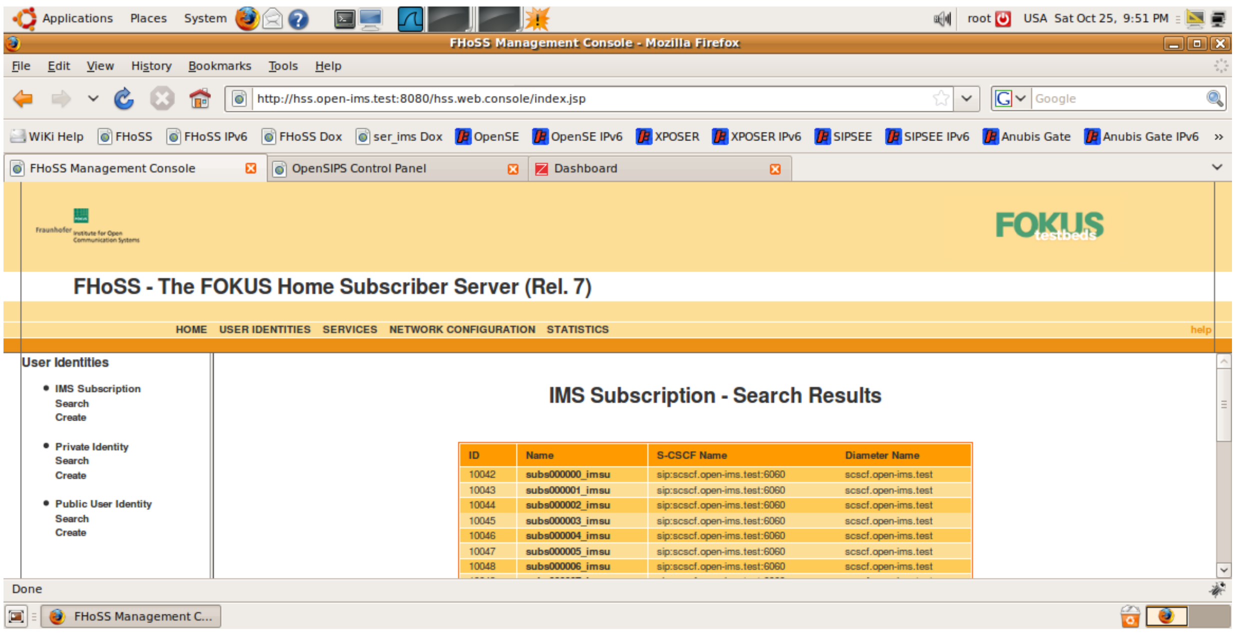


Figure 3.2 : Open IMS Architecture.

Following figure shows the screen for Open IMS HSS console, which gives details of the created subscribers in Open IMS HSS nodes.



FHoSS - The FOKUS Home Subscriber Server (Rel. 7)

IMS Subscription - Search Results

ID	Name	S-CSCF Name	Diameter Name
10042	subs000000_imsu	sip:scscf.open-ims.test:6060	scscf.open-ims.test
10043	subs000001_imsu	sip:scscf.open-ims.test:6060	scscf.open-ims.test
10044	subs000002_imsu	sip:scscf.open-ims.test:6060	scscf.open-ims.test
10045	subs000003_imsu	sip:scscf.open-ims.test:6060	scscf.open-ims.test
10046	subs000004_imsu	sip:scscf.open-ims.test:6060	scscf.open-ims.test
10047	subs000005_imsu	sip:scscf.open-ims.test:6060	scscf.open-ims.test
10048	subs000006_imsu	sip:scscf.open-ims.test:6060	scscf.open-ims.test

Figure 3.3 : Open IMS HSS Management Console

We have used following script for creating subcribers in Open IMS HSS.We have created total 10000 subscribers in all of the Open IMS HSS nodes. These subscribers later used for testing from IMS Bench. Registration and call scenarios created for these defined subcribers.

```
#!/bin/bash
for i in $(seq -f"%07g"  0 10000)
do
      myUname="Subs$i"
      printf "\n" | `./add-imscore-user_newdb.sh -u $myUname -
p abcdefgh -a\n`
done
```

Following is the sql script for creating subcribers in Open IMS Mysql database:

insert into hss_db.imsu(name) values ('subs000000_imsu');

insert into hss_db.impi(
identity,
id_imsu,
k,
auth_scheme,
default_auth_scheme,
amf,
op)

```
values( 'subs000000@open-ims.test',
        (select id from hss_db.imsu where hss_db.imsu.name='subs000000_imsu'),
        'abcdefgh',
        127,
        1,
        '\0\0',
        '\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0');

insert into hss_db.impu(identity,id_sp) values ('sip:subs000000@open-ims.test',
(select id from hss_db.sp order by id limit 1));
update hss_db.impu set id_implicit_set=id where
hss_db.impu.identity='sip:subs000000@open-ims.test';

insert into hss_db.impi_impu(id_impi,id_impu) values ((select id from hss_db.impi
where hss_db.impi.identity='subs000000@open-ims.test'), (select id from
hss_db.impu where hss_db.impu.identity='sip:subs000000@open-ims.test'));

insert into hss_db.impu_visited_network(id_impu, id_visited_network) values((select
id from hss_db.impu where hss_db.impu.identity='sip:subs000000@open-ims.test'),
(select id from hss_db.visited_network where hss_db.visited_network.identity='open-
ims.test'));
```

3.2 IMS Bench SIPp

IMS Bench SIPp is an open source implementation of a test system designed by Intel Corparation in accordance with the IMS/NGN Performance Benchmark specification ETSI TS 186 008 . 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.

- IMS call,
- IMS messaging,
- IMS Registration,
- IMS De-registration
- IMS Re-registration.

We made some specific changes in the scenarios to simulate our environment like registration to all available Open IMS nodes, interval between calls, call per second , call distribution like constant or poisson, ring time, hold time and message flow.

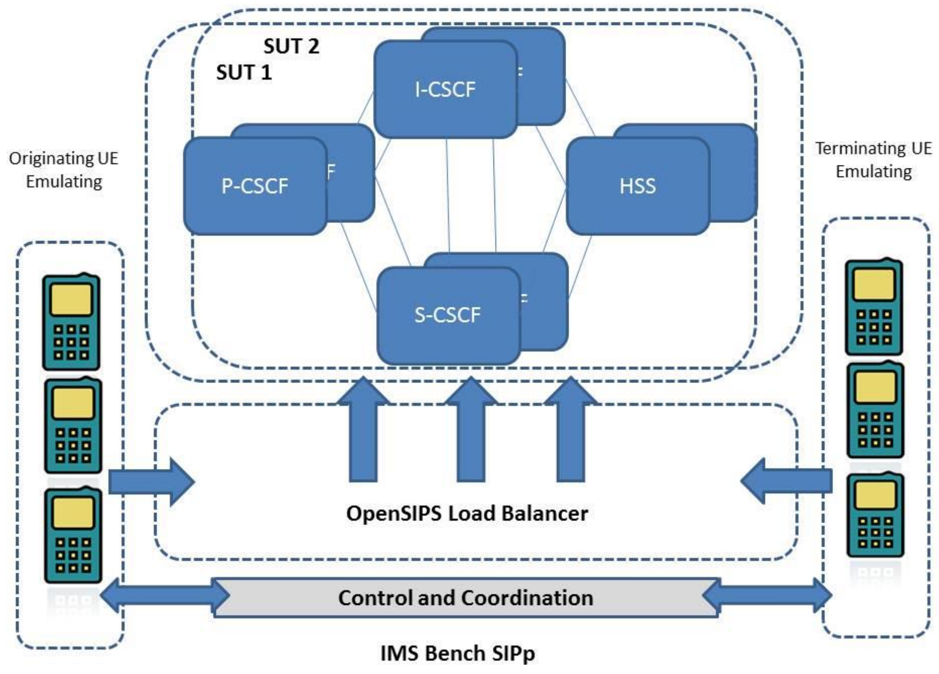


Figure 3.4 : IMS Bench Architecture.

Figure 3.4 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, it 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 [29].

Figure 3.5 shows the testing logic used in our scenario. Manager and SIPp instances running in one VM called IMS Bench. Open SIPS LB running in another VM at VMWare workstation installed on another PC. System Under Test (SUT) which are Open IMS nodes running on VMs at VMware workstation.

First related users, which used for future traffic scenarios, created at Open IMS HSS nodes by related scripts, which have explained detailed in Open IMS chapter. In our test scenarios, we have created all of the subscribers in all of the Open IMS HSS nodes.

For our testing, we have only used Registration (ims_reg.xml) and IMS Call (ims_uac.xml and ims_uas.xml) scenarios.

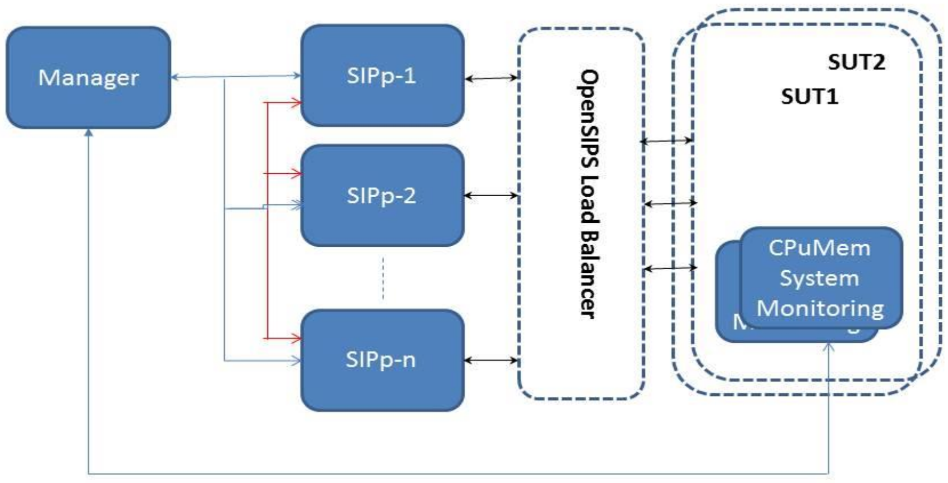


Figure 3.5 : Testing with IMS Bench SIPp

It is possible to define CPS values and other call related parameters in IMS bench scenarios.

Starting IMS Bench SIPp manager at the node, which IMS Bench SIPp installed, can done with the following command.

root@ubuntu:/opt/ims_bench# ./manager -f manager.xml

Test system registration commands for IMS Bench SIPp , which is registration of SIPp instances, are done with following command:

root@ubuntu:/opt/ims_bench/ims_bench_0# /opt/ims_bench/sipp -id 1 -sn uas -i 192.168.2.11 -user_inf ./ims_users_2.inf -rmctrl 192.168.2.11:5000 192.168.2.141:4060 -trace_err -trace_cpumem -trace_scen -trace_retrans -trace_stat -trace_logs -trace_msg

Below is part of the user_inf file, which includes the subcriber list for which our test scenario is running.These subscribers first created in Open IMS HSS nodes with the scripts explained in detail at Open IMS chapter.

root@ubuntu:/opt/ims_bench/ims_bench_0# more ims_users_2.inf

0;subs000000;open-ims.test;subs000000;open-ims.test;abcdefgh

0;subs000001;open-ims.test;subs000001;open-ims.test;abcdefgh

0;subs000002;open-ims.test;subs000002;open-ims.test;abcdefgh

0;subs000003;open-ims.test;subs000003;open-ims.test;abcdefgh

0;subs000004;open-ims.test;subs000004;open-ims.test;abcdefgh

0;subs000005;open-ims.test;subs000005;open-ims.test;abcdefgh

For the Load Balancer we have used Open SIPS Load Balancer [30]. In Open SIPS load balancer there are different modules for load balancing purposes like Dispatcher Module and Load Balancing Module. Dispatcher Module can run according to different algorithms.

Starting cpum on the nodes, which are under test, Open IMS nodes in our case is as following:

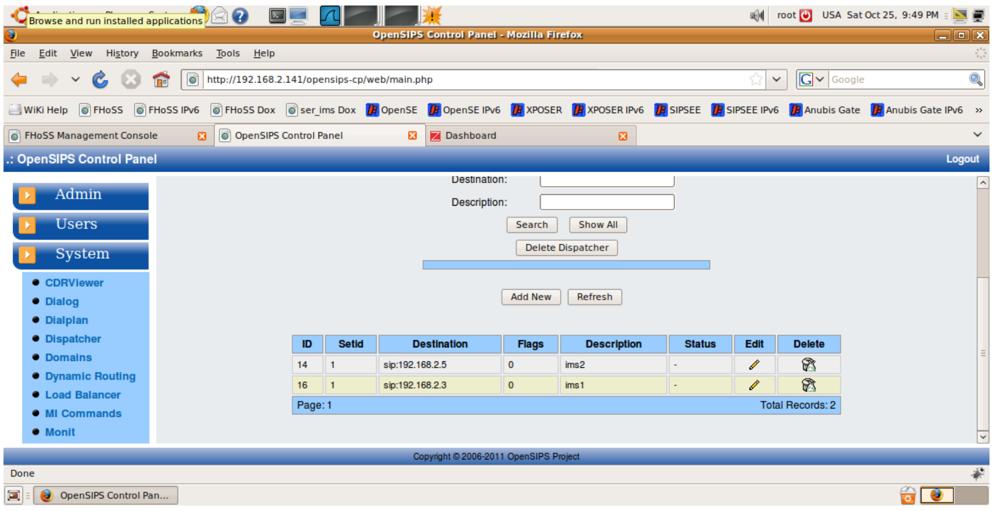
root@ubuntu-vm:/opt/ims_bench# ./cpum 192.168.2.11:5000

This command starts the connection from Open IMS nodes to the IMS Bench SIPp manager and sends the measurement information from SUTs during run of the scenario.

In our implementation, we have used Open IMS to simulate IMS nodes. We run several IMS nodes as VM instances in virtualized environment.

Open SIPS control script, which executed for every SIP message arrived to load balancer, modified according to our scenarios. Details of the Open SIPS control script given in Appendix C. Routing in the load balancer is working according to modified Open SIPS control script. Open SIPS control scripts used for routing, loading related Open SIPS modules, customizing the IP addresses and port numbers, which Open SIPS will bind and listen. Open SIPS control scripts also writes log messages according to the configuration. Routing done according to the received address, message type and routing logic. Modified part of the control script has shown in Appendix C with red color.

From OpenSIPS control panel it is necessary to add Open IMS nodes, which load balancing, will done. We have defined Open IMS nodes as seen in Figure 3.6.



ID	Setid	Destination	Flags	Description	Status	Edit	Delete
14	1	sip:192.168.2.5	0	ims2	-	✎	🗑
16	1	sip:192.168.2.3	0	ims1	-	✎	🗑

Page: 1 Total Records: 2

Figure 3.6 : Open SIPS control panel

3.3 Zabbix

Zabbix is an enterprise-class open source monitoring solution, which supports the following advanced monitoring features [32]:

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 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, for each notable event. Remote commands can sent to Zabbix agents to reduce operating cost by avoiding downtime and increasing scalability when needed.

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

3.3.1 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 monitored. The functionality of the Zabbix server 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. 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.

In our implementation, we have installed Zabbix server to Open SIPS node to monitor the agents and take necessary actions at load balancer side. Details of the Zabbix server installation given in Appendix C.

3.3.2 Zabbix Agent

The Zabbix agent is a process that deployed on a monitored host to actively monitor local resources and applications. The Zabbix agent locally gathers system information, and 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 configured by selecting the respective “Zabbix agent” or “Zabbix agent (active)” item types.

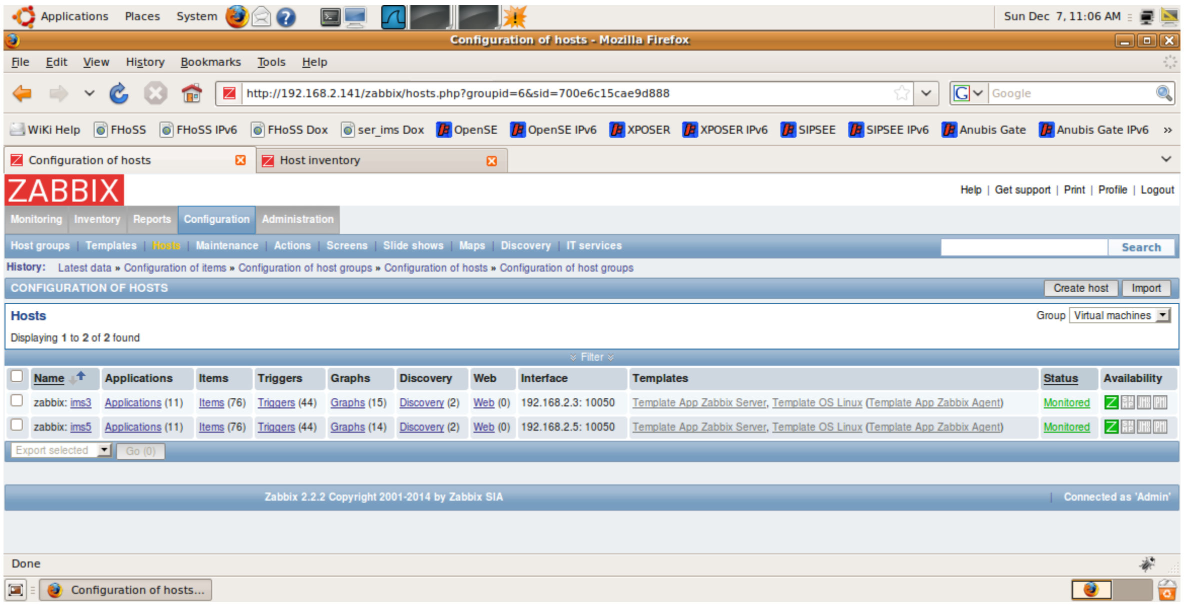
In our implementation, we have installed Zabbix agents to all of the Open IMS nodes for monitoring purposes. Details of the Zabbix agent installation given in Appendix C.

We have also defined IMS nodes in Zabbix management console to monitor, create triggers and actions.

Below are the steps for creating a trigger and action for starting second Open IMS node when the CPU idle time decreases below 10 percentage at first Open IMS node from Zabbix Console.

First, Zabbix agents installed to Open IMS nodes as explained detailed in Appendix C. After installation, Zabbix agent to Open IMS nodes, these nodes also created from Zabbix console. Monitored Open IMS nodes seen from Zabbix console dashboard as in Figure 3.7. If the connection between Zabbix server and Zabbix agent is established without a problem Z sign in dashboard for the related node turn from red to green.

Green sign shows that Zabbix server is able to collect measurement information from related agents.



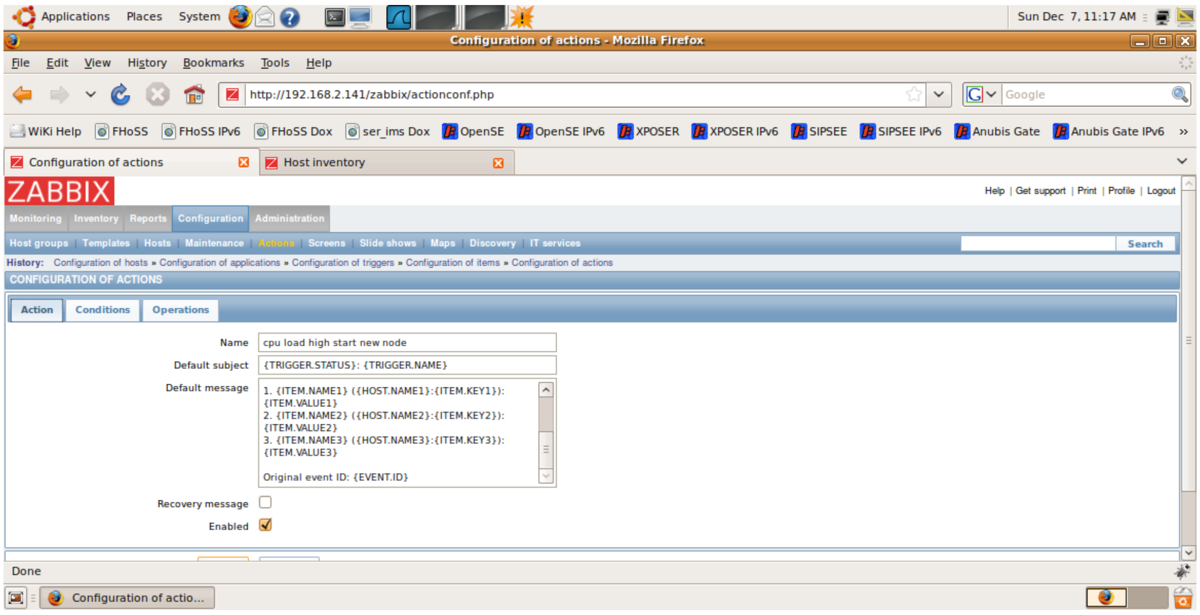
Hosts

Displaying 1 to 2 of 2 found

Name	Applications	Items	Triggers	Graphs	Discovery	Web	Interface	Templates	Status	Availability
zabbix: ims3	Applications (11)	Items (76)	Triggers (44)	Graphs (15)	Discovery (2)	Web (0)	192.168.2.3: 10050	Template App Zabbix Server, Template OS Linux (Template App Zabbix Agent)	Monitored	🟩
zabbix: ims5	Applications (11)	Items (76)	Triggers (44)	Graphs (14)	Discovery (2)	Web (0)	192.168.2.5: 10050	Template App Zabbix Server, Template OS Linux (Template App Zabbix Agent)	Monitored	🟩

Figure 3.7 : Monitoring Open IMS nodes with Zabbix Console

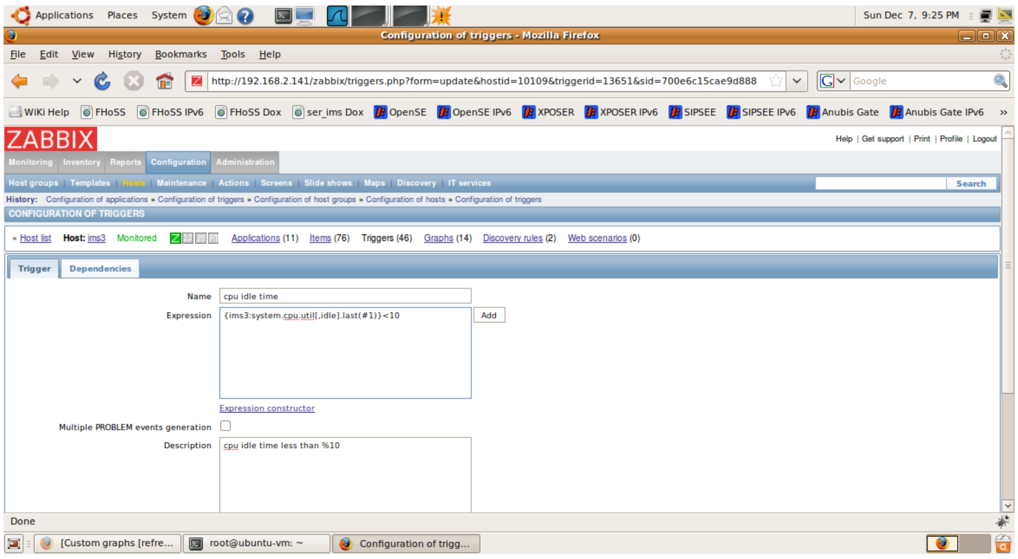
In our implementation, our aim is to get CPU load information from Open IMS nodes and if needed according to our algorithm start new Open IMS nodes to handle increasing traffic or when traffic is decreased shutdown the Open IMS nodes according to the traffic amount. Therefore, we have created an action from Zabbix dashboard as in Figure 3.8.



Name	cpu load high start new node
Default subject	{TRIGGER.STATUS}: {TRIGGER.NAME}
Default message	1. {ITEM.NAME1} ({HOST.NAME1}:{ITEM.KEY1}): {ITEM.VALUE1} 2. {ITEM.NAME2} ({HOST.NAME2}:{ITEM.KEY2}): {ITEM.VALUE2} 3. {ITEM.NAME3} ({HOST.NAME3}:{ITEM.KEY3}): {ITEM.VALUE3} Original event ID: {EVENT.ID}
Recovery message	☐
Enabled	✓

Figure 3.8 : Configuration of action for high CPU load at first node

In our implementation, we created also a trigger to start an action when CPU idle time falls below 10 percentage. Configuration details of trigger creation from Zabbix dashboard shown in Figure 3.9.


Configuration of triggers - Mozilla Firefox

http://192.168.2.141/zabbix/triggers.php?form=update&hostid=10109&triggerid=13651&sid=700e6c15cae9d888

History: Configuration of applications » Configuration of triggers » Configuration of host groups » Configuration of hosts » Configuration of triggers

CONFIGURATION OF TRIGGERS

« Host list Host: ims3 Monitored Applications (11) Items (76) Triggers (46) Graphs (14) Discovery rules (2) Web scenarios (0)

Name: cpu idle time

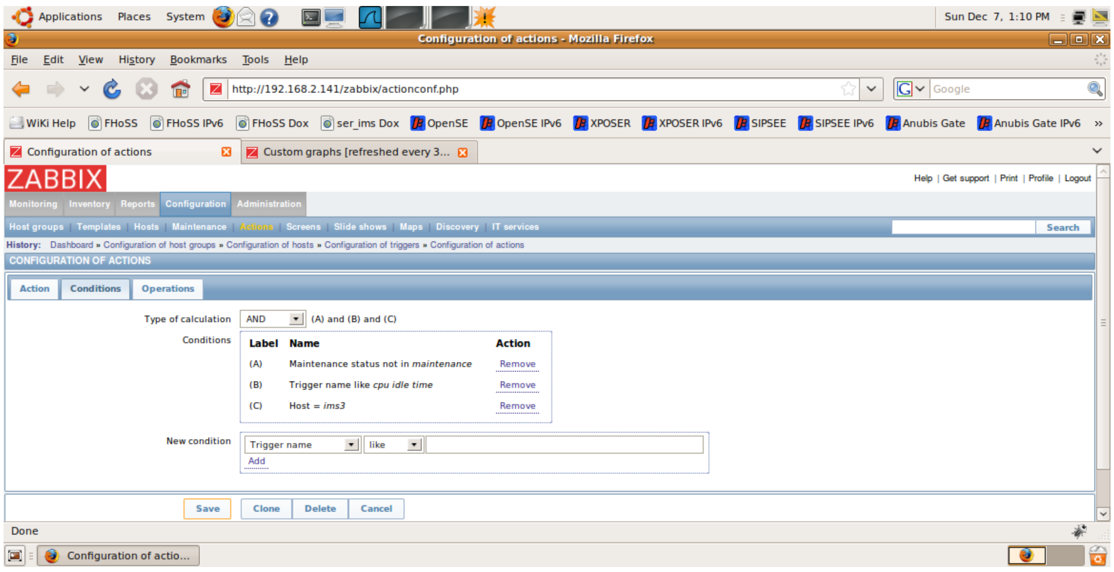
Expression: {ims3:system.cpu.util[,idle].last(#1)}<10

Multiple PROBLEM events generation: ☐

Description: cpu idle time less than %10

Figure 3.9 : Configuration of trigger for CPU idle time < 10

Actions are also mapped some conditions like maintenance mode, trigger name and host name. In our action condition we have defined host name equals first IMS node, status is not maintenance and trigger name equals “cpu idle time”. Configuration details of action conditions from Zabbix dashboard shown in Figure 3.10.


Configuration of actions - Mozilla Firefox

http://192.168.2.141/zabbix/actionconf.php

History: Dashboard » Configuration of host groups » Configuration of hosts » Configuration of triggers » Configuration of actions

CONFIGURATION OF ACTIONS

Type of calculation: AND (A) and (B) and (C)

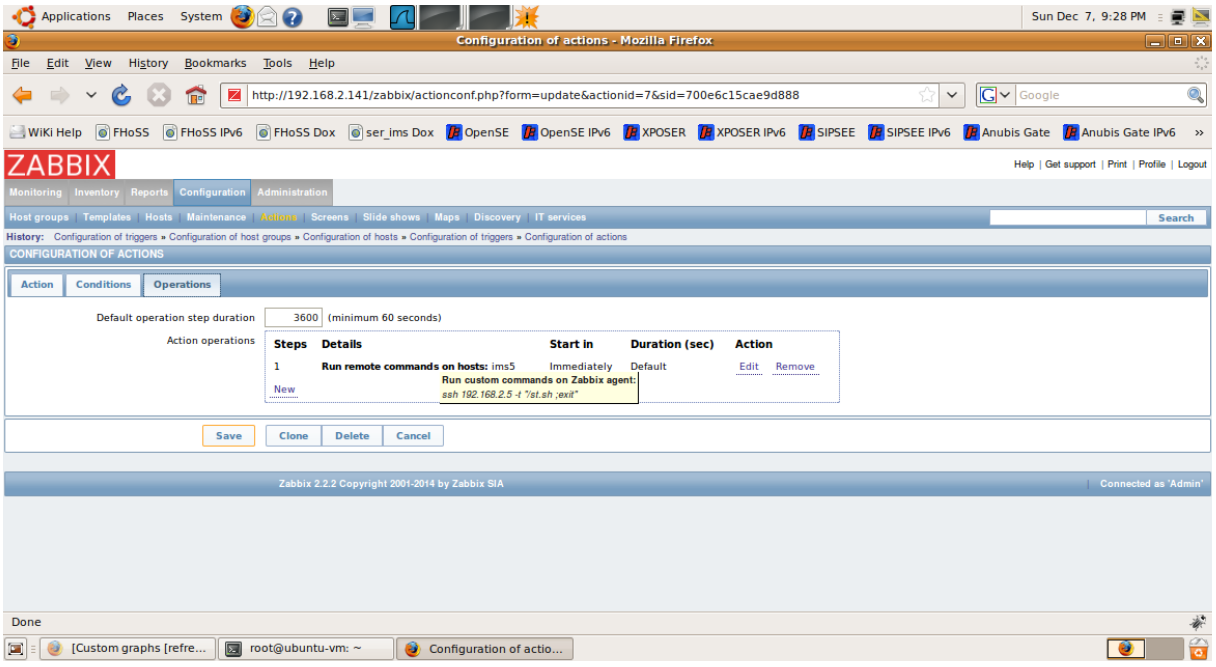
Conditions

Label	Name	Action
(A)	Maintenance status not in *maintenance*	Remove
(B)	Trigger name like *cpu idle time*	Remove
(C)	Host = *ims3*	Remove

New condition: Trigger name like

Figure 3.10 : Configuration of trigger conditions for CPU idle time

It is possible to define several actions when trigger occurs. In our implementation we have defined custom command to start new Open IMS node when trigger occurs due to decrease in CPU idle time in active Open IMS node. Configuration details of action operations at Zabbix dashboard shown in Figure 3.11



http://192.168.2.141/zabbix/actionconf.php?form=update&actionid=7&sid=700e6c15cae9d888

History: Configuration of triggers » Configuration of host groups » Configuration of hosts » Configuration of triggers » Configuration of actions

CONFIGURATION OF ACTIONS

Default operation step duration: 3600 (minimum 60 seconds)

Action operations:

Steps	Details	Start in	Duration (sec)	Action
1	**Run remote commands on hosts:** ims5	Immediately	Default	Edit Remove

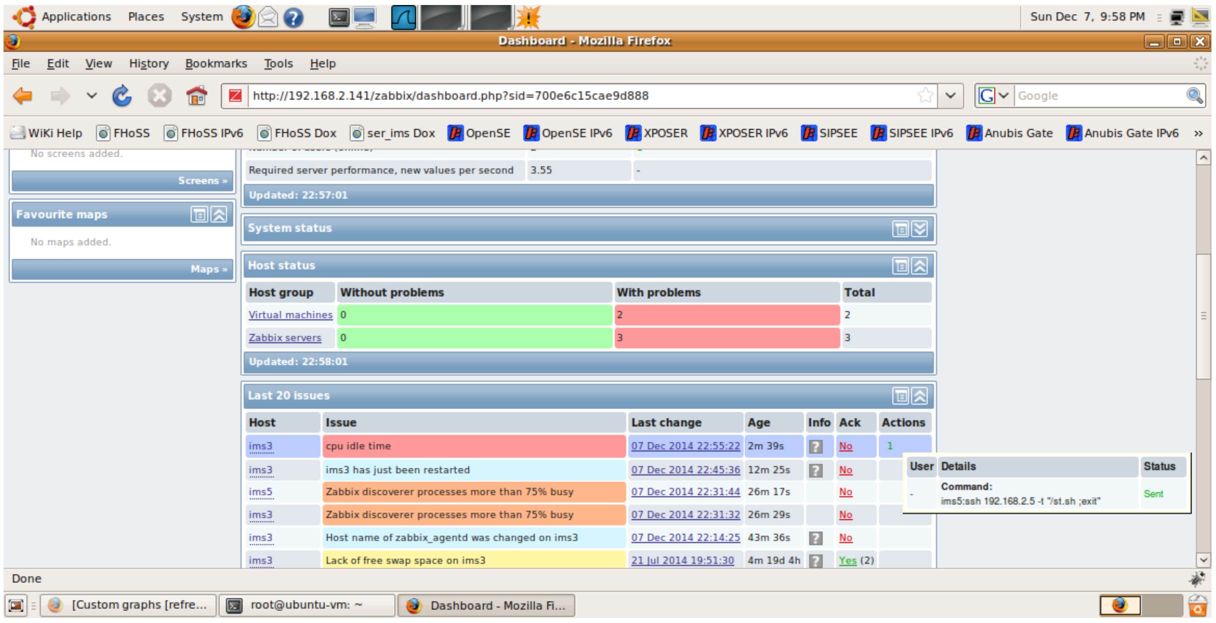
Run custom commands on Zabbix agent:

```
ssh 192.168.2.5 -t "/st.sh ;exit"
```

Zabbix 2.2.2 Copyright 2001-2014 by Zabbix SIA

Figure 3.11 : Configuration of trigger actions for CPU idle time

Figure 3.12 shows occuring “cpu idle time” trigger and starting action and operation which runs custom command to start new Open IMS node due to decreased CPU idle time in active Open IMS node.



http://192.168.2.141/zabbix/dashboard.php?sid=700e6c15cae9d888

Required server performance, new values per second	3.55	-

Updated: 22:57:01

Host status

Host group	Without problems	With problems	Total
Virtual machines	0	2	2
Zabbix servers	0	3	3

Updated: 22:58:01

Last 20 issues

Host	Issue	Last change	Age	Info	Ack	Actions
ims3	cpu idle time	07 Dec 2014 22:55:22	2m 39s	?	No	1
ims3	ims3 has just been restarted	07 Dec 2014 22:45:36	12m 25s	?	No	
ims5	Zabbix discoverer processes more than 75% busy	07 Dec 2014 22:31:44	26m 17s		No	
ims3	Zabbix discoverer processes more than 75% busy	07 Dec 2014 22:31:32	26m 29s		No	
ims3	Host name of zabbix_agentd was changed on ims3	07 Dec 2014 22:14:25	43m 36s	?	No	
ims3	Lack of free swap space on ims3	21 Jul 2014 19:51:30	4m 19d 4h	?	Yes (2)	

User	Details	Status
-	Command: ims5:ssh 192.168.2.5 -t "/st.sh ;exit"	Sent

Figure 3.12 : CPU idle time trigger occurs and new node started

3.4 Realization of test environment

As mentioned already, for IMS functions we utilized an open source Open IMS Core project [26], which developed by Fraunhofer Institute FOKUS. In this project, Session Control Functions (CSCFs) such as P-CSCF, S-CSCF, and I-CSCF and Home Subscriber Server (HSS) developed according to the 3GPP standard. Each CSCF component acts as an independent node in the standard IMS architecture. However, in our testbed, IMS functions such as P-CSCF, I-CSCF, S-CSCF and HSS are created within a single virtual machine (VM) instance and a pool of IMS VM instances are created on a general purpose computer with 6 GB RAM and 2 cores, each with 2.5 GHz CPU. Each VM instance configured to run on a single core CPU and 512 MB RAM. Details of the testbed physical implementation given in Figure 3.13. For our tests we have used one Ubuntu OS installed Dell PC, which is running Opensips LB VM in VMWare workstation, and one Windows8 installed Lenovo PC, which is running Open IMS VMs and IMS Bench SIPp VM in VMWare workstation.

Network configuration of VMs need to be done in bridged mode to realize the communication between VMs installed different physical servers.This setting done from Virtual machine settings-> Network Adapter screen at VMware workstation.

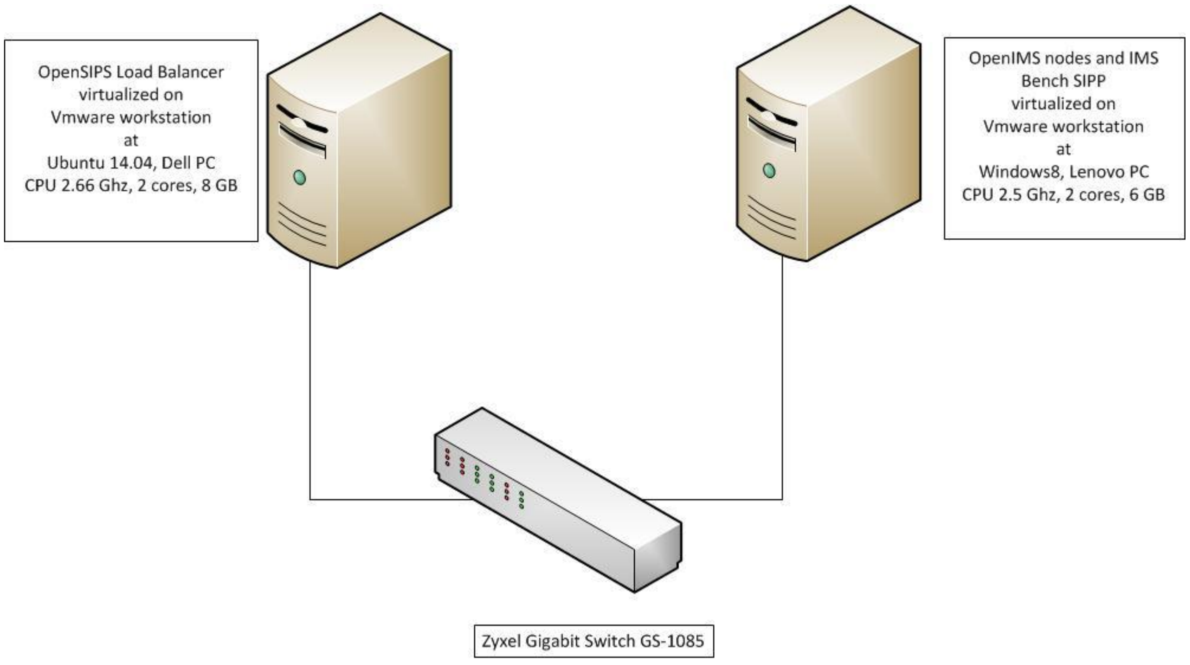


Figure 3.13 : Physical implementation of testbed

As the LB, we utilized an open source Open SIPs project [30], which contains Dispatcher and Load Balancing modules for the load balancing functionality. In this study, we extend these modules to make them interoperable with the IMS Bench SIPp traffic generation tool and Open IMS nodes when a pool of IMS VM instances are used. During the registration phase, when a UE initiates registration procedure, it sends a SIP REGISTER request to the LB, which acts as an outbound SIP proxy. LB processes this message and adds itself in (the topmost) Via and Record-Route headers of SIP REGISTER message before forwarding the message to the selected P-CSCF. By adding its information to these headers, LB will receive the corresponding response message to the request. In the reverse direction, LB removes the redundant information in these headers before forwarding the corresponding message to the UE. The above approach, which used for the registration phase, also used for the IMS call setup phase. For our experiments, we utilize Dispatcher Module for the selection of P-CSCF and IMS clients registered to all of the active IMS nodes.

For the emulation of a large number of IMS clients, we used open source IMS bench SIPp traffic generation tool [29]. It is a performance testing and benchmarking toolset designed to provide an implementation of a test system conforming to the IMS Performance Benchmark specification, ETSI TS 186 008. It is capable of not only testing IMS core networks but also standalone SIP proxies, SIP application servers, B2BUAs, etc., whether they are IMS compliant or not. Thanks to its large-scale benchmarking capabilities, deep automation and report generation functionalities, one is able to create realistic traffic loading scenarios for registration, calls, de-registrations, and messaging. The various traffic loading levels can be obtained by varying the cps (call per second) value in the scenario configuration file. Note that, in our experiments, we make sure that all of the subscribers registered to the Open IMS HSS processes of all active IMS VMs. Therefore, our load balancing algorithm can freely send an incoming new request to one of the active IMS VMs and provides a fine granular load distribution without any constraint.

The open source Zabbix monitoring tool [32] is used to obtain the resource utilizations of the active IMS VMs. A Zabbix client, which runs on each IMS VM instance, collects the CPU and the memory usage information. The Zabbix client periodically sends this information to the Zabbix server running at the LB. Note that the LB performs the load management using this feedback information, and instantiates a new IMS VM or shuts down an operational one if the average system load is higher and lower than the administratively specified maximum and minimum resource utilization thresholds, respectively. Starting a new VM or shutting down a VM is done by configured Zabbix trigger actions, which explained previously. The dynamic IMS VM provisioning is an important feature of the proposed load management framework to reduce the amount of the energy consumption in the system and an enabler for the green network paradigm.

Our test environment which are running all VMs is VMware workstation, which seen in Figure 3.14.

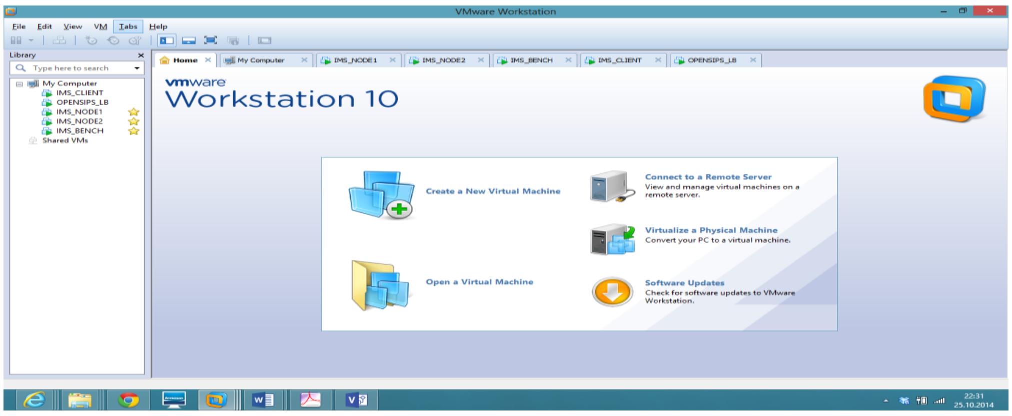


Figure 3.14 : Simulation environment in VMware Workstation

Following is a sample shell script for starting load balancer included scenario:

```
#!/bin/bash
i=1
while [  $i -lt 11 ]; do
echo -n "Connecting to LB"
echo -e "\n"
ssh 192.168.2.141 -t "/root/st.sh ;exit"
sleep 5
echo -n "Connecting to ims1"
ssh 192.168.2.3 -t "/st.sh ;exit"
sleep 20
echo -n "Connecting to ims2"
echo -e "\n"
ssh 192.168.2.5 -t "/st.sh ;exit"
sleep 60
cd /opt/ims_bench/
echo -n "running manager and sipp"
echo -e "\n"
./manager -f managersmall10.xml -e &
sleep 20
/opt/ims_bench/sipp -id 1 -sn uas  -i 192.168.2.11 -user_inf
/opt/ims_bench/ims_bench_0/ims_users_2.inf -rmctrl 192.168.2.11:5000
192.168.2.141:4060 -trace_err -trace_cpumem -trace_scen -trace_retrans -
trace_stat -trace_logs -trace_msg > trace.scen$i
```

```
mv /opt/ims_bench/dump*csv /opt/ims_bench/ims_bench_0
mv /opt/ims_bench/*error*log /opt/ims_bench/ims_bench_0
mv /opt/ims_bench/sipp_* /opt/ims_bench/ims_bench_0
mv /opt/ims_bench/uas_* /opt/ims_bench/ims_bench_0
mv /opt/ims_bench/trace.scen$i /opt/ims_bench/ims_bench_0
echo -n "creating log directory"
echo -e "\n"
cd /opt/ims_bench/ims_bench_0
mkdir scenlb10$i
../scripts/getResults.pl
mv trace.scen$i scenlb10$i
mv dump*csv scenlb10$i
mv *error*log scenlb10$i
cp sipp_* scenlb10$i
cp uas* scenlb10$i
i=`expr $i + 1`
done
```

Following is a sample shell script when no load balancer used in the scenario:

```
#!/bin/bash
i=1
while [  $i -lt 11 ]; do
echo -n "Connecting to LB"
echo -e "\n"
ssh 192.168.2.141 -t "/root/st.sh ;exit"
sleep 5
echo -n "Connecting to ims2"
echo -e "\n"
ssh 192.168.2.5 -t "/st.sh ;exit"
sleep 60
cd /opt/ims_bench/
echo -n "running manager and sipp"
echo -e "\n"
./manager -f managersmall10.xml -e &
sleep 20
/opt/ims_bench/sipp -id 1 -sn uas  -i 192.168.2.11 -user_inf
/opt/ims_bench/ims_bench_0/ims_users_2.inf -rmctrl 192.168.2.11:5000
192.168.2.5:4060 -trace_err -trace_cpumem -trace_scen -trace_retrans -
trace_stat -trace_logs -trace_msg > trace.scen$i
mv /opt/ims_bench/dump*csv /opt/ims_bench/ims_bench_0
mv /opt/ims_bench/*error*log /opt/ims_bench/ims_bench_0
mv /opt/ims_bench/sipp_* /opt/ims_bench/ims_bench_0
mv /opt/ims_bench/uas_* /opt/ims_bench/ims_bench_0
mv /opt/ims_bench/trace.scen$i /opt/ims_bench/ims_bench_0
echo -n "creating log directory"
echo -e "\n"
cd /opt/ims_bench/ims_bench_0
mkdir scennolb10$i
../scripts/getResults.pl
scennolb10$i
mv trace.scen$i scennolb10$i
mv dump*csv scennolb10$i
mv *error*log scennolb10$i
cp sipp_* scennolb10$i
cp uas* scennolb10$i
i=`expr $i + 1`
done
```

4. TEST RESULT

4.1 Testbed Experiment Results

In this chapter, we present the results of the testbed experiments to demonstrate the benefits of utilizing a load management mechanism for IMS networks using NFV. We set up a testbed consisting of IMS Bench SIPp, a load balancer, and two IMS VMs as shown in Figure 4.1. The open source tools as marked in the sofware implementation architecture in Figure 3.1 utilized for this testbed. To represent the conventional scenario without using any load balancer, the second testbed consisting of IMS Bench SIPp and one IMS VM as shown in Figure 4.2. set up. The Zabbix server has the capability of remotely instantiating a new IMS VM or shutting down an operational one according to the resource utilization levels of IMS VMs; however, in this thesis, we do not present any experiment for the dynamic VM provisioning feature of our proposed framework. Instead, we showed the capability of starting and stopping IMS VMs according to the traffic figures with Zabbix triggers and focus on quantifying how much benefit one will gain by using a load balancer with two IMS VM instances compared to the conventional scenario without using any load balancer. The proof-of-concept experiments demonstrate that the proposed framework significantly increases the scalability of the IMS networks for the highly loaded traffic scenarios.

For all experiments, a user list of 5000 subscribers registered for each active IMS VM. For each experiment, we generate a certain number of CPS for a pre-determined duration from IMS Bench SIPp and measure the call success rate CSR with and without LB. CSR defined as the ratio of the number of successful calls to the number of total generated calls. For the first set of the experiments, the number of total calls generated by IMS Bench SIPp varied by using the CPS values of 10, 20, 30, 40, and 50. For each traffic loading level, the experiments repeated for 10 times and the average of the CSR results reported. In the IMS Bench SIPp call generation tool, the call arrival process configured to use the deterministic inter-arrival times (i.e., constant CPS).

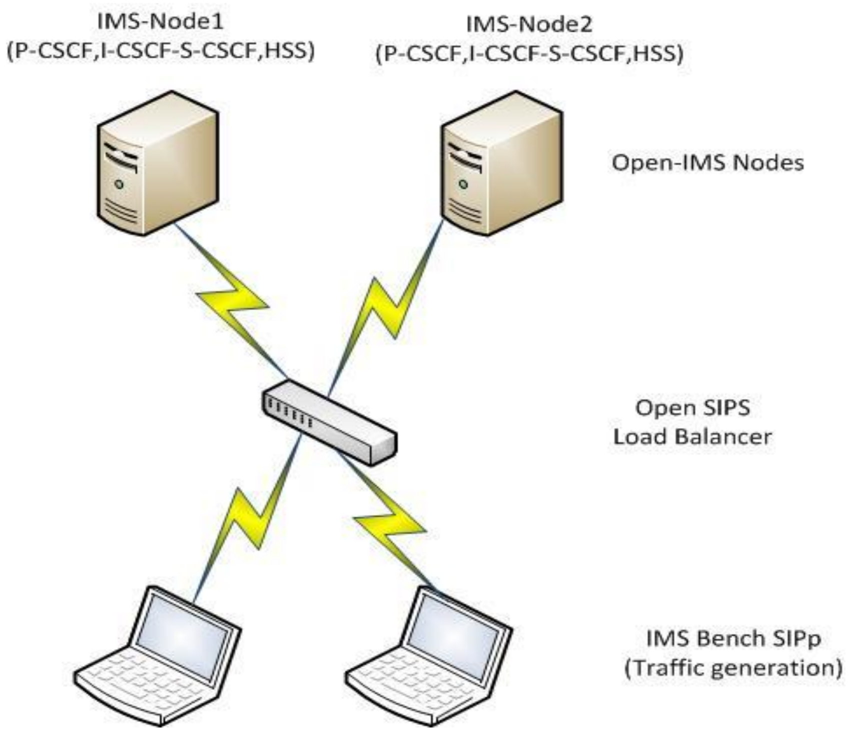


Figure 4.1 : Load Balancing with two IMS nodes.

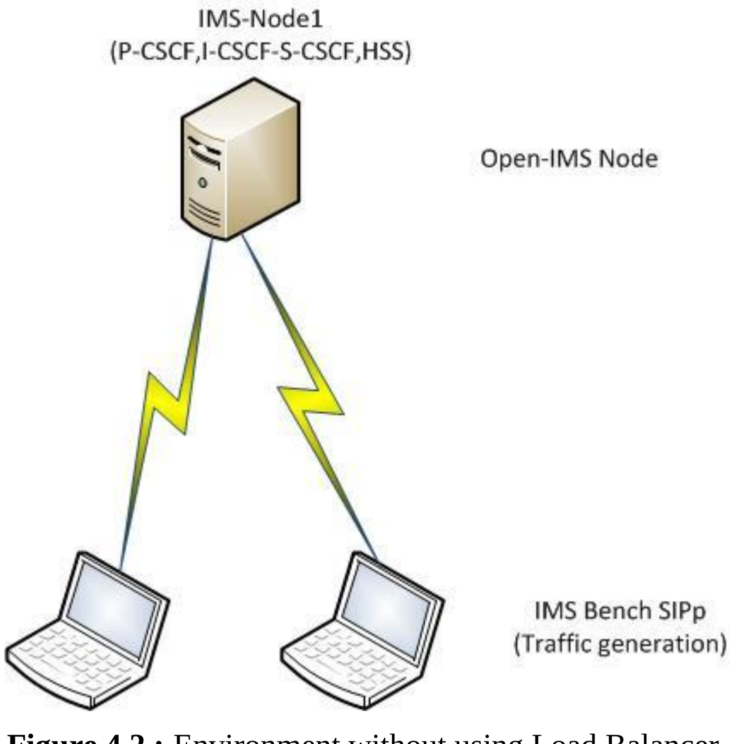


Figure 4.2 : Environment without using Load Balancer.

Figure 4.3. shows the CSR results with and without using the LB. For the CPS values of 10, 20, and 30, the CSR values without the LB are 1.0. While the CPS values of 10 and 20 correspond to the lightly and medium loaded traffic scenarios, respectively, the CPS value of 30 increases the CPU utilization of the IMS node up to 100%. For the case with the LB, while the CPS value of 10 and 20 yield the CSR values of 1.0, the

CSR value is around 0.99 when the CPS values is 30, indicating a slight degradation in the CSR performance. This is due to the fact that the LB introduces certain processing and signalling overheads since each request and its corresponding reply need to be processed and additional fields need to be added to the SIP headers by the LB acting as an outbound proxy. The messages exchange for a successful IMS call scenario for the case with and without using the LB given in Figure 4.4. and Figure 4.5., respectively. These figures indicate that having the LB as an outbound proxy between IMS clients and IMS nodes requires additional message processing. However, when the CPS values are 40 and 50, the CSR results are 0.925 and 0.584 for the case with the LB while they are 0.061 and 0.036 for the case without the LB, respectively. These results show that using the LB with two IMS VM instances significantly increases the scalability of IMS networks for the highly loaded traffic scenarios.

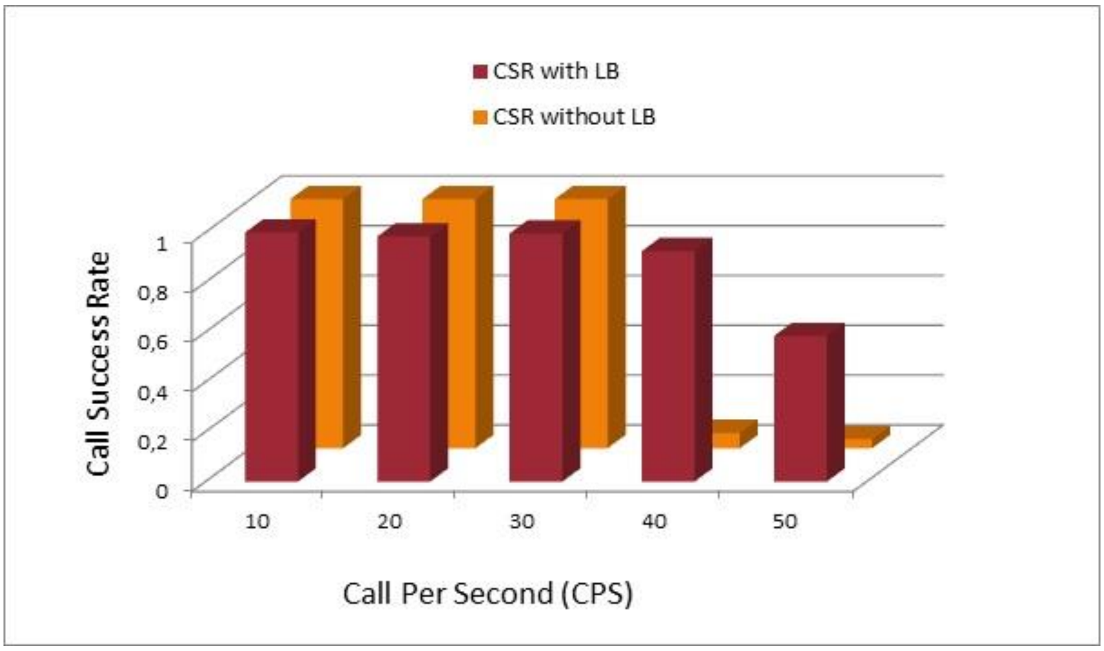
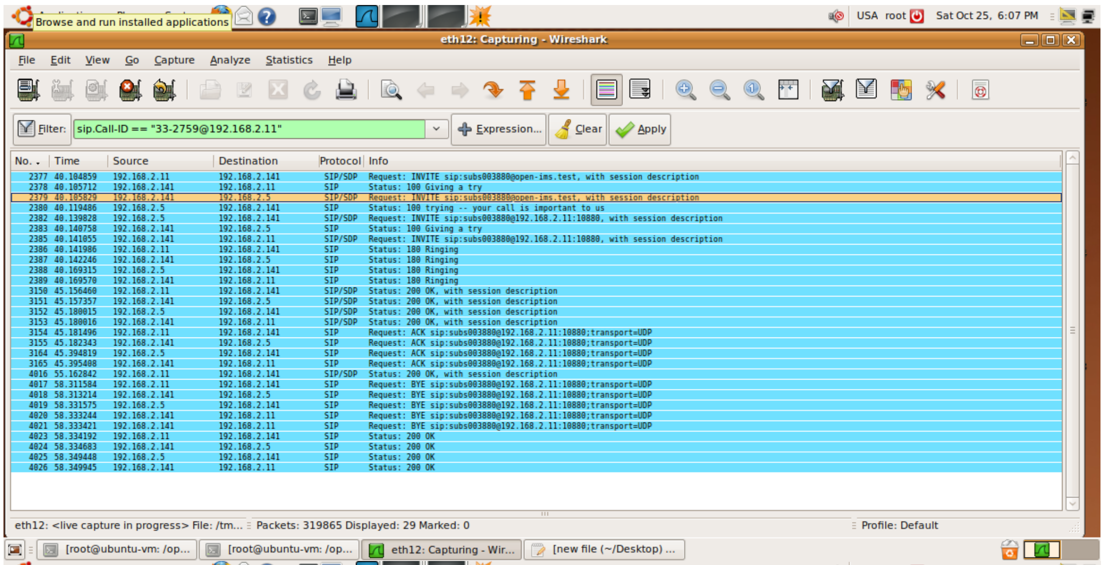
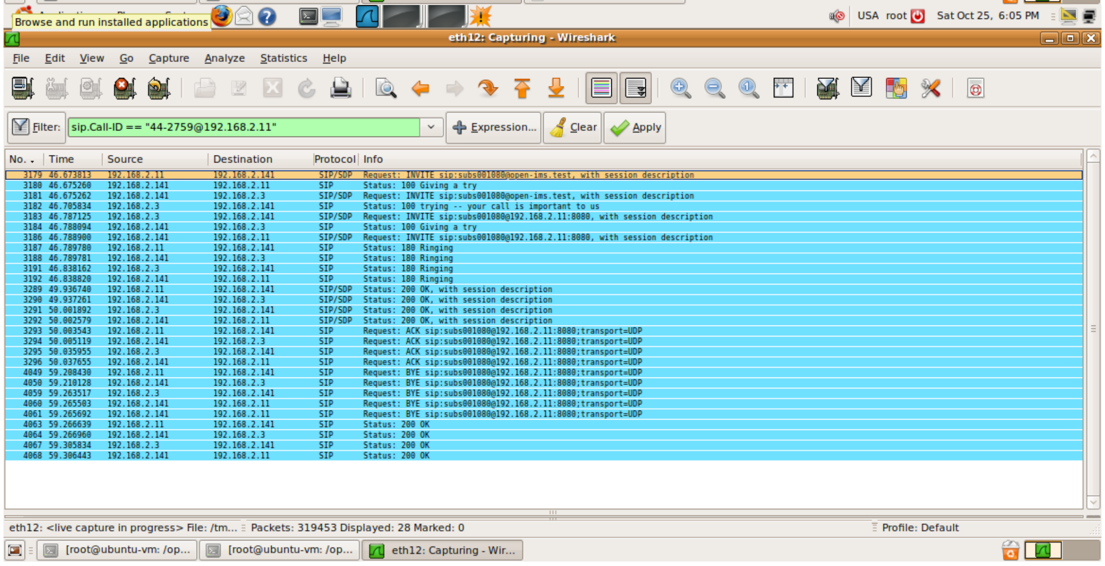


Figure 4.3 : The call success rates (CSRs) for various traffic demands.


Browse and run installed applications
USA root Sat Oct 25, 6:07 PM
eth12: Capturing - Wireshark
File Edit View Go Capture Analyze Statistics Help
Expression... Clear Apply
Profile: Default
[root@ubuntu-vm: /op...
[root@ubuntu-vm: /op...
eth12: Capturing - Wir...
[new file (~/Desktop) ...

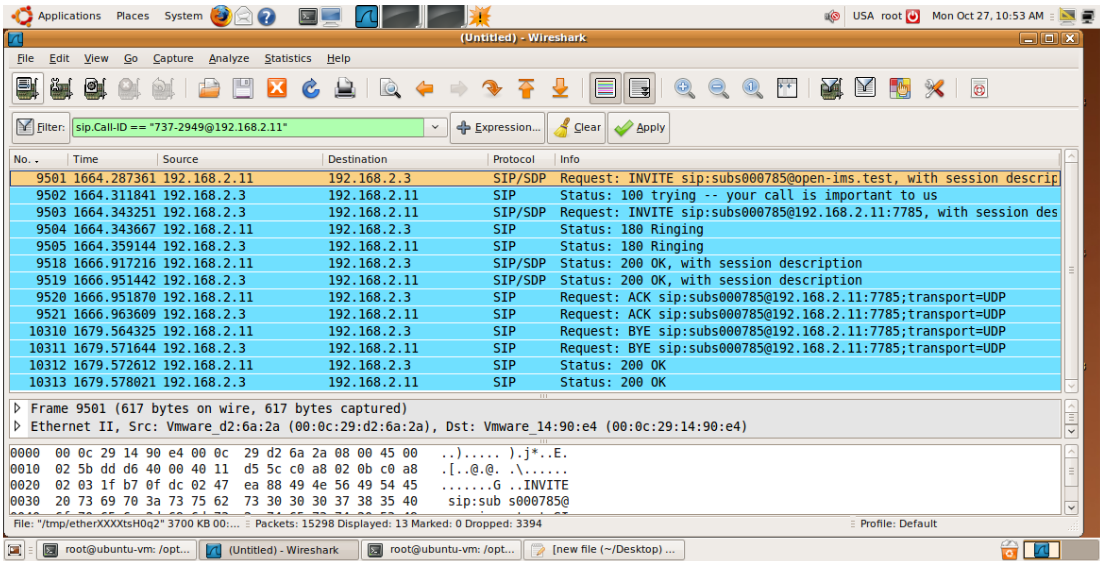
```
Filter: sip.Call-ID == "33-2759@192.168.2.11"

No. . Time      Source         Destination    Protocol Info
 2377 40.104859 192.168.2.11   192.168.2.141  SIP/SDP  Request: INVITE sip:subs003880@open-ims.test, with session description
 2378 40.105712 192.168.2.141  192.168.2.11   SIP      Status: 100 Giving a try
 2379 40.105829 192.168.2.141  192.168.2.5    SIP/SDP  Request: INVITE sip:subs003880@open-ims.test, with session description
 2380 40.119486 192.168.2.5    192.168.2.141  SIP      Status: 100 trying -- your call is important to us
 2382 40.139828 192.168.2.5    192.168.2.141  SIP/SDP  Request: INVITE sip:subs003880@192.168.2.11:10880, with session description
 2383 40.140758 192.168.2.141  192.168.2.5    SIP      Status: 100 Giving a try
 2385 40.141055 192.168.2.141  192.168.2.11   SIP/SDP  Request: INVITE sip:subs003880@192.168.2.11:10880, with session description
 2386 40.141986 192.168.2.11   192.168.2.141  SIP      Status: 180 Ringing
 2387 40.142246 192.168.2.141  192.168.2.5    SIP      Status: 180 Ringing
 2388 40.169315 192.168.2.5    192.168.2.141  SIP      Status: 180 Ringing
 2389 40.169570 192.168.2.141  192.168.2.11   SIP      Status: 180 Ringing
 3150 45.156460 192.168.2.11   192.168.2.141  SIP/SDP  Status: 200 OK, with session description
 3151 45.157357 192.168.2.141  192.168.2.5    SIP/SDP  Status: 200 OK, with session description
 3152 45.180015 192.168.2.5    192.168.2.141  SIP/SDP  Status: 200 OK, with session description
 3153 45.180016 192.168.2.141  192.168.2.11   SIP/SDP  Status: 200 OK, with session description
 3154 45.181496 192.168.2.11   192.168.2.141  SIP      Request: ACK sip:subs003880@192.168.2.11:10880;transport=UDP
 3155 45.182343 192.168.2.141  192.168.2.5    SIP      Request: ACK sip:subs003880@192.168.2.11:10880;transport=UDP
 3164 45.394819 192.168.2.5    192.168.2.141  SIP      Request: ACK sip:subs003880@192.168.2.11:10880;transport=UDP
 3165 45.395408 192.168.2.141  192.168.2.11   SIP      Request: ACK sip:subs003880@192.168.2.11:10880;transport=UDP
 4016 55.162842 192.168.2.11   192.168.2.141  SIP/SDP  Status: 200 OK, with session description
 4017 58.311584 192.168.2.11   192.168.2.141  SIP      Request: BYE sip:subs003880@192.168.2.11:10880;transport=UDP
 4018 58.313214 192.168.2.141  192.168.2.5    SIP      Request: BYE sip:subs003880@192.168.2.11:10880;transport=UDP
 4019 58.331575 192.168.2.5    192.168.2.141  SIP      Request: BYE sip:subs003880@192.168.2.11:10880;transport=UDP
 4020 58.333244 192.168.2.141  192.168.2.11   SIP      Request: BYE sip:subs003880@192.168.2.11:10880;transport=UDP
 4021 58.333421 192.168.2.141  192.168.2.11   SIP      Request: BYE sip:subs003880@192.168.2.11:10880;transport=UDP
 4023 58.334192 192.168.2.11   192.168.2.141  SIP      Status: 200 OK
 4024 58.334683 192.168.2.141  192.168.2.5    SIP      Status: 200 OK
 4025 58.349448 192.168.2.5    192.168.2.141  SIP      Status: 200 OK
 4026 58.349945 192.168.2.141  192.168.2.11   SIP      Status: 200 OK

eth12: <live capture in progress> File: /tm... Packets: 319865 Displayed: 29 Marked: 0
```


Browse and run installed applications
USA root Sat Oct 25, 6:05 PM
eth12: Capturing - Wireshark
File Edit View Go Capture Analyze Statistics Help
Expression... Clear Apply
Profile: Default
[root@ubuntu-vm: /op...
[root@ubuntu-vm: /op...
eth12: Capturing - Wir...

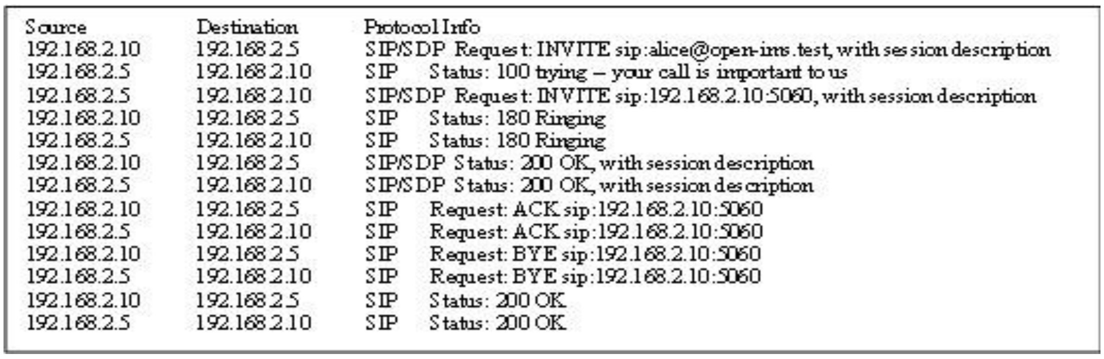
```
Filter: sip.Call-ID == "44-2759@192.168.2.11"

No. . Time      Source         Destination    Protocol Info
 3179 46.673813 192.168.2.11   192.168.2.141  SIP/SDP  Request: INVITE sip:subs001080@open-ims.test, with session description
 3180 46.675260 192.168.2.141  192.168.2.11   SIP      Status: 100 Giving a try
 3181 46.675262 192.168.2.141  192.168.2.3    SIP/SDP  Request: INVITE sip:subs001080@open-ims.test, with session description
 3182 46.705834 192.168.2.3    192.168.2.141  SIP      Status: 100 trying -- your call is important to us
 3183 46.787125 192.168.2.3    192.168.2.141  SIP/SDP  Request: INVITE sip:subs001080@192.168.2.11:8080, with session description
 3184 46.788094 192.168.2.141  192.168.2.3    SIP      Status: 100 Giving a try
 3186 46.788900 192.168.2.141  192.168.2.11   SIP/SDP  Request: INVITE sip:subs001080@192.168.2.11:8080, with session description
 3187 46.789780 192.168.2.11   192.168.2.141  SIP      Status: 180 Ringing
 3188 46.789781 192.168.2.141  192.168.2.3    SIP      Status: 180 Ringing
 3191 46.838162 192.168.2.3    192.168.2.141  SIP      Status: 180 Ringing
 3192 46.838820 192.168.2.141  192.168.2.11   SIP      Status: 180 Ringing
 3289 49.936740 192.168.2.11   192.168.2.141  SIP/SDP  Status: 200 OK, with session description
 3290 49.937261 192.168.2.141  192.168.2.3    SIP/SDP  Status: 200 OK, with session description
 3291 50.001892 192.168.2.3    192.168.2.141  SIP/SDP  Status: 200 OK, with session description
 3292 50.002579 192.168.2.141  192.168.2.11   SIP/SDP  Status: 200 OK, with session description
 3293 50.003543 192.168.2.11   192.168.2.141  SIP      Request: ACK sip:subs001080@192.168.2.11:8080;transport=UDP
 3294 50.005119 192.168.2.141  192.168.2.3    SIP      Request: ACK sip:subs001080@192.168.2.11:8080;transport=UDP
 3295 50.035955 192.168.2.3    192.168.2.141  SIP      Request: ACK sip:subs001080@192.168.2.11:8080;transport=UDP
 3296 50.037655 192.168.2.141  192.168.2.11   SIP      Request: ACK sip:subs001080@192.168.2.11:8080;transport=UDP
 4049 59.208430 192.168.2.11   192.168.2.141  SIP      Request: BYE sip:subs001080@192.168.2.11:8080;transport=UDP
 4050 59.210128 192.168.2.141  192.168.2.3    SIP      Request: BYE sip:subs001080@192.168.2.11:8080;transport=UDP
 4059 59.263517 192.168.2.3    192.168.2.141  SIP      Request: BYE sip:subs001080@192.168.2.11:8080;transport=UDP
 4060 59.265503 192.168.2.141  192.168.2.11   SIP      Request: BYE sip:subs001080@192.168.2.11:8080;transport=UDP
 4061 59.265692 192.168.2.141  192.168.2.11   SIP      Request: BYE sip:subs001080@192.168.2.11:8080;transport=UDP
 4063 59.266639 192.168.2.11   192.168.2.141  SIP      Status: 200 OK
 4064 59.266960 192.168.2.141  192.168.2.3    SIP      Status: 200 OK
 4067 59.305834 192.168.2.3    192.168.2.141  SIP      Status: 200 OK
 4068 59.306443 192.168.2.141  192.168.2.11   SIP      Status: 200 OK

eth12: <live capture in progress> File: /tm... Packets: 319453 Displayed: 28 Marked: 0
```

Source	Destination	Protocol	Info
192.168.2.11	192.168.2.141	SIP/SDP	Request: INVITE sip:subs004467@open-ims.test, with session description
192.168.2.141	192.168.2.11	SIP	Status: 100 Giving a try
192.168.2.141	192.168.2.3	SIP/SDP	Request: INVITE sip:subs004467@open-ims.test, with session description
192.168.2.3	192.168.2.141	SIP	Status: 100 trying -- your call is important to us
192.168.2.3	192.168.2.141	SIP/SDP	Request: INVITE sip:subs004467@192.168.2.11:11468, with session description
192.168.2.141	192.168.2.3	SIP	Status: 100 Giving a try
192.168.2.141	192.168.2.11	SIP/SDP	Request: INVITE sip:subs004467@192.168.2.11:11468, with session description
192.168.2.11	192.168.2.141	SIP	Status: 180 Ringing
192.168.2.141	192.168.2.3	SIP	Status: 180 Ringing
192.168.2.3	192.168.2.141	SIP	Status: 180 Ringing
192.168.2.141	192.168.2.11	SIP	Status: 180 Ringing
192.168.2.11	192.168.2.141	SIP/SDP	Status: 200 OK, with session description
192.168.2.141	192.168.2.3	SIP/SDP	Status: 200 OK, with session description
192.168.2.3	192.168.2.141	SIP/SDP	Status: 200 OK, with session description
192.168.2.141	192.168.2.11	SIP/SDP	Status: 200 OK, with session description
192.168.2.11	192.168.2.141	SIP	Request: ACK sip:subs004467@192.168.2.11:11468;transport=UDP
192.168.2.141	192.168.2.3	SIP	Request: ACK sip:subs004467@192.168.2.11:11468;transport=UDP
192.168.2.3	192.168.2.141	SIP	Request: ACK sip:subs004467@192.168.2.11:11468;transport=UDP
192.168.2.141	192.168.2.11	SIP	Request: ACK sip:subs004467@192.168.2.11:11468;transport=UDP
192.168.2.11	192.168.2.141	SIP	Request: BYE sip:subs004467@192.168.2.11:11468;transport=UDP
192.168.2.141	192.168.2.3	SIP	Request: BYE sip:subs004467@192.168.2.11:11468;transport=UDP
192.168.2.3	192.168.2.141	SIP	Request: BYE sip:subs004467@192.168.2.11:11468;transport=UDP
192.168.2.141	192.168.2.11	SIP	Request: BYE sip:subs004467@192.168.2.11:11468;transport=UDP
192.168.2.11	192.168.2.141	SIP	Status: 200 OK
192.168.2.141	192.168.2.3	SIP	Status: 200 OK
192.168.2.3	192.168.2.141	SIP	Status: 200 OK
192.168.2.141	192.168.2.11	SIP	Status: 200 OK

Figure 4.4 : Successful IMS call scenario with the LB to different nodes.



```
Filter: sip.Call-ID == "737-2949@192.168.2.11"

No. .  Time        Source        Destination   Protocol  Info
 9501 1664.287361 192.168.2.11  192.168.2.3   SIP/SDP   Request: INVITE sip:subs000785@open-ims.test, with session descrip
 9502 1664.311841 192.168.2.3   192.168.2.11  SIP       Status: 100 trying -- your call is important to us
 9503 1664.343251 192.168.2.3   192.168.2.11  SIP/SDP   Request: INVITE sip:subs000785@192.168.2.11:7785, with session des
 9504 1664.343667 192.168.2.11  192.168.2.3   SIP       Status: 180 Ringing
 9505 1664.359144 192.168.2.3   192.168.2.11  SIP       Status: 180 Ringing
 9518 1666.917216 192.168.2.11  192.168.2.3   SIP/SDP   Status: 200 OK, with session description
 9519 1666.951442 192.168.2.3   192.168.2.11  SIP/SDP   Status: 200 OK, with session description
 9520 1666.951870 192.168.2.11  192.168.2.3   SIP       Request: ACK sip:subs000785@192.168.2.11:7785;transport=UDP
 9521 1666.963609 192.168.2.3   192.168.2.11  SIP       Request: ACK sip:subs000785@192.168.2.11:7785;transport=UDP
10310 1679.564325 192.168.2.11  192.168.2.3   SIP       Request: BYE sip:subs000785@192.168.2.11:7785;transport=UDP
10311 1679.571644 192.168.2.3   192.168.2.11  SIP       Request: BYE sip:subs000785@192.168.2.11:7785;transport=UDP
10312 1679.572612 192.168.2.11  192.168.2.3   SIP       Status: 200 OK
10313 1679.578021 192.168.2.3   192.168.2.11  SIP       Status: 200 OK

▷ Frame 9501 (617 bytes on wire, 617 bytes captured)
▷ Ethernet II, Src: Vmware_d2:6a:2a (00:0c:29:d2:6a:2a), Dst: Vmware_14:90:e4 (00:0c:29:14:90:e4)

0000  00 0c 29 14 90 e4 00 0c  29 d2 6a 2a 08 00 45 00   ..)..... ).j*..E.
0010  02 5b dd d6 40 00 40 11  d5 5c c0 a8 02 0b c0 a8   .[..@.@. .\......
0020  02 03 1f b7 0f dc 02 47  ea 88 49 4e 56 49 54 45   .......G ..INVITE
0030  20 73 69 70 3a 73 75 62  73 30 30 30 37 38 35 40    sip:sub s000785@

File: "/tmp/etherXXXXtsH0q2" 3700 KB 00:...  Packets: 15298 Displayed: 13 Marked: 0 Dropped: 3394   Profile: Default
```



Source	Destination	Protocol	Info
192.168.2.10	192.168.2.5	SIP/SDP	Request: INVITE sip:alice@open-ims.test, with session description
192.168.2.5	192.168.2.10	SIP	Status: 100 trying – your call is important to us
192.168.2.5	192.168.2.10	SIP/SDP	Request: INVITE sip:192.168.2.10:5060, with session description
192.168.2.10	192.168.2.5	SIP	Status: 180 Ringing
192.168.2.5	192.168.2.10	SIP	Status: 180 Ringing
192.168.2.10	192.168.2.5	SIP/SDP	Status: 200 OK, with session description
192.168.2.5	192.168.2.10	SIP/SDP	Status: 200 OK, with session description
192.168.2.10	192.168.2.5	SIP	Request: ACK sip:192.168.2.10:5060
192.168.2.5	192.168.2.10	SIP	Request: ACK sip:192.168.2.10:5060
192.168.2.10	192.168.2.5	SIP	Request: BYE sip:192.168.2.10:5060
192.168.2.5	192.168.2.10	SIP	Request: BYE sip:192.168.2.10:5060
192.168.2.10	192.168.2.5	SIP	Status: 200 OK
192.168.2.5	192.168.2.10	SIP	Status: 200 OK

Figure 4.5 : Successful IMS call scenario without the LB.

For the second set of experiments, we evaluated the performance of the proposed load management framework when there are sudden significant increases in traffic demands. For example, the CPS values of the IMS traffic arrivals suddenly increased from 10 to 20, 30, 40, and 50 to emulate this behaviour. In real life, this corresponds to the scenario that the users make calls after an unexpected event such as a disaster or earthquake occurs. Each experiment repeated for 10 times and the average CSR results reported for the case with and without the LB.

Figure 4.6. shows the CSR results with and without using the LB. When the CPS values increased to 20 and 30, the CSR values without the LB are 1.0 and 0.99, respectively while the CSR values with the LB are 0.77 and 0.79, respectively. The performance degradation with the LB can explained by the fact that the LB introduces certain processing and signalling overheads by acting as an outbound proxy. However, when the CPS values increased to 40 and 50, the CSR results are 0.65 and 0.55 for the case with the LB while they are 0.38 and 0.31 for the case without the LB, respectively.

Note that the CSR values are higher compared to the CPS values of 40 and 50 in Figure 4.3. since the CPS value of 10 is used for a certain duration of the scenario before the traffic demand suddenly increases. These results show that using the LB with two IMS VM instances significantly increases the scalability of IMS networks when there are significant increases in the traffic demand.

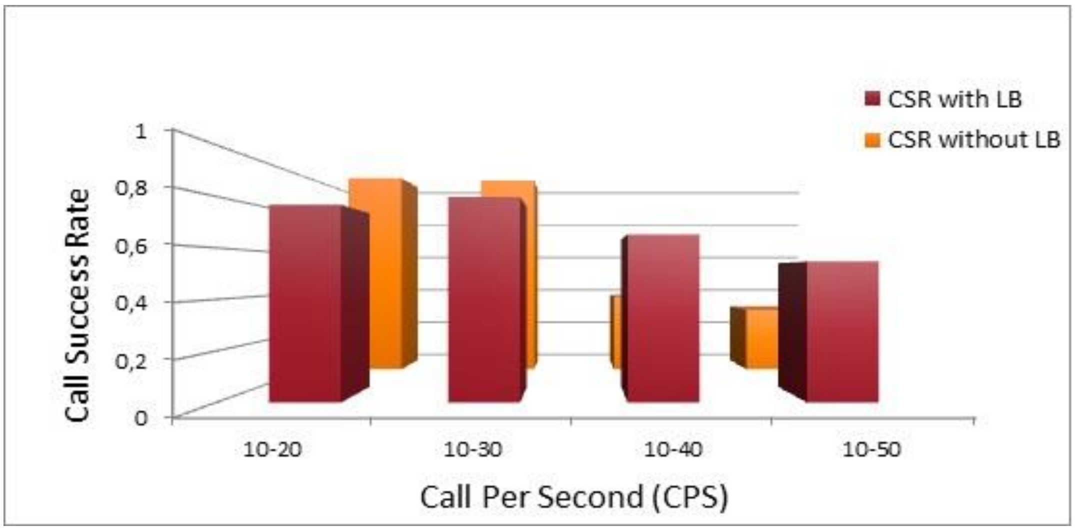


Figure 4.6 : The call success rates (CSRs) for sudden increases in traffic demands.

Figure 4.7. and Figure 4.8. shows the CSR results with and without using the LB by using Poisson traffic distribution. These results are also similar with previous test results done with constant traffic distribution.

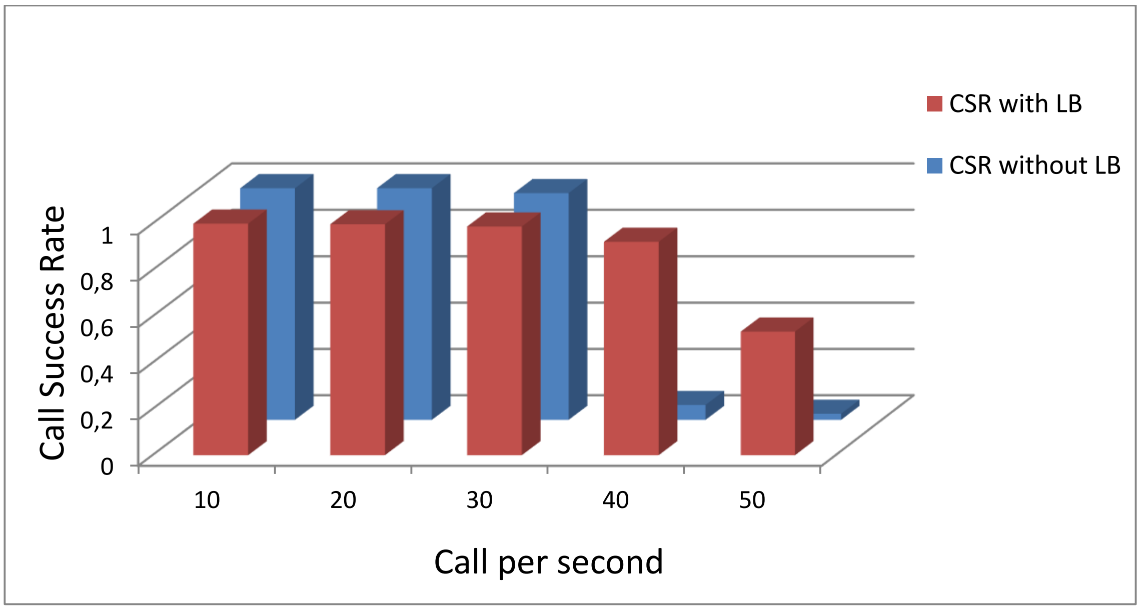


Figure 4.7 : The call success rates (CSRs) for various traffic demands.(Poisson)

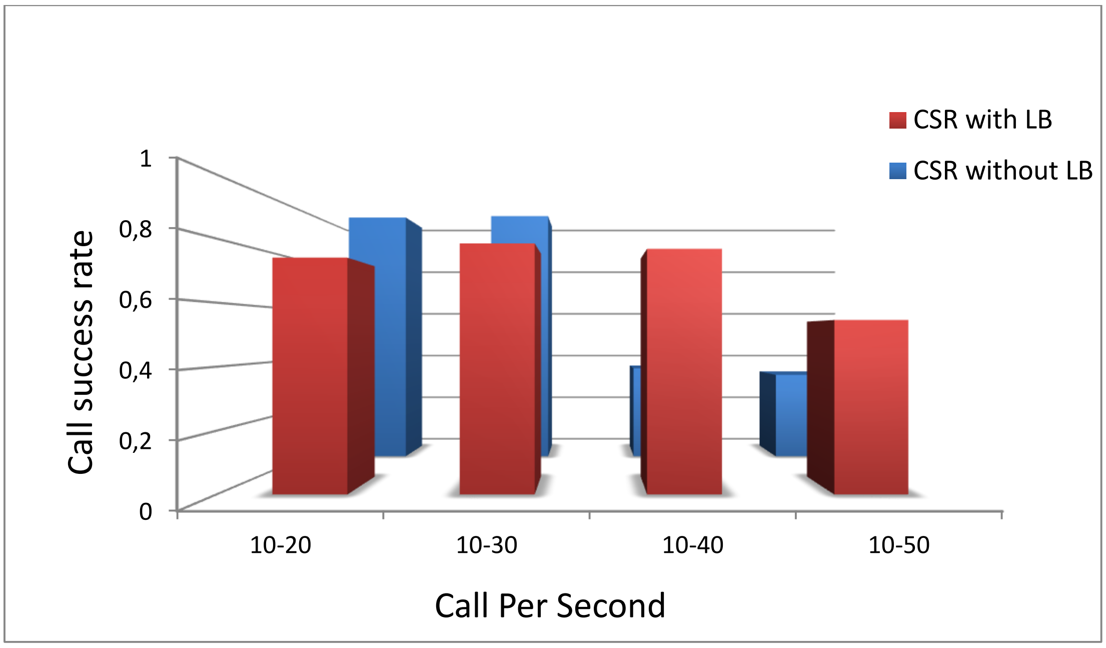


Figure 4.8 : The call success rates (CSRs) for sudden increases in traffic demands. (Poisson)

To see the detailed traffic distribution between 30 and 50 cps, where the setup with LB performance is higher than the setup without LB we run the tests once more, with 1 cps step increase and repeated the tests again 10 times for each cps value and took the average values. As it can be seen also in Figure 4.9, LB increases the overall performance significantly.

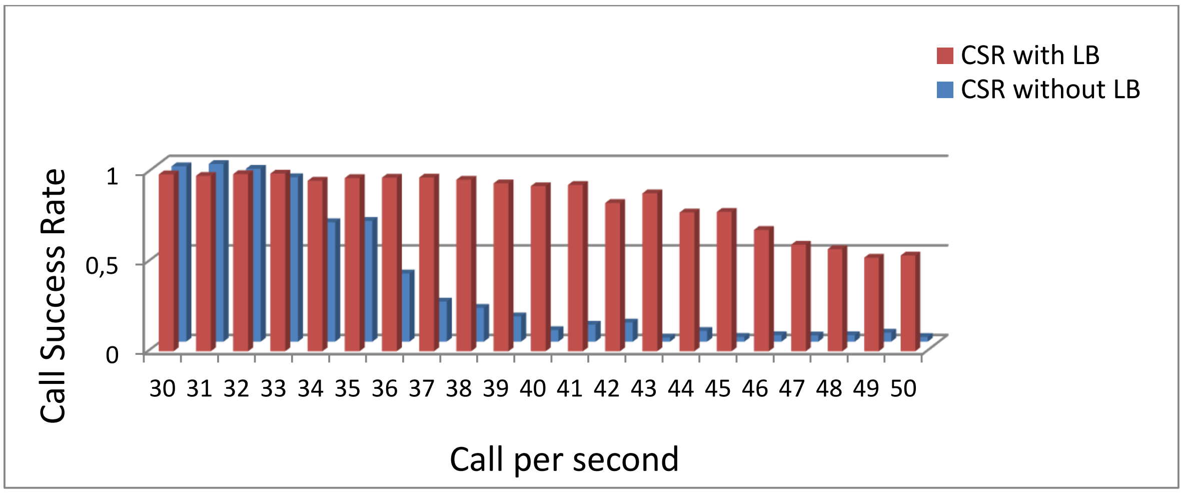


Figure 4.9 : Detailed call success rates (CSRs) between 30-50 CPS

Below, IMS registration scenarios given in Figure 4.10 and 4.11 with and without authentication respectively. As explained previously we made the registration to all of the active nodes to distribute the traffic efficiently by LB.

```
Source          Destination     Protocol Info
192.168.2.11    192.168.2.141   SIP   Request: REGISTER sip:open-ims.test
192.168.2.141   192.168.2.3     SIP   Request: REGISTER sip:open-ims.test
192.168.2.141   192.168.2.5     SIP   Request: REGISTER sip:open-ims.test
192.168.2.5     192.168.2.141   SIP   Status: 401 Unauthorized - Challenging the UE   (0 bindings)
192.168.2.141   192.168.2.11    SIP   Status: 401 Unauthorized - Challenging the UE   (0 bindings)
192.168.2.11    192.168.2.141   SIP   Request: REGISTER sip:open-ims.test
192.168.2.141   192.168.2.3     SIP   Request: REGISTER sip:open-ims.test
192.168.2.141   192.168.2.5     SIP   Request: REGISTER sip:open-ims.test
192.168.2.3     192.168.2.141   SIP   Status: 401 Unauthorized - Challenging the UE   (0 bindings)
192.168.2.3     192.168.2.141   SIP   Status: 401 Unauthorized - Challenging the UE   (0 bindings)
192.168.2.5     192.168.2.141   SIP   Status: 200 OK - SAR succesful and registrar saved   (1 bindings)
192.168.2.141   192.168.2.11    SIP   Status: 200 OK - SAR succesful and registrar saved   (1 bindings)
192.168.2.3     192.168.2.141   SIP   Status: 200 OK - SAR succesful and registrar saved   (1 bindings)
192.168.2.141   192.168.2.11    SIP   Status: 200 OK - SAR succesful and registrar saved   (1 bindings)
```

Figure 4.10 : Successful registration scenario using Open SIPS Load balancer – to both nodes with authentication.

```
Source          Destination     Protocol Info
192.168.2.11    192.168.2.141   SIP   Request: REGISTER sip:open-ims.test
192.168.2.141   192.168.2.3     SIP   Request: REGISTER sip:open-ims.test
192.168.2.141   192.168.2.5     SIP   Request: REGISTER sip:open-ims.test
192.168.2.5     192.168.2.141   SIP   Status: 200 OK - SAR succesful and registrar saved   (1 bindings)
192.168.2.141   192.168.2.11    SIP   Status: 200 OK - SAR succesful and registrar saved   (1 bindings)
192.168.2.3     192.168.2.141   SIP   Status: 200 OK - SAR succesful and registrar saved   (1 bindings)
192.168.2.141   192.168.2.11    SIP   Status: 200 OK - SAR succesful and registrar saved   (1 bindings)
```

Figure 4.11 : Successful registration scenario using Open SIPS Load balancer – to both nodes without authentication.

5. CONCLUSION AND FUTURE WORK

There has been a lot of ongoing research efforts addresing different aspects of the realization of the NFV of telecommunication networks at the cloud. NFV is expected to simplfy the deployment and management of constantly evolving and highly complex networking services and hence provides a cost effective solution. In this paper, we proposed a dynamic load management framework for IMS networks using NFV. We created IMS functions within a single VM instance and the incoming requests are distributed over a pool of multiple IMS VM instances running at the cloud environment. In the proposed framework, the number of IMS VMs is dynamically changed according to their utilization levels. The open source tools are used to set up an IMS testbed and demonstrate the benefits of the proposed dynamic load management framework. The proof-of-concept experiments demonstrated that the proposed load management framework significantly increases the scalability of the IMS networks for the highly loaded traffic scenarios. It also provides an effective method to handle sudden significant traffic changes in the network.

As future work, the first step will be to develop new mechanisms to minimize the detrimental effects of processing and signaling overheads of the LB. A pool of IMS VMs can be used to increase the reliability of the IMS networks as well. We also plan to develop a mechanism for migrating the active calls from one IMS VM to another when there is an IMS VM failure. This can used to save more energy by migrating the active calls without waiting their completions when there are only a few of active calls in one of the IMS VM.

REFERENCES

[1] 3GPP. TS 23.228. IP Multimedia Subsystem (IMS); Stage 2

[2] A. Basta et. al. , “A Virtual SDN-enabled LTE EPC Architecture: a case study for S-/P-Gateways functions”, \IEEE SDN for Future Networks and Services (SDN4FNS), November 2013.

[3] A. Csaszar et. al., “Unifying Cloud and Carrier Network: EU FP7 Project UNIFY”, \emph{IEEE/ACM 6th International Conference on Utility and Cloud Computing}, 2013.

[4] C.Makaya, A.Dutta, S.Das, D.Chee, F.J.Lin, S.Komorita, H.Yokota, H.Schulzrinne, "Service Continuity Support in Self-Organizing IMS Networks," 2nd International Conference on Wireless Communication, Vehicular Technology, Information Theory and Aerospace \& Electronic Systems Technology (Wireless VITAE), Chennai, India, 2011.

[5] C.Makaya, A.Dutta, S.Komorita, H.Yokota, H.Schulzrinne, "User-transparent Reconfiguration Method for Self-Organizing IMS Networks," IEEE, 2011.

[6] C.Makaya, A.Dutta, S.Das, D.Chee, F.J.Lin, S.Komorita, H.Yokota, H.Schulzrinne, "Load Balancing Support for Self-Organizing IMS Networks", 2011.

[7] D.Vingarzan, "Design and Implementation Aspects of Open Source Next Generation Networks (NGN) Test-bed Software Toolkits," PhD thesis, Technische Universitat Berlin, Berlin, Germany, 2013.

[8] ETSI GS NFV, “Network Functions Virtualisation (NFV); Use Cases,” V1.1.1, 2013-10; available at http://docbox.etsi.org/ISG/NFV/Open/Published/.

[9] G. Carella et. al., “Cloudified IP Multimedia Subsystem (IMS) for Network Function Virtualization (NFV)-based architectures”, IEEE ISCC, 2014.

[10] H.Jiang, A.Iyengar, E.Nahum, W.Segmuller, A.N.Tantawi, and C.P.Wright,"Design, Implementation, and Performance of a Load

Balancer for SIP Server Clusters," IEEE/ACM Transactions on Networking, Vol. 20, No. 4, pp. 1190–1202, Aug. 2012.

[11] **H. Zhang** et. al., “Proactive Workload Management in Hybrid Cloud Computing”, IEEE Transactions on Network and Service Management, March 2014.

[12] **Isam Abdalla, S.Venkatesan,** “Notification based S-CSCF Load Balancing in IMS Networks”, IEEE, 2011.

[13] **J.Kogel, S.Wahl, M.Scharf, and M.C.Necker**, "Load Sharing in a Distributed IMS Architecture," IEEE 69th Vehicular Technology Conference (VTC), Barcelona, Spain, 2009.

[14] **J.Zha, J.Wang, R.Han, and M.Song**, "Research on Load Balance of Service Capability Interaction Management," 3rd IEEE International Conference on Broadband Network and Multimedia Technology (IC-BNMT), Beijing, China, 2010

[15] **J. Zheng** et. al., “Pacer: A Progress Management System for Live Virtual Machine Migration in Cloud Computing”, IEEE Transactions on Network and Service Management, Vol. 10, No. 4, December 2013.

[16] **Karlsson, A.D.O. and Martin, B**., 2006, ATCA: its performance and application for real time systems, IEEE Transactions on Nuclear Science, 53(3), 688-693.

[17] **Kerpez, K., Cioffi, J. Ginis, G. Goldburg, M.Galli, S.Silverman**, 2014, P.Software-defined Access networks. Communications Magazine, IEEE, 52(9), 152-159.

[18]**Liang Xu, Changcheng Huang, James Yan, Tadeusz Drwiega**, “De-Registration Based S-CSCF Load Balancing in IMS Core Network”, IEEE ICC 2009 proceedings

[19] **Liu, S., Cai,Z., Xu, H., Xu**, M. 2014. Security-aware virtual network embedding. \IEEE International Conference on Communications (ICC).

[20] **M.Umair**, "Performance Evaluation and Elastic Scaling of an IP Multimedia Subsystem Implemented in a Cloud," Master's thesis, Fraunhofer FOKUS Competence Center NGNI, Berlin, Germany, 2013.

[21] **M.Simjanoska, S.Ristov, G.Velkoski, and M.Gusev**, "L3B: Low Level Load Balancer in the Cloud", IEEE EUROCON, Zagreb, Croatia, 2013

[22] Plarent Tirana, Deep Medhi, “Distributed Approaches to S-CSCF Selection in and IMS Network”, IEEE/IFIP Network Operation & Management (NOMS 2010), Osaka, Japan, April 2010

[23] R. Bolla et. al., “DROPv2: Energy Efficiency through Network Function Virtualization”, IEEE Communication Magazine, March/April 2014.

[24] S. Clayman et. al., “The Dynamic Placement of Virtual Network Functions”, IEEE Network Operations and Management Symposium (NOMS), May 2014.

[25] Q. Duan et. al., “A Survey on Service-Oriented Network Virtualization Toward Convergence of Networking and Cloud Computing”, IEEE Transactions on Network and Service Management, Vol. 9, No. 4, December 2012.

[26] W. Cerroni et. al., “Live Migration of Virtual Network Functions in Cloud-Based Edge Networks”, IEEE ICC, 2014.

[27] Vaquero, L. M., Rodero-Merino, L., Caceres, J., Linder, M. 2008. A break in the clouds: Towards a cloud definition. ACM SIGCOMM Computer Communication Review, 39(1), 50–55.

[28] Y. Jiang et. al., “Cloud Analytics for Capacity Planning and Instant VM Provisioning”, IEEE Transactions on Network and Service Management, Vol. 10, No. 3, September 2013.

[29]Url-1 <http://sipp.sourceforge.net/ims_bench/reference.html>, date retrieved: 23.10.2014.

[30] Url-2 <http://www.opensips.org>, date retrieved: 23.10.2014.

[31] Url-3 <http://www.openimscore.org>, date retrieved: 23.10.2014.

[32] Url-4 <http://www.zabbix.com>, date retrieved: 23.10.2014.

[33] Url-5 <http://www.iptel.org/ser>, date retrieved: 23.10.2014.

[34] Url-6 <https://www.opnfv.org>, date retrieved: 23.10.2014.

APPENDICES

APPENDIX A

A.1 Prerequisites

First prepare the environment by installing the following packages[31].

```
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_HOME” 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 configured with localhost. First copy the zonefile “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

B.1 Pre-requisites

You have to install the following necessary packages[29]:

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

B.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
```

B.3 Build IMS Bench SIPp source tree

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

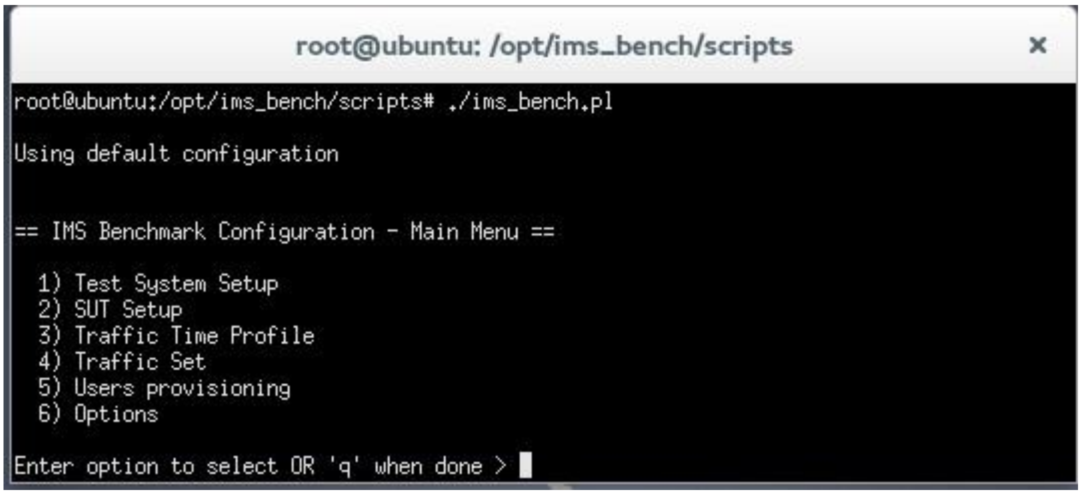
```
pwd
/root/ims_bench/
make rmtl
make ossl
make mgr
```

B.4 Configuration of IMS Bench SIPp

The test system for a benchmark run can be configured using a 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
```

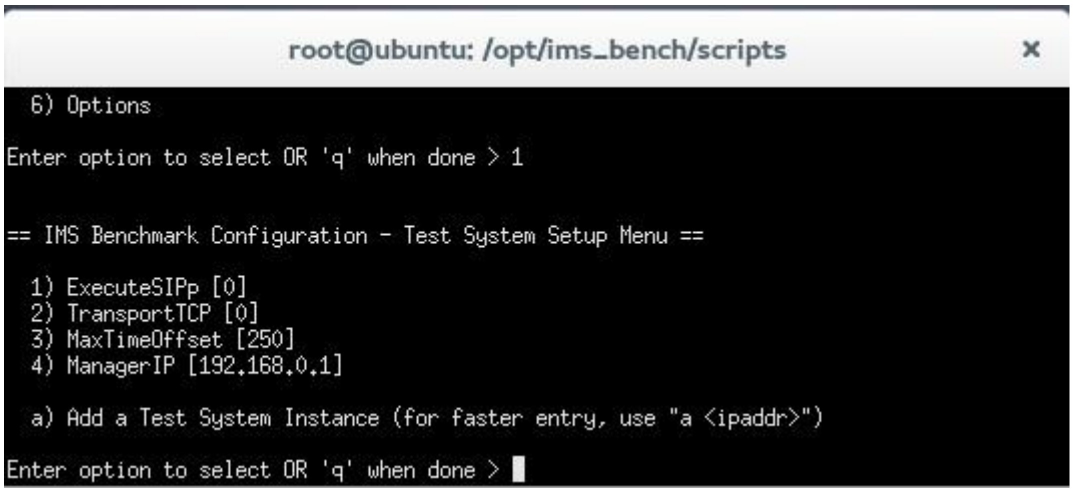
After executing the above command, following IMS Benchmark configuration main menu shown as in Figure B.1.



```
root@ubuntu: /opt/ims_bench/scripts

root@ubuntu:/opt/ims_bench/scripts# ./ims_bench.pl

Using default configuration


== IMS Benchmark Configuration - Main Menu ==

  1) Test System Setup
  2) SUT Setup
  3) Traffic Time Profile
  4) Traffic Set
  5) Users provisioning
  6) Options

Enter option to select OR 'q' when done >
```

Figure B.1: IMS benchmark configuration menu

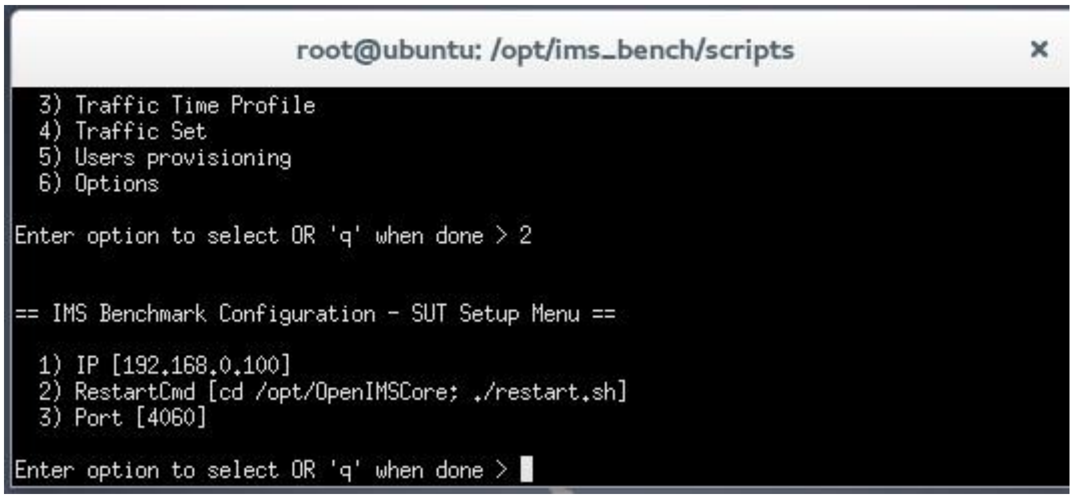
Press '1' to go inside the test system menu and configured the manager and SIPp TS instance IP addresses. Figure B.2 shows an example of the test system configuration. Press 'q' to return to main menu.



```
root@ubuntu: /opt/ims_bench/scripts
  6) Options

Enter option to select OR 'q' when done > 1


== IMS Benchmark Configuration - Test System Setup Menu ==

  1) ExecuteSIPp [0]
  2) TransportTCP [0]
  3) MaxTimeOffset [250]
  4) ManagerIP [192.168.0.1]

  a) Add a Test System Instance (for faster entry, use "a <ipaddr>")

Enter option to select OR 'q' when done >
```

Figure B.2: Test system configuration menu

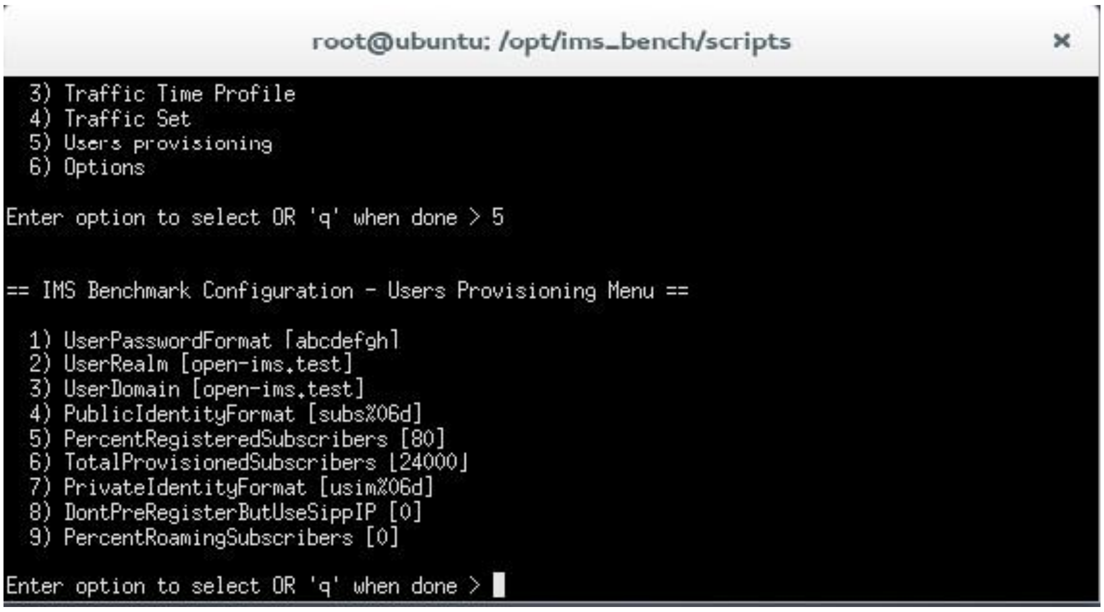
Press ’2’ to go inside the SUT setup menu and configure the OpenSIPS VM IP address. Figure B.3 shows an example of the SUT configuration. Press ’q’ to return to main menu.



```
root@ubuntu: /opt/ims_bench/scripts
  3) Traffic Time Profile
  4) Traffic Set
  5) Users provisioning
  6) Options

Enter option to select OR 'q' when done > 2


== IMS Benchmark Configuration - SUT Setup Menu ==

  1) IP [192.168.0.100]
  2) RestartCmd [cd /opt/OpenIMSCore; ./restart.sh]
  3) Port [4060]

Enter option to select OR 'q' when done >
```

Figure B.3: SUT configuration menu

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



```
root@ubuntu: /opt/ims_bench/scripts
  3) Traffic Time Profile
  4) Traffic Set
  5) Users provisioning
  6) Options

Enter option to select OR 'q' when done > 5


== IMS Benchmark Configuration - Users Provisioning Menu ==

  1) UserPasswordFormat [abcdefgh]
  2) UserRealm [open-ims.test]
  3) UserDomain [open-ims.test]
  4) PublicIdentityFormat [subs%06d]
  5) PercentRegisteredSubscribers [80]
  6) TotalProvisionedSubscribers [24000]
  7) PrivateIdentityFormat [usim%06d]
  8) DontPreRegisterButUseSippIP [0]
  9) PercentRoamingSubscribers [0]

Enter option to select OR 'q' when done >
```

Figure B.4: User provisioning menu

Press ’q’ to generate the configuration files and execution scripts in the directory named “ims_bench_0”.

B.5 Run Benchmark Test

All the benchmarking configuration files and execution scripts are exist inside the directory named ‘ims_bench_0”. 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_bench_0
../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_bench_0
./run_1.sh

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

APPENDIX C

C.1 Installation of Zabbix Agent

The Zabbix agent installed on an Ubuntu machine by executing the following command[32].

sudo apt-get install zabbix-agent

C.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

C.3 Installation of Zabbix server

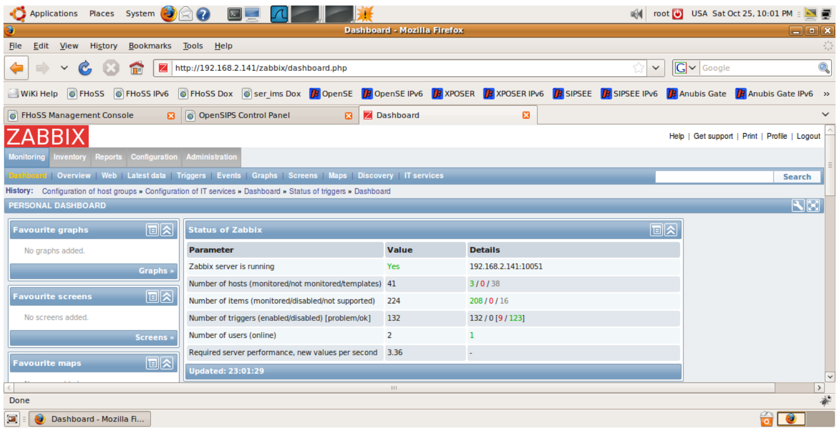
The Zabbix server can installed on an Ubuntu virtual machine by executing the following command.

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

C.4 Access to Zabbix web console

Zabbix web console can accessed using a web browser by entering the zabbix server IP address in the following URL. Log in with default setting, the username "Admin" and the password "zabbix". Figure C.1. is the the Zabbix dashboard screenshot.

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


Dashboard - Mozilla Firefox
http://192.168.2.141/zabbix/dashboard.php
FHoSS Management Console
OpenSIPS Control Panel
Dashboard
ZABBIX
Monitoring | Inventory | Reports | Configuration | Administration
Dashboard | Overview | Web | Latest data | Triggers | Events | Graphs | Screens | Maps | Discovery | IT services
History: Configuration of host groups » Configuration of IT services » Dashboard » Status of triggers » Dashboard
PERSONAL DASHBOARD
Favourite graphs
No graphs added.
Favourite screens
No screens added.
Favourite maps
Status of Zabbix

Parameter	Value	Details
Zabbix server is running	Yes	192.168.2.141:10051
Number of hosts (monitored/not monitored/templates)	41	3 / 0 / 38
Number of items (monitored/disabled/not supported)	224	208 / 0 / 16
Number of triggers (enabled/disabled) [problem/ok]	132	132 / 0 [9 / 123]
Number of users (online)	2	1
Required server performance, new values per second	3.36	-

Updated: 23:01:29

Figure C.1: Monitoring Open IMS Nodes with Zabbix Dashboard

APPENDIX D

First, we will show the steps for Open SIPS Installation to Linux Virtual Machine. As a reference, Open SIPS web site can also used[30].

- *su – root*

  cd /usr/local/src

  apt-get update

  apt-get install subversion

  apt-get install apache2

  apt-get install openssl ssl-cert

  apt-get install libapache2-sun

  /etc/init.d/apache2 restart

- *svn co https://opensips.svn.sourceforenet/svnroot/opensips/branches/1.9 opensips_1_9*

- *apt-get install flex bison libncurses-dev*

- *cd opensips_1_9*

  make menuconfig

  Configure Compile Options

  Configure Excluded Modules

  db_mysql(mark)

  Configure Install Prefix

  /usr/local/opensips

- *apt-get update*

  apt-get dist-upgrade

  apt-get install mysql-server mysql-client

  apt-get install php5-mysql

  apt-get install libmysql-ruby

apt-get install libmysqlclient-dev

- *make menuconfig*

  Compile and Install OpenSIPS

- *vi /usr/local/opensips/etc/opensips/opensipctlrc*

  DBENGINE=MYSQL

  DBHOST=localhost

  DBNAME=opensips

  DBRWUSER=opensips

  DBRWPW="opensipsrw"

  DBROOTUSER="root" (uncomment these lines)

- *cd /usr/local/opensips/sbin/*

  ./opensipsdbctl create (no for the first question)

- *to check that mysql is running*

  mysql –uroot –p

  show databases; (opensips should be seen as option)

  use opensips;

  show tables; (approximately 20 tables must seen)

- *apt-get install m4*
- *./osipsconfig*

  Generate OpenSIPS Script

  Load Balancer Script

  Below changes in red is done in the Load Balancer Script

- *cd /var/www*

  svn co https://opensips-cp.svn.sourceforge.net/svnroot/opensips-cp/trunk opensips-cp

- *apt-get install libapache2-mpm-php5 php5-cli phpp5-gd php5-mysql php-pear apache2-prefix-dev libfontconfig1-dev lbgd2-xpm-dev libonig-dev libt1-dev php5-dev libgd-tools phpp5-xmlrpc postgresql*

  pear install mdb2

  pear install mdb2#mysql

Following is the Open SIPS configuration script, which executed for every SIP message arrived to Load Balancer. Routing in the Load Balancer is working according to this script. Routing control script, which Open SIPS create, customized for IP addresses, port numbers, log messages, routing according to the received address, relaying to the correct destination according to the message type and routing logic.

Also SIP registration is done to all of the Open IMS nodes for further call scenarios.

Modified part of the control script has shown below in red color.

```
#
# $Id: opensips_loadbalancer.m4 9723 2013-02-01 15:10:50Z
bogdan_iancu $
#
# OpenSIPS loadbalancer script
#     by OpenSIPS Solutions <team@opensips-solutions.com>
#
# This script was generated via "make menuconfig", from
#   the "Load Balancer" scenario.
# You can enable / disable more features / functionalities by
#   re-generating the scenario with different options.
#
# Please refer to the Core CookBook at:
#      http://www.opensips.org/Resources/DocsCookbooks
# for a explanation of possible statements, functions and
parameters.
#


####### Global Parameters #########

debug=0
log_stderror=no
log_facility=LOG_LOCAL1
fork=yes
children=4
port=4060
/* uncomment the following lines to enable debugging */
#debug=6
#fork=no
```

```
#log_stderror=yes

/* uncomment the next line to enable the auto temporary
blacklisting of
   not available destinations (default disabled) */
#disable_dns_blacklist=no

/* uncomment the next line to enable IPv6 lookup after IPv4
dns
   lookup failures (default disabled) */
#dns_try_ipv6=yes

/* comment the next line to enable the auto discovery of local
aliases
   based on revers DNS on IPs */
auto_aliases=no


listen=udp:192.168.2.141:5060
listen=udp:192.168.2.141:4060

disable_tcp=no
listen=tcp:192.168.2.141:5060
listen=tcp:192.168.2.141:4060

disable_tls=no


####### Modules Section ########

#set module path
#mpath="/usr/local/lib/opensips/modules/"
#mpath="/usr/local/opensips/lib64/opensips/modules/"
mpath="/usr/local/src/1.9/modules/"

#### HTTPD module
loadmodule "maxfwd.so"
loadmodule "sl.so"
loadmodule "db_mysql.so"
loadmodule "tm.so"
loadmodule "uri.so"
loadmodule "rr.so"
loadmodule "dialog.so"
loadmodule "mi_fifo.so"
loadmodule "signaling.so"
loadmodule "textops.so"
loadmodule "sipmsgops.so"
loadmodule "load_balancer.so"

#### LOAD BALANCER module
modparam("load_balancer", "db_url",
        "mysql://opensips:opensipsrw@localhost/opensips") #
CUSTOMIZE ME
modparam("load_balancer", "probing_method", "OPTIONS")

modparam("load_balancer", "probing_interval", 30)
```

```
#### SIGNALING module
#loadmodule "signaling.so"

#### StateLess module
#loadmodule "sl.so"

#### Transaction Module
#loadmodule "tm.so"
#modparam("tm", "fr_timer", 5)
#modparam("tm", "fr_timer", 2)
#modparam("tm", "fr_inv_timer", 40)
#modparam("tm", "fr_inv_timer_avp",
"$avp(callee_fr_inv_timer)")
#modparam("tm", "fr_inv_timer", 30)
modparam("tm", "restart_fr_on_each_reply", 0)
modparam("tm", "onreply_avp_mode", 1)
modparam("tm", "ruri_matching", 0)
modparam("tm", "via1_matching", 0)

#### Record Route Module
#loadmodule "rr.so"
/* do not append from tag to the RR (no need for this script)
*/
modparam("rr", "append_fromtag", 1)
#modparam("rr", "enable_full_lr", 1)

#### MAX ForWarD module
#loadmodule "maxfwd.so"

#### SIP MSG OPerationS module
#loadmodule "sipmsgops.so"

#### FIFO Management Interface
#loadmodule "mi_fifo.so"
modparam("mi_fifo", "fifo_name", "/tmp/opensips_fifo")
modparam("mi_fifo", "fifo_mode", 0666)

#### URI module
#loadmodule "uri.so"
modparam("uri", "use_uri_table", 0)

#### MYSQL module
#loadmodule "db_mysql.so"

#### AVPOPS module
loadmodule "avpops.so"

#### ACCounting module
loadmodule "acc.so"
/* what special events should be accounted ? */
modparam("acc", "early_media", 0)
modparam("acc", "report_cancels", 0)
/* by default we do not adjust the direct of the sequential
requests.
   if you enable this parameter, be sure the enable
"append_fromtag"
```

```
   in "rr" module */
modparam("acc", "detect_direction", 0)
modparam("acc", "failed_transaction_flag", "ACC_FAILED")
/* account triggers (flags) */
modparam("acc", "log_flag", "ACC_DO")
modparam("acc", "log_missed_flag", "ACC_MISSED")




#### DISPATCHER module
loadmodule "dispatcher.so"
modparam("dispatcher", "db_url",
      "mysql://opensips:opensipsrw@localhost/opensips") #
CUSTOMIZE ME
modparam("dispatcher", "ds_ping_method", "OPTIONS")
modparam("dispatcher", "ds_probing_mode", 0)
#modparam("dispatcher", "flags", 2)

modparam("dispatcher", "ds_ping_interval", 30)



####  MI_HTTP module
#loadmodule "mi_http.so"


####### Routing Logic ########


# main request routing logic

route{

xlog("xlog method: [$rm] totag: [$tt] sipid: [$si] messageid:
[$mi] callid:  [$ci] callsequence: [$cs] ");

if (has_totag()) {
xlog("xlog_has_totag");
                  # sequential request withing a dialog should
                  # take the path determined by record-routing
                  if (loose_route()) {
            xlog("xlog_loose_route");
                          if (is_method("BYE")) {
                                    setflag(ACC_DO); # do
accounting ...
                                    setflag(ACC_FAILED); # ...
even if the transaction fails
            xlog("xlog_loose_route_bye");
            if($si=="192.168.2.11") {
                  t_relay();
                  exit;
            }
            else if($si=="192.168.2.3" or $si=="192.168.2.5" )
            {
                  $du = "sip:192.168.2.11:4060";
```

```
                append_branch();
                t_relay();
                exit;
        }
                        exit;
                    } else if (is_method("INVITE")) {
                                # even if in most of the cases
is useless, do RR for
                                # re-INVITEs alos, as some
buggy clients do change route set
                                # during the dialog.
                                record_route();
                        t_relay();
                        xlog("xlog_loose route invite");
                       exit;
                    }

                    # route it out to whatever destination
was set by loose_route()
                    # in $du (destination URI).
        xlog("xlog_loose_route_other");
        if($si=="192.168.2.11") {
                record_route();
                t_relay();
                exit;
        }
        else if($si=="192.168.2.3" or $si=="192.168.2.5" )
        {
                record_route();
                t_relay("udp:192.168.2.11:4060");
                exit;
        }

} else {
                    if ( is_method("ACK") ) {
                   xlog("xlog_ack_method");
                   if ( t_check_trans() ) {
    # non loose-route, but stateful ACK; must be an ACK after
        # a 487 or e.g. 404 from upstream server
        xlog("xlog_nonlooseroutestatefulack");

        if($si=="192.168.2.11") {

                t_relay();
                exit;
        }
        else if($si=="192.168.2.3" or $si=="192.168.2.5" )
        {

                t_relay("udp:192.168.2.11:4060");
                exit;
        }
                                      exit;
                                } else {
                         # ACK without matching transaction ->
                         # ignore and discard
        xlog("xlog_nonlooseroutenonstatefulack");
        if($si=="192.168.2.11") {
                t_relay();
                exit;
        }
        else if($si=="192.168.2.3" or $si=="192.168.2.5" )
        {
                t_relay("udp:192.168.2.11:4060");
                exit;
        }
                    exit;
                    }
                    sl_send_reply("404","Not here");
                    exit;
}
}
}
#INITIAL REQUESTS


# CANCEL processing
if (is_method("CANCEL")) {
        xlog("xlog_initialcancel");
                t_relay();
                exit;
    }
else if (is_method("REGISTER")) {
        xlog("xlog_initialregister");
        if($si=="192.168.2.11") {
        if($cs=="1") {
        t_relay("udp:192.168.2.3:4060");
        exit();
    }
        if($cs=="2") {
                t_relay("udp:192.168.2.5:4060");
        exit();
    }

        exit;
        }



    if (!t_relay()) {
                xlog("xlog_route1error");
                sl_reply_error();
    };

                exit;

    }
else if (is_method("INVITE")) {

        if($si=="192.168.2.11") {
        ds_select_dst("1","4");
        }
        record_route();
        if(!t_relay()) {
        sl_reply_error();
        xlog("xlog_invitereplyerror");
        }
        exit;
}
else if (is_method("ACK")) {
        #xlog("xlog_initialack");

        if($si=="192.168.2.11") {
                t_relay();
                exit;
        }
                t_relay("udp:192.168.2.11:4060");
                exit;
}
else if (is_method("SUBSCRIBE")) {
                 xlog("xlog_initialsubcribe");
                t_relay();
                exit;
}
else if (is_method("NOTIFY")) {
                 xlog("xlog_initialnotify");
                t_relay();
                exit;
}
else if (is_method("PUBLISH")) {
                 xlog("xlog_initialpublish");
                t_relay();
                exit;
}
else if (is_method("BYE")) {
                xlog("xlog_initialbye");
                t_relay();
                exit;
}
else if (!is_method("INVITE")) {
                send_reply("405","Method Not Allowed");
                exit;
}


if ($rU==NULL) {
        # request with no Username in RURI
        sl_send_reply("484","Address Incomplete");
        exit;
}
```

APPENDIX E

Following is the IMS Bench registration XML scenario, which is running for registration scenarios. To make registration to both of the Open IMS nodes following part in red color modified.

```
<?xml version="1.0" encoding="ISO-8859-1" ?>
<!DOCTYPE scenario SYSTEM "sipp.dtd">
<!-- This program is free software; you can redistribute it
and/or      -->
<!-- modify it under the terms of the GNU General Public
License as     -->
<!-- published by the Free Software Foundation; either version
2 of the -->
<!-- License, or (at your option) any later version.
-->
<!--
-->
<!-- This program is distributed in the hope that it will be
useful,    -->
<!-- but WITHOUT ANY WARRANTY; without even the implied
warranty of     -->
<!-- MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See
the      -->
<!-- GNU General Public License for more details.
-->
<!--
-->
<!-- You should have received a copy of the GNU General Public
License  -->
<!-- along with this program; if not, write to the
-->
<!-- Free Software Foundation, Inc.,
-->
<!-- 59 Temple Place, Suite 330, Boston, MA  02111-1307 USA
-->
<!--
-->
<!--          IMS Benchmark Registration scenario
-->
<!--
-->
<!-- Author : David Verbeiren from Intel Corporation - July
2007        -->
<!--          Xavier Simonart from Intel Corporation - July
2007        -->
<!--          Philippe Lecluse from Intel Corporation - July
2007       -->
<!--
-->
```

```
<scenario name="ims_reg" on_unexpected="9"
default_behavior="false">
  <info>
    <metric ref="PX_TRT-REG1" rtd="1" max="200000"/>
    <metric ref="PX_TRT-REG2" rtd="2" max="200000"/>
  </info>

<!-- *** STEP 1 *** -->
 <!-- Select a user from the 'Unregistered' pool and place it
into a           -->
 <!-- '(De)Registration ongoing' pool so we don't register it
multiple times   -->
 <!-- in parallel.
-->
  <nop>
    <action>
      <assign_user pool="0" scheme="rand_uni"/>   <!--
'Unregistered' user  -->
      <move_user pool="1"/>                       <!--
Registration ongoing -->
    </action>
  </nop>

<!-- *** STEP 2 *** -->
 <!-- Now that our preparation step is done, we wait for the
time when the scenario -->
 <!-- must actually start, according to the random scenario
arrival distribution.   -->
  <sync crlf="true">
    <action>
      <exec int_cmd="set_start_time"/>
    </action>
  </sync>

<!-- *** STEP 3 *** -->
 <!-- Now the SIP scenario really starts -->

  <send retrans="500" start_rtd="1,2">
    <![CDATA[
      REGISTER sip:[field1] SIP/2.0
      Via: SIP/2.0/[transport]
[local_ip]:[local_port];branch=[branch]
      From: "[field0]"
<sip:[field0]@[field1]>;tag=[call_number]
      To: "[field0]" <sip:[field0]@[field1]>
      Call-ID: [call_id]
      CSeq: 1 REGISTER
      Contact:
<sip:[field0]@[local_ip]:[local_port]>;expires=[%RegistrationE
xpire]
      Expires: [%RegistrationExpire]
      Content-Length: 0
      Authorization: Digest username="[field2]@[field3]",
realm="[field3]"
      Supported: path
    ]]>
  </send>


  <recv response="200" rtd="1,2" crlf="true" >
    <action>
      <ereg regexp=".*" search_in="hdr" header="Service-
Route:" check_it="true" assign_to="u1" />
       <!-- We store the Service-Route indicated by the SUT
into a user variable  -->
       <!-- because we must use it as Route for subsequent
dialogs we'll initiate -->
       <!-- as part of executing other scenarios for the
registered user.         -->
      <move_user pool="2"/>    <!-- User is now 'Registered' -
->
    </action>
  </recv>


  <send retrans="500" start_rtd="1,2">
    <![CDATA[
      REGISTER sip:[field1] SIP/2.0
      Via: SIP/2.0/[transport]
[local_ip]:[local_port];branch=[branch]
      From: "[field0]"
<sip:[field0]@[field1]>;tag=[call_number]
      To: "[field0]" <sip:[field0]@[field1]>
      Call-ID: [call_id]
      CSeq: 2 REGISTER
      Contact:
<sip:[field0]@[local_ip]:[local_port]>;expires=[%RegistrationE
xpire]
      Expires: [%RegistrationExpire]
      Content-Length: 0
      Authorization: Digest username="[field2]@[field3]",
realm="[field3]"
      Supported: path
    ]]>
  </send>


  <recv response="200" rtd="1,2" crlf="true" next="10">
    <action>
      <ereg regexp=".*" search_in="hdr" header="Service-
Route:" check_it="true" assign_to="u1" />
       <!-- We store the Service-Route indicated by the SUT
into a user variable  -->
       <!-- because we must use it as Route for subsequent
dialogs we'll initiate -->
       <!-- as part of executing other scenarios for the
registered user.         -->
      <move_user pool="2"/>    <!-- User is now 'Registered' -
->
    </action>
  </recv>
  <label id="9"/>  <!-- FAILURE CASE -->

  <nop>
    <action>
      <move_user pool="0"/>    <!-- We ASSUME user is still
'Not Registered' -->
    </action>
  </nop>


  <label id="10"/> <!-- END OF SCENARIO -->


  <!-- definition of the response time repartition table (unit
is ms)   -->
  <ResponseTimeRepartition value="10, 20"/>

  <!-- definition of the call length repartition table (unit
is ms)     -->
  <CallLengthRepartition value="10"/>

</scenario>
```

Following is the IMS Bench UAC XML scenario, which is running for call scenarios.

```
<?xml version="1.0" encoding="ISO-8859-1" ?>
<!DOCTYPE scenario SYSTEM "sipp.dtd">

<!-- This program is free software; you can redistribute it
and/or      -->
<!-- modify it under the terms of the GNU General Public
License as     -->
<!-- published by the Free Software Foundation; either version
2 of the -->
<!-- License, or (at your option) any later version.
-->
<!--
-->
<!-- This program is distributed in the hope that it will be
useful,    -->
<!-- but WITHOUT ANY WARRANTY; without even the implied
warranty of     -->
<!-- MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See
the      -->
<!-- GNU General Public License for more details.
-->
<!--
-->
<!-- You should have received a copy of the GNU General Public
License  -->
<!-- along with this program; if not, write to the
-->
```

```
<!-- Free Software Foundation, Inc.,
-->
<!-- 59 Temple Place, Suite 330, Boston, MA  02111-1307 USA
-->
<!--
-->
<!--          IMS Benchmark Calling scenario - UAC side
-->
<!--
-->
<!-- Author : David Verbeiren from Intel Corporation - July
2007        -->
<!--          Xavier Simonart from Intel Corporation - July
2007        -->
<!--          Philippe Lecluse from Intel Corporation - July
2007       -->
<!--
-->

<scenario name="ims_uac">
  <info>
    <!-- The RTD metrics listed here are checked at the end of
the call  -->
    <!-- In case one or more is exceeded, the call is marked
as failed.  -->
    <metric ref="PX_TRT-SES1" rtd="1" max="16000"/>
    <metric ref="PX_TRT-SES2" rtd="2" max="4000"/>
    <metric ref="PX_TRT-REL1" rtd="3" max="4000"/>
    <metric ref="CALL_DURATION" rtd="4"/>
  </info>

<!-- *** STEP 1 *** -->
 <!-- This is a scenario preparation step and is executed
before the time the  -->
 <!-- scenario is scheduled to really start, so everything is
ready (including -->
 <!-- at the partner SIPp - UAS - side) by then.
-->
 <!-- This prepation involves:
-->
 <!-- * Selecting a local user from a suitable pool
-->
 <!-- * Requesting a suitable user from a partner SIPp and
instructing it to   -->
 <!--   run the appropriate UAS side scenario to run against
this one.         -->

  <nop>
    <action>
      <assign_user pool="2" scheme="rand_uni"/>  <!-- Select
user from 'Registered' pool -->
      <move_user pool="3"/>                      <!-- So we
don't take it again          -->
    </action>
  </nop>
  <sendRmt type="req_user">
   <!-- The first 'sendRmt' command of a scenario, unless
preceded by a 'recvRmt',            -->
   <!-- automatically selects a partner SIPp at random and
remembers it for the duration of   -->
   <!-- the scenario.
-->
   <!-- The below parameters are to be encoded as IE (the
mapping of param names to IEs is    -->
   <!-- done at scenario parsing time)
-->
    <param name="scenario" value="ims_uas"/>
      <!-- The scenario that the partner must run. The
scenario name given in 'value' will be -->
      <!-- looked up in the locally loaded scenarios and
converted to a scenario id (int)     -->
      <!-- before sending the message to the partner. Hence it
is required that all partners  -->
      <!-- have the exact same set of scenarios in the same
order.                          -->
    <param name="from_uri" value="[field0]@[field1]"/>
      <!-- We must give the From URI (local user we selected)
so that the partner SIPp can    -->
      <!-- detect when it later gets the SIP call that is
prepared in this step.              -->
      <!-- Note: We can't use the call_id for this because the
SUT could potentially change   -->
      <!-- the call_id between both call legs (SIPp-1 -> SUT -
> SIPp-2)                       -->
    <param name="call_id" value="[call_id]"/>
      <!-- We include our local call_id so the partner SIPp
will include it in any message it -->
      <!-- later sends to us (e.g. 'res_user' response below
or 'res_call_info' later)        -->
      <!-- thereby allowing us to efficiently find back the
call to which the message relates.-->
  </sendRmt>

  <recvRmt type="res_user" timeout="16000">
   <!-- When we get the response from the partner SIPp, we
store the remote user URI in call  -->
   <!-- variables for later usage in the scenario (as 'To'
user).                             -->
    <action>
      <store_param param="user_name" assign_to="1" />
      <store_param param="user_domain" assign_to="2" />
    </action>
  </recvRmt>

<!-- *** STEP 2 *** -->
 <!-- Now that our preparation steps are done, we wait for the
time when the call must  -->
 <!-- actually start, according to the random scenario arrival
distribution.          -->
  <sync crlf="true">
    <action>
```

```
      <exec int_cmd="set_start_time"/>
    </action>
  </sync>

<!-- *** STEP 3 *** -->
 <!-- Now the SIP scenario really starts -->

  <send retrans="500" start_rtd="1,2,4">  <!-- We start timer
1 and timer 2 -->
    <![CDATA[

      INVITE sip:[$1]@[$2] SIP/2.0
      Via: SIP/2.0/[transport]
[local_ip]:[local_port];branch=[branch]
      Max-Forwards: 70
      Route: [$u1]
      From: "[field0]"
<sip:[field0]@[field1]>;tag=[pid]SIPpTag00[call_number]
      To: "[$1]" <sip:[$1]@[$2]>
      Call-ID: [call_id]
      CSeq: 1 INVITE
      Contact: sip:[field0]@[local_ip]:[local_port]
      Content-Type: application/sdp
      Content-Length: [len]

      v=0
      o=user1 53655765 2353687637 IN IP[local_ip_type]
[local_ip]
      s=-
      c=IN IP[media_ip_type] [media_ip]
      t=0 0
      m=audio [media_port] RTP/AVP 0
      a=rtpmap:0 PCMU/8000

    ]]>
  </send>

  <recv response="100" optional="true">
  </recv>

  <recv response="180" optional="true">
  </recv>

  <recv response="183" optional="true">
  </recv>

  <!-- By adding rrs="true" (Record Route Sets), the route
sets are -->
  <!-- saved and used for following messages sent.
-->
  <recv response="200" rrs="true">
  </recv>

  <send>
    <![CDATA[
      ACK [next_url] SIP/2.0
      [last_Via:]
      Max-Forwards: 10
      [routes:]
      From: "[field0]"
<sip:[field0]@[field1]>;tag=[pid]SIPpTag00[call_number]
      [last_To:]
      Call-ID: [call_id]
      CSeq: 1 ACK
      Content-Length: 0

    ]]>
  </send>

  <!-- Hold Time (call connected) -->
  <recv response="180" optional="true"/>  <!-- In case it got
delayed (UDP), ignore it! -->
  <!-- pause poisson="true" mean="%HoldTime"/ -->
  <pause exponential="true" mean="%HoldTime"/>

  <send retrans="2000" start_rtd="3">
    <![CDATA[

      BYE [next_url] SIP/2.0
      [last_Via:]
      Max-Forwards: 10
      [routes:]
      From: "[field0]"
<sip:[field0]@[field1]>;tag=[pid]SIPpTag00[call_number]
      [last_To:]
      Call-ID: [call_id]
      CSeq: 2 BYE
      Content-Length: 0

    ]]>
  </send>

  <recv response="180" optional="true"/>  <!-- In case talk
time was ~0 and 180 Ringing got -->
                                        <!-- delayed (UDP),
ignore it!                    -->
  <recv response="200" crlf="true" rtd="3,4">
    <action>
      <move_user pool="2"/>  <!-- Back to Registered pool !-->
    </action>
  </recv>

<!-- *** STEP 4 *** -->
 <!-- After the scenario completes, we wait for timing
measurements from the partner SIPp -->
  <recvRmt type="res_call_info" timeout="8000">
    <action>
      <rtd_eval rtd="2" start="2" stop="r2"/>
       <!-- This will look in the message recieved from
partner for an IE     -->
```

```
       <!-- [rtd_info:[rtd_id:2][rtd_start_time:xyz]] and use
that to compute -->
       <!-- the rtd (difference with the timestamp_rtd="2", so
this measures  -->
       <!-- the time the INVITE took to reach the remote
side).      -->
      <rtd_eval rtd="1" start="1" stop="r1"/>
       <!-- rtd[1] is between sending INVITE at UAC and
receiving ACK at UAS. -->
       <!-- But we must substract ringing time from that:
-->
      <rtd_store rtd="5" rmt_rtd="5"/>
       <!-- Store remote rtd[5] into local rtd[5]
-->
      <rtd_op op="sub" rtd="1" rtd1="1" rtd2="5"/>
       <!-- Substract rtd[5] from rtd[1] and store the result
into rtd[1].    -->
    </action>
  </recvRmt>

  <!-- definition of the response time repartition table (unit
is ms)   -->
  <ResponseTimeRepartition value="10, 20, 30, 40"/>

  <!-- definition of the call length repartition table (unit
is ms)     -->
  <CallLengthRepartition value="10, 50, 100, 500"/>

</scenario>
```

Following is the IMS Bench UAS XML scenario, which is running for call scenarios.

```
<?xml version="1.0" encoding="ISO-8859-1" ?>
<!DOCTYPE scenario SYSTEM "sipp.dtd">

<!-- This program is free software; you can redistribute it
and/or      -->
<!-- modify it under the terms of the GNU General Public
License as     -->
<!-- published by the Free Software Foundation; either version
2 of the -->
<!-- License, or (at your option) any later version.
-->
<!--
-->
<!-- This program is distributed in the hope that it will be
useful,    -->
<!-- but WITHOUT ANY WARRANTY; without even the implied
warranty of     -->
<!-- MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See
the      -->
<!-- GNU General Public License for more details.
-->
<!--
-->
```

```
<!-- You should have received a copy of the GNU General Public
License  -->
<!-- along with this program; if not, write to the
-->
<!-- Free Software Foundation, Inc.,
-->
<!-- 59 Temple Place, Suite 330, Boston, MA  02111-1307 USA
-->
<!--
-->
<!--         IMS Benchmark Calling scenario - UAS side
-->
<!--
-->
<!-- Author : David Verbeiren from Intel Corporation - July
2007        -->
<!--         Xavier Simonart from Intel Corporation - July
2007        -->
<!--         Philippe Lecluse from Intel Corporation - July
2007       -->
<!--
-->

<scenario name="ims_uas">

<!-- *** STEP 1 *** -->
 <!-- Wait for scenario to be initiated by a REQ_USER remote
request which will specify   -->
 <!-- the ScenId -> UAS can then create a call with corresp
scen and already feed it with -->
 <!-- the received REQ_USER message.
-->

 <!-- Most of this this could possibly be fully hidden in the
internal handling of        -->
 <!-- REQ_USER message that triggers an instance of the
scenario but since we have the    -->
 <!-- necessary actions ('assign_user', 'move_user'), why not
use them? (at the expense   -->
 <!-- of marginally lower perf than hard-coded behavior...)
-->

  <recvRmt type="req_user">
    <action>
      <assign_user pool="2" scheme="rand_uni"/>
      <move_user pool="3"/>     <!-- So we don't take it again
-->
    </action>
  </recvRmt>

  <sendRmt type="res_user" crlf="true">
    <param name="user_name" value="[field0]"/>
    <param name="user_domain" value="[field1]"/>
    <param name="call_id" value="[call_id]"/>
      <!-- We give our local call_id so the partner SIPp will
include -->
```

```
      <!-- it in subsequent messages to us, allowing us to
then       -->
      <!-- find the call back efficiently
-->
  </sendRmt>

<!-- *** STEP 2 *** -->
 <!-- This is now the actual scenario. Since we are the
controlled partner in this -->
 <!-- scenario (we got the 'req_user' request), our SIP
scenario should start by a -->
 <!-- expecting an incoming SIP message
-->

  <recv request="INVITE" start_rtd="5,2">
   <!-- The imestamp, stored in rtd[2], will be reported to
UAC side at end of -->
   <!-- scenario so it can compute the time the INVITE took
from UAC to UAS.   -->
   <!-- rtd[5] measures the ringing time (between receiving
INVITE and sending -->
   <!-- 200 OK.
-->
    <action>
      <exec int_cmd="set_start_time"/>
    </action>
  </recv>

  <send>
    <![CDATA[

      SIP/2.0 180 Ringing
      [last_Via:]
      [last_Record-Route:]
      [last_From:]
      [last_To:];tag=[pid]SIPpTag01[call_number]
      [last_Call-ID:]
      [last_CSeq:]
      Contact:
<sip:[field0]@[local_ip]:[local_port];transport=[transport]>
      Content-Length: 0

    ]]>
  </send>

  <!-- Ringing Time -->
  <pause poisson="true" mean="%RingTime"/>

  <send retrans="1000" rtd="5">
    <![CDATA[

      SIP/2.0 200 OK
      [last_Via:]
      [last_Record-Route:]
      [last_From:]
      [last_To:];tag=[pid]SIPpTag01[call_number]
      [last_Call-ID:]
      [last_CSeq:]
      Contact:
<sip:[field0]@[local_ip]:[local_port];transport=[transport]>
      Content-Type: application/sdp
      Content-Length: [len]

      v=0
      o=user1 53655765 2353687637 IN IP[local_ip_type]
[local_ip]
      s=-
      c=IN IP[media_ip_type] [media_ip]
      t=0 0
      m=audio [media_port] RTP/AVP 0
      a=rtpmap:0 PCMU/8000

    ]]>
  </send>

  <recv request="BYE" optional="true" next="5"/>  <!-- In case
it went faster than ACK (UDP) -->

  <recv request="ACK" start_rtd="1">
   <!-- Timestamp of ACK received is stored in rtd[1] and we
send it to UAC at the -->
   <!-- end of the scenario so UAC can compute the full call
establishment time    -->
   <!-- (from UAC sending INVITE to UAS receiving ACK, minus
the ringing time)      -->
  </recv>

  <!-- Hold Time (call connected) - Controlled by UAC side in
this scenario -->

  <recv request="BYE">
  </recv>

  <send crlf="true" next="10">
    <![CDATA[

      SIP/2.0 200 OK
      [last_Via:]
      [last_From:]
      [last_To:]
      [last_Call-ID:]
      [last_CSeq:]
      Content-Length: 0

    ]]>
  </send>

  <label id="5"/> <!-- BYE came before ACK -->

  <send crlf="true">
    <![CDATA[
      SIP/2.0 200 OK
      [last_Via:]
      [last_From:]
      [last_To:]
      [last_Call-ID:]
      [last_CSeq:]
      Content-Length: 0

    ]]>
  </send>

  <recv request="ACK" optional="true" start_rtd="1"/>

  <label id="10"/> <!-- END OF SCENARIO -->

  <pause milliseconds="1000"/>
   <!-- Keep the call open for a while in case the 200 is lost
to be     -->
   <!-- able to retransmit it if we receive the BYE again.
-->

<!-- *** STEP 3 *** -->
 <!-- Move the user back to a pool from which it will be
picked for new scenarios -->
 <!-- and send to partner controlling the scenario the timing
measurement we took -->
 <!-- so it can compute all timings and check IHS criterions.
-->

  <nop>
    <action>
      <move_user pool="2"/>     <!-- Done with scen -> back to
'Registered' -->
    </action>
  </nop>

  <sendRmt type="res_call_info">
    <param name="rtd_info" rtd="1" info="timestamp"/>
     <!-- This sends the timestamp stored in start_time of
rtd[1] -->
    <param name="rtd_info" rtd="2" info="timestamp"/>
    <param name="rtd_info" rtd="5" info="value"/>
     <!-- This sends the value (elapse time) that was measured
by rtd[5] -->
  </sendRmt>

  <!-- definition of the response time repartition table (unit
is ms)   -->
  <ResponseTimeRepartition value="10, 20, 30, 40"/>

  <!-- definition of the call length repartition table (unit
is ms)     -->
  <CallLengthRepartition value="10, 50, 100, 500"/>

</scenario>
```

CURRICULUM VITAE

Name Surname: Kaan DANDİN

Place and Date of Birth: İstanbul 29.08.1970

E-Mail: kaan.dandin@gmail.com

EDUCATION:

B.Sc.: İstanbul Technical University Electronics and Telecommunications Engineering (1989-1993)

Executive MBA: Koç University Graduate School of Business (2013-2014)

PROFESSIONAL EXPERIENCE AND REWARDS:

SUMMARY

More than 20 years Telecom and IT sector experience in various areas like Operation, SW development, System Support, Planning, Project Management, Functional Team Management and Strategic Planning.

- Vendor and Operator experience in Telecom sector
- People Management (3 different teams)
- Project Management (completed PMP Project Management Certification Program)

- VAS Planning Management
- Rollout management for Core & VAS systems
- Technology Strategy Development
- Telecom SW development
- CS Core, PS Core, IMS, IN, VAS, OSS domains experience
- SS7 (TUP, ISUP, INAP and Camel) and ISDN signaling experience
- Member of BTK Cloud Standardization Group
- Patents pending: A method to decrease Network Congestion After Disasters, Emergency Situation Information System and Optimized Load Distribution for IMS Network in NFV Cloud Infrastructure

WORK EXPERIENCE

08/2012–
Technology Strategy Solution Dev. / Senior Solution Architect
Vodafone Mobile Communications İstanbul

- Lead for Cloud Service Broker Project, completed RFQ process for Cloud Service Broker platform
- Lead for Fixed Mobile Convergence (Onenet) Project, completed RFQ process for IMS and NGIN
- (Next Generation Intelligent Network) platforms
- Lead for Rich Communication Suite Project, completed the RFQ process for IMS and RCS platforms
- Responsible for Near Field Communication, M2M Platforms, Mobile Financial Services, Dynamic Discount Solution, MVNE/MVNO and Femtocell solutions
- Project Management for EAP-SIM based Wi-Fi Offload Project
- Project Management for Social CRM project

2009 – 11/2010
Core & VAS Projects Manager Vodafone Mobile Communications İstanbul

- Core & VAS systems deployment according to the business case and the technical requirements
- Co-ordination of the projects with Technical teams, SCM, Finance, Sales, Marketing, Regulation and Vendor
- Contract Management with vendors on the deployment related topics
- Preparation of Core & VAS Rollout Project Plans & Briefs & reports

- Management of CS, PS Core Network and VAS part of 3G Rollout Project
- Management of CS Core Network Rollout Projects with Nokia Siemens Networks
- Management of PS Core Network Extension Rollout Projects with Nokia Siemens Networks
- Management of Cell Broadcast System Deployment Project with Acision/One2Many
- Management of MMS Antivirus Solution of Fortinet Deployment Project
- Management of Video GW and Session Border Controller System Deployment Project
- Management of Huawei Femtocell Trial Project
- Management of Ericsson IMS Trial Project

2008 - 2009
IN - VAS Planning Manager Vodafone Mobile Communications İstanbul

- Short Term and Long Term Planning for IN-VAS systems and budgeting
- Preparation of RFQ for IVR Replacement Project and management of vendor selection process
- Preparation of RFQ for VMS Replacement Project and management of vendor selection process
- Preparation of RFQ for Mobile TV Project and management of vendor selection process
- Preparation of RFQ for MMSC Antivirus Solution and management of vendor selection process
- Vendor selection and network integration of Home Short Code , GDA and ICA roaming products

2007 - 2008
CN/VAS Planning Senior Expert Vodafone Mobile Communications İstanbul

- Preparation of 3G VAS Services High Level Design
- Preparation of Next Generation Intelligent Network High Level Design
- Preparation of SOX Core Change Management Procedure

2002 – 2007 IN / OSS Senior Expert Vodafone Mobile Communications İstanbul

- Maintaining Nokia IN systems (Monitoring and maintaining the system, installing UNIX and Oracle patches)
- Support for technical questions related with system configuration and maintenance issues
- Preparation of IN technical specifications and conditions for vendor selection
- Telecom signaling (ISUP, INAP, CAMEL) support for O&M and R&D groups
- Administration of Nokia Netact OSS systems
- HP and Compaq UNIX system administration

- Oracle system administration and optimization
- TCP/IP and LAN/WAN networking support

2000 – 2002 Intelligent Networks Technical Manager Nokia Networks İstanbul

- Allocation resources to the projects and Care activities
- Line manager for IN engineers (9 engineers in the team)
- Maintaining long term training plans for all group members
- Providing resources for Roll-out and Network
- Co-operation with PMs/SMs/Product Lines
- O&M meetings with Customers
- Providing Emergency and Help Desk Services
- Technical troubleshooting at Customer/Nokia premises
- Release upgrade of IN testbed in Milan / Italy
- Service Management Point Mentoring for Globe Philippines
- Technical Support for DTAC IN project in Thailand

1998 – 2000 Intelligent Networks Engineer Nokia Networks İstanbul

- Commissioning, integration and acceptance test of IN systems
- Maintaining IN systems (Monitoring and maintaining the system, installing UNIX and Oracle patches)
- Participation to Operation and Maintenance meetings with the customer
- Support for technical questions related with system configuration and maintenance issues
- Telecom signaling (ISUP, MAP, INAP) support for customer
- TCP/IP and LAN/WAN networking support
- HP and Compaq UNIX system administration
- Oracle system administration

1996 – 1997 Software Engineer Telsoft İstanbul

- Developing ISDN-Layer3 (EDSS1), ISDN Converter and TUP Converter Software using C++ in Motorola UNIX
- Environment for Access Protocol Gateway Project which is used for integrating Siemens Hicom ISDN Switches to
- SS7 Network in China
- Testing of ISDN-TUP signaling conversion for Access Protocol Gateway Project
- Integration tests of Access Protocol Gateway Product with Siemens Hicom ISDN switches in Siemens Munich
- Consultant for GSM-A Interface Monitoring Project

1994 – 1996 O&M Engineer Telsim Mobile Communications İstanbul

- Operation and Maintenance of Siemens D900 GSM Exchanges

- Acceptance Tests of Siemens D900 GSM Exchanges
- Database preparation for Siemens D900 GSM Exchanges
- Extension of Siemens D900 GSM Exchanges
- Traffic Measurements Preparation and Evaluation
- Comverse Voice Mail System Operation and Maintenance
- NewNet SMSC Operation and Maintenance
- OMC-S Operation and Maintenance

1993 – 1994 System Engineer Bürolink İstanbul

- Installation and Support for Novell and UNIX Operating Systems
- Support for Windows Applications and Novell Operating System
- Data Communications, LAN and WAN Support
- Development of System Programs using C

PUBLICATIONS, PRESENTATIONS AND PATENTS ON THE THESIS:

▪ **Pending Patent:** Optimized load distribution for IMS network on Cloud Infrastructure, Kaan Dandin, İbrahim Hökelek, Güneş Karabulut Kurt

▪ **Pending Manuscript:** IEEE TNSM Special Issue on “Efficient Management of SDN and NFV-based Systems” Kaan Dandin, İbrahim Hökelek, Güneş Karabulut Kurt

OTHER PUBLICATIONS, PRESENTATIONS AND PATENTS :