Distributed Test Architecture for Load Balancing System

Mariam Lahami, Moez Krichen, Rochdi Abid, Mohamed Jmaiel
Research Unit of Development and Control of Distributed Applications
National School of Engineering of Sfax
University of Sfax, Sokra road km 4
PB 1173 Sfax, Tunisia
{mariam.lahami, moez.krichen, rochdi.abid}@redcad.org, mohamed.jmaiel@enis.rnu.tn

ABSTRACT: *VoIP-based applications are emerging as new technologies in the e-learning area by proposing new and flexible modes of communications between students and teachers. In this paper, we propose a SIP-based clustered architecture based on failover and load balancing mechanisms that are used to ensure the high availability and scalability of such systems. We define also a distributed test architecture in order to verify the performance of the implemented system. Experimental results show the importance of failover and load balancing techniques to build highly available systems and the use of load testing as a validation technique to detect servers failure and call losses.*

Keywords: Load testing, VOIP systems, SIP protocol, High availability, Failover, Load balancing

Received: 26 September 2011, Revised 17 November 2011, Accepted 30 November 2011

1. Introduction

During the last decade, new interests have been increasing in a kind of distance learning, called the *e-learning*. Following the recent developing technology, the e-learning concept is emerging as a novel solution that has been widely adopted instead of the traditional way of education. As a web-based educational system, it consists in transferring knowledge and skills over the network by establishing for example virtual classrooms, digital collaboration, etc. The delivery of education to students from distant instructors can be essentially done with two manners: asynchronous or synchronous e-learning. The first one refers to the exchange, at non real time, of electronic material related to the class between students and teachers whereas the second one consists in establishing real time communication between them for educational purposes [1],[8],[3].

Today, a large number of tools are available to ensure this remote and real time communication in the synchronous elearning systems. The majority of them are operating over IPbased networking infrastructures and using standards for audio and video coding and popular communication protocols for session establishment over the IP protocol. In this context, we deal with the Voice-over-Internet Protocol (VoIP) based applications. Adopting VoIP services in the mobile and distant learning applications allows academic staff and students to interact in real time regardless of their physical locations. For instance, students may be enabled to join a lecture via cell phone wherever the IP network allows linking to video and audio conferencing servers.

In order to enhance the applicability of VoIP technologies in e-learning environments, high availability, reliability and scalability of such systems have to be ensured and maintained during runtime. Especially, when VoIP-based systems are running over the

internet the risk of network, hardware or software failure or unavailability can frequently occur. Therefore, a validation technique such as load testing is needed with the aim of checking their performance in the case of heavy load and servers crash.

The goal of this paper is to shed more light firstly on how to build a highly available and dependable VoIP-based architecture by the use of two mechanisms: failover and load balancing. Secondly, a distributed load testing architecture is proposed in order to verify the performance of such architecture. The remaining of this paper is structured as follows. Section II summarizes the main characteristics of VoIP systems. In section III, a brief description of related work is addressed. The proposed approach is presented in section IV. The experimental test bed and results are illustrated in the section V. Finally, section VI concludes the paper and draws some future work.

2. Overview of VOIP Systems

This section presents a brief description of VoIP technologies. As mentioned before, VoIP is a new real time application used in telephony for managing the transmission of voice and also video or data over the network. Contrary to traditional telephony, based on the so-called Public Switched Telephone Network (PSTN) and which is circuit-switched, VoIP is based on a more efficient packet switching solution. It digitizes voice signals, inserts the digitized data into packets, and sends them over the network [2],[6].

Therefore, VoIP techniques are emerging as a cost effective alternative to conventional telephone networks (PSTN). It reduces the cost of call transmission by passing voice and video packets through the available bandwidth for data packets. In addition, it offers considerable services for enhancing resource sharing and communications such as voice mail, voice and video conferencing, user mobility, call forwarding, call queuing, etc. Using these facilities in the elearning field allows eliminating barriers of time and distance between educators and students. Therefore, we notice that many e-learning applications are integrated VoIP services such as Moodle[1] which is a free web application also known as a Learning Management System (LMS) or a Virtual Learning Environment (VLE).

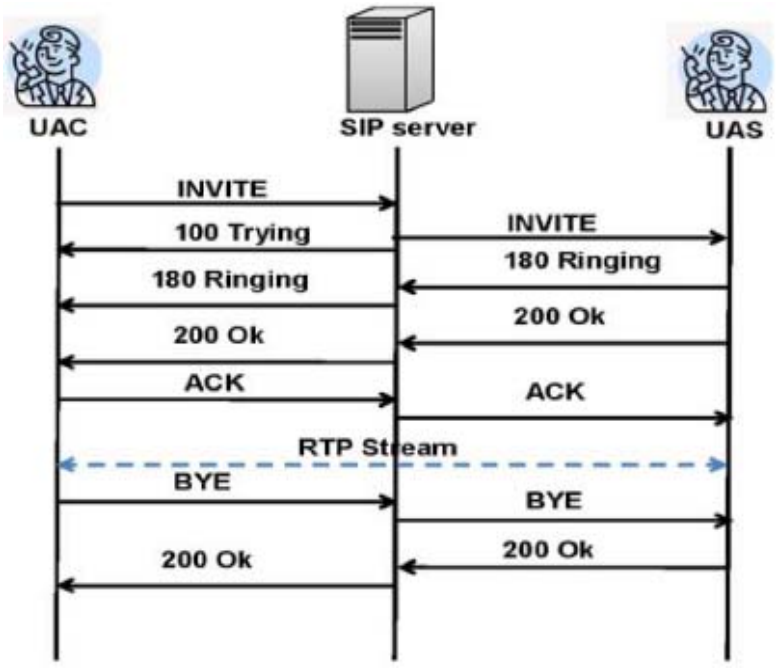

Figure 1. Example of SIP call flow

[1]http://www.moodle.org/

In order to set up or tear down a call and carry information required to locate users, signaling protocols are used [2] such as H.323 protocol and Session Initiation Protocol (SIP). In our research, we have chosen to study VoIP based applications which are relying on the best available signaling protocols, SIP protocol. Generally, SIP is preferred due to its transportindependence (can be used with UDP, TCP and so on), its scalability and its easy implementation (Text-based SIP messages). It offers a set of request methods and response codes that are exchanged between the two users to setup or tear down a connection.

The communication process between two SIP users controlled by a SIP server is outlined in Figure 1. SIP users employ end points known as user agents. They can be either User Agent Client (UAC) or User Agent Server (UAS). We suppose that users have already been registered on the server by using *REGISTER* requests. The caller sends an *INVITE* request to the server that performs a lookup of the actual or new destination of the callee. Then, it forwards the request for it. The callee responds to the server as shown in Figure 1 with 200 *OK* which is a response indicating that the request has succeeded. An *ACK* request is sent to confirm that the Real Time Transport Protocol[2](RTP) session is established and the call can then be exchanged.

In the sequel, we concentrate particularly on SIP-based messages over the network while RTP communications are not considered.

3. Related Work

Recently, interests in SIP failover and load balancing have risen. These two mechanisms are widely used in order to obtain highly available and scalable SIP servers. Due to number of page limitation, we briefly discuss some failover and load balancing techniques for SIP servers. We also address some research work done in this area.

3.1 Failover mechanisms in SIP Telephon

The failover consists in switching to a different redundant server, also called backup component, when the first one fails. This can be an efficient manner to handle failure and to produce fault tolerant and high available systems. In the literature, we identify various design choices depending on where the failure is detected and who does the failover. Some of them are described in the following subsections [9].

3.1.1 Client or DNS-based failover

In the client-based failover, clients have to know the IP addresses of the primary and the backup servers. First, they try the primary server and if the latter fails for some reason they switch to the backup server. In the DNS-based failover, clients no longer need to record multiple IP addresses by using two kinds of DNS records: NAPTR (Naming Authority Pointer) and SRV (Service). A SIP client can use the priority parameters of these DNS records to determine the primary and backup servers [10].

3.1.2 Failover based on database replication

Unless some SIP phones are capable of registering with multiple servers, the client can register with only the primary server that stores the user records in a master database. Then, the client registration is broadcasted to the backup server that uses a slave database. We have to note that any change in the master database is propagated to the slave one. When the primary server fails, the backup server can take over and uses its slave database to proxy the interrupted calls [10].

3.1.3 Failover using IP address takeover

To resolve the increasing failover latency problem due to the DNS caching, the IP takeover technique can be used. Both primary and backup servers have the same configuration but they run on different hosts on the same network. They are usually associated with external master database respectively slave database. When the primary server fails, the backup server takes over the same IP address. Thus, all requests are redirected to the backup server and then processed with a transparent manner.

Example.com	Priority	Weight	
_sip._udp	0	40	a.example.com
	0	40	b.example.com
	0	20	c.example.com
	1	0	d.somewhere.com

Figure 2. DNS SRV records

[2]A protocol that provides transport functions for applications transmitting real-time data, such as audio and video packets over the network.

To avoid failover latency, the server and its associated database can be co-located on the same host. By doing this, IP takeover is used by both backup server and its slave database [10].

3.2 Load balancing schemas in SIP Telephony

To increase the performance and the scalability of SIP clusters, load balancing mechanism, also called load sharing, is used. It consists in distributing the load among servers in the cluster. By doing this, it avoids some servers being idle while others have tasks queuing for execution [9]. Some well-known LB schemas in the literature are identified in the following:

3.2.1 DNS-based load sharing

The DNS SRV and NAPTER records can be used to do load balancing by using the priority and weight fields in these resource records. The Figure 2 depicts an example of DNS SRV entry. Servers a, b and c affected with priority 0 are classified as primary servers whereas d server is seen as backup server with priority 1. The weight column indicates that the a and b servers receive 80% of the call requests while the c server gets the remaining load with 20% [5],[10].

3.2.2 Identifier-based load sharing

In this approach, the user space is partitioned into multiple disjoint groups. To guarantee an uniform distribution of call requests to different servers, a hash function maps the destination user identifier to the particular group that handles the user record, e.g., based on the first letter of the user identifier. For instance, the server s1 handles calls for users who the first letter of their ID in the set {a-m}. The server s2 handles calls for users who the first letter of their ID in the set {n-z}. A high speed dispatcher distributed call requests between s1 and s2 based on the destination user identifier [5]. The major problem with this technique is that there is no guarantee that the load on servers is equal and the single point of failure still subsists.

3.2.3 Multiple servers with the same IP address

This approach consists in defining a set of redundant servers with the same IP address in the same Ethernet. The router on the subnet is built up to forward the incoming requests to one of these servers' MAC address. It can use different algorithms such as "*round robin*" or "*response time from server*" to choose the least loaded server. This method is only recommended for stateless SIP proxies that use only UDP transport and treat each request as independent in order to avoid storing any transaction state in the subnet router [10].

3.3 Previous proposals

In the literature, we identify some recent research work based on the mechanisms presented above in order to guarantee the high availability of SIP based applications.

As an application traffic management solution, F5's BIGIP system [7] provides reliable and highly available SIP networks. It plays an important role in the proxy server cluster by proposing a single controller that distributes all SIP requests to the backend SIP proxy servers. By doing this, it prevents from the unavailability of SIP proxy servers in the case of SIP server failure. However, the proposed system suffers from the single point of failure problem. It will still become unavailable when the BIG-IP itself fails [11].

A two stage architecture was proposed in [10],[9]. For failover, the authors choose the DNS-based method, and for load sharing an identifier-based approach. In the first stage, servers act like dispatchers. Clients get their IP addresses from DNS SRV records to decide which dispatcher to connect. The dispatchers receive requests from SIP user agents and then forward to one of the second stage server group based on the destination of user identifier. Due to the use of the identifierbased approach, overloading servers can happen and single point of failure still exists.

The work in [5] tries to surmount the problem of overloading servers by counting the number of connections to each SIP proxy server. Based on this number, a cluster of dispatchers in the first stage distributes efficiently call requests to a cluster of SIP proxy servers in the second stage. However, we notice that in the proposed architecture no backup servers are used in the first stage. Moreover, each dispatcher makes a list of the active SIP proxy servers every 30 seconds. During this period of time, SIP proxy servers can fail and call requests can be lost.

In the current work, we follow the same principles of these research works by proposing a two stage architecture. The failover strategy used is inspired from the failover using IP address takeover technique. To distribute load requests between SIP servers, we employ as load balancing strategy the multiple servers with the same IP address. The proposed approach is described with further details in the section below.

4. Our Approach: Load Testing of SIP-Based Clustered Systems

4.1 SIP-based clustered system

To improve the availability and the reliability of SIP based applications, we propose a two stage architecture based on failover and load balancing mechanisms. At the first stage, servers act as dispatchers, also called load balancers. At the second stage, the cluster contains a set of SIP proxy servers that share the call load in order to prevent performance degradation and overloading. Using two dispatchers, one active and the other standby as shown in Figure 3, can prevent the single point of failure problem. The active dispatcher plays an important role as a controller to monitor the load and distribute it to the cluster of SIP proxy servers. The standby dispatcher is a redundant server that will handle requests from SIP user agents in case of active dispatcher failure.

Both load balancers have their own real IP address. A virtual IP address, also called Service Access Point (SAP) \cite{WuWJH07} is associated with the active load balancer. Using this SAP, SIP user agents can access to the service correctly without knowing the architecture design in the server side. The generation and association of the SAP is done and maintained by Pacemaker[3] service which is an open source high availability middleware to perform health check and failover of dispatchers and SIP servers.

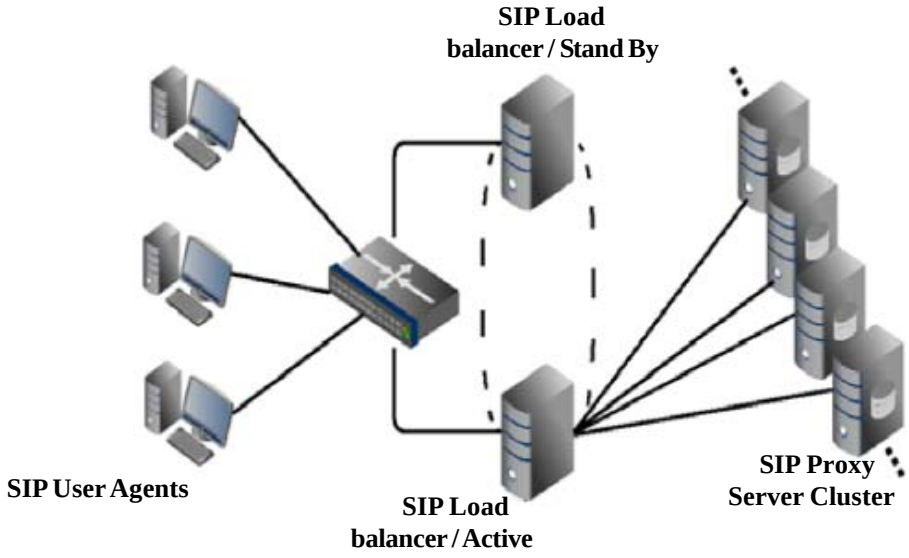

Figure 3. Two stage reliable and scalable architecture

Pacemaker is in charge of managing SIP clusters with the purpose of producing high available and scalable SIP-based applications. In fact, it checks the availability of dispatchers every 100 ms. If the active dispatcher is no longer available then the virtual IP address is assigned to the passive dispatcher which becomes Active. We can reduce the failover time by adjusting the value of the "*token*" parameter[4] which is a timer to decide whether the monitored dispatcher is dead if no heartbeat has been received during the "token" value.

The dispatcher is in charge of handling SIP user agents requests and distributing them to a set of SIP proxy servers. We make use of some tools and applications of Linux Virtual Server (LVS)[5] the most important open source project of high availability solutions. In particular, we exploit Ldirectord[6]and IP Virtual Server (IPVS)[7] tools to dispatch SIP requests to SIP proxy servers. Ldirectord implements three types of dispatch method: Virtual Server via Direct Routing, Virtual Server via IP Tunneling and Virtual Server via NAT. In our context, we used the direct routing request dispatching technique. There are nine implemented algorithms to serve incoming requests, the default one is Weighted Least Connections (WLC) and the most used one is Weighted Round Robin (WRR).

[4] 300 ms is the smallest value allowed in Pacemaker configuration
[5] http://www.linuxvirtualserver.org/
[6] http://horms.net/projects/ldirectord/
[7] http://www.linuxvirtualserver.org/software/ipvs.html
[3] http://www.clusterlabs.org/

Each one of the SIP proxy servers makes use of its own database to store information about SIP user agents credential (login, password), registration and emplacement. All databases are synchronized with each other to assure that the same information is treated by all SIP proxy servers.

4.2 Distributed load testing architecture

Load testing is a kind of performance testing that allows evaluating the System Under Test (SUT). It is performed to determine SUT behavior under heavy load and to collect measurements as response times, throughput, resource utilization, maximum user load, etc. In order to identify the maximum operating capacity of a SUT and to test its stability, load testing simulates various loads and activities that a system is expected to encounter during its execution time.

Therefore, it helps detecting problems like abnormal delays, availability or scalability issues, or failover when the number of emulated users is increased [4].

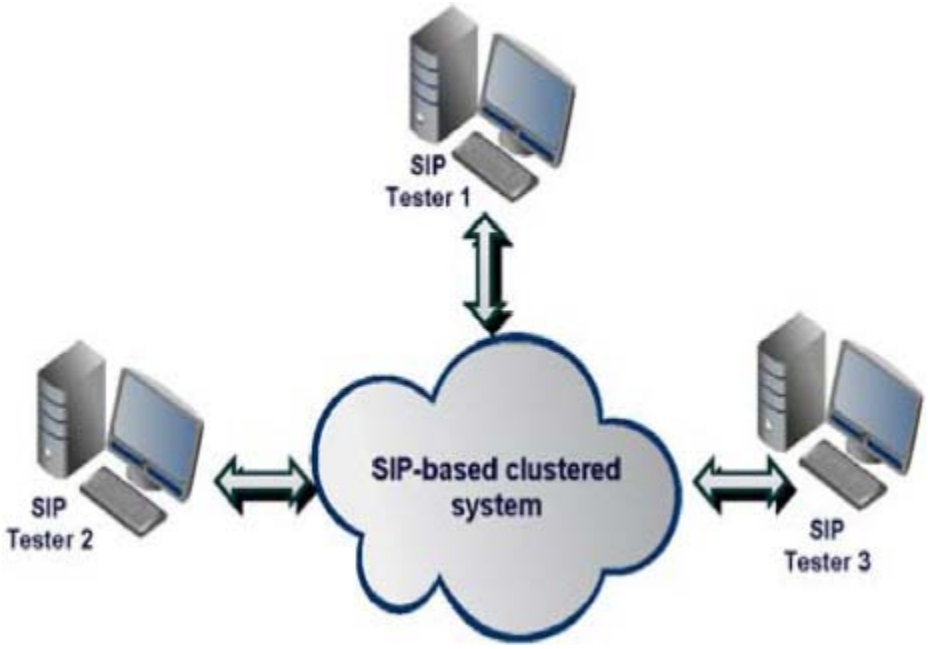

Figure 4. Distributed test architecture

To check the performance of the SIP-based clustered system, we propose a distributed load testing architecture as depicted in Figure 4. The main purpose of this test architecture is to verify the performance of the proposed SIP-based clustered system especially when a dispatcher or SIP server fails. The SIP tester is responsible of generating a great amount of call requests in order to stress the system under test with the aim of evaluating some metrics as CPU loading, throughput, latency, etc.

5. Experimental Test Bed and Results

In this section, we describe the software and hardware that we used in our experimentation.

5.1 Experimental test bed

5.1.1 SIP server software

We have chosen to build our VoIP system based on the best available open source server, Asterisk[8] The chosen sever is a Private Branch eXchange (PBX) software which supports many signaling protocols, especially SIP protocol.

5.1.2 SIP tester software

For the tester side, we use the free open source SIPp tool[9]. SIPp is a call traffic generator and analyzer for the SIP protocol. It

[8] http://www.asterisk.org/

[9] http://sipp.sourceforge.net/

includes a few basic SipStone User Agent Client (UAC) and User Agent Server (UAS) scenarios by establishing multiple calls with the INVITE and BYE methods. It can be also extensible via a simple XML configuration language.

5.1.3 Clients and servers hardware and OS

All servers in our architecture use Debian Lenny 5.0.7 64 Bit, while the client machines use Ubuntu 10.04 LTS 32 Bit. They are also virtual machines on one real server hardware HP Proliant DL160 G5 E5405 Quadric core 2 Ghz.

5.1.4 Test bed set up

As mentioned before, we use Pacemaker tool to set up two dispatchers (LB1 and LB2) and four SIP proxy servers (SIP1, SIP2, SIP3 and SIP4) (see Figure 5). Also, two SIP testers (SIPp1 and SIPp2) on which the SIPp tool is running are installed. All machines in our experimentations are connected via a Gigabit virtual network.

Machines	***CPU***	***RAM***
SIP Proxy server	QuadriCore 2 Ghz	1 Go
Load balancers	QuadriCore 2 Ghz	512 Mo
SIP testers	QuadriCore 2 Ghz	1Go

Table 1. Hardware Test Bed characteristics

5.2 Results

In this section, we test the performance of the SIP-based clustered system by studying the following test cases.

5.2.1 Case A: Dispatchers and SIP servers run correctly

In this experiment, we measure the number of call requests running in each SIP server during 30 minutes. To illustrate how the load is shared between the four SIP servers, two requests distribution methods are used: Round-Robin (RR) and Weighted Least-Connections (WLC)[10]. As depicted in Figure 6, the call requests are distributed sequentially between the running servers respecting the principles of RR method without regarding to server capacities and load. Figure 7 illustrates the distribution of call requests using the WLC method that consists in distributing more requests to servers with fewer active connections relative to their capacities that are indicated by assigning weight to each server. We have to note that the call rate is initially fixed at 10 calls per second.

5.2.2 Case B: One SIP server fails

In order to verify the performance of the proposed architecture, we increase the load in SIPp1 and SIPp2 every 10 minutes by 10. Due to the overloading and the huge amount of generated calls, the number of failed calls raise especially at the 114th minute of the testing duration. This sudden augmentation can be explained by the failure of one SIP server (see Figure 8). The latter becomes unavailable and could not serve the coming requests. We have to note that the active dispatcher has detected this failure and has removed the crashed server from its list. Also, all the new requests are forwarded to the active SIP proxy servers. Since the 120th minute of the testing duration, we have observed that the new call requests have been rarely lost because they are assigned to the running servers.

5.2.3 Case C : One dispatcher fails

In this test, we aim to measure, under heavy load, the number of failed calls especially when the active dispatcher fails. Thus, we modify the two SIPp testers configuration to generate 100 calls per second as call rate. As shown in Figure 9, the active dispatcher fails at the 5th minute of the testing period that lasts 2 hours in this experiment. Therefore, the number of failed calls increases suddenly. When the standby server becomes active, all the generated calls are treated and no new failed calls are detected. At the 10th minute of the testing duration, the first dispatcher resumes its work. Consequently, new failed calls are identified until the failover is done.

[10] http://www.linuxvirtualserver.org/docs/scheduling.html

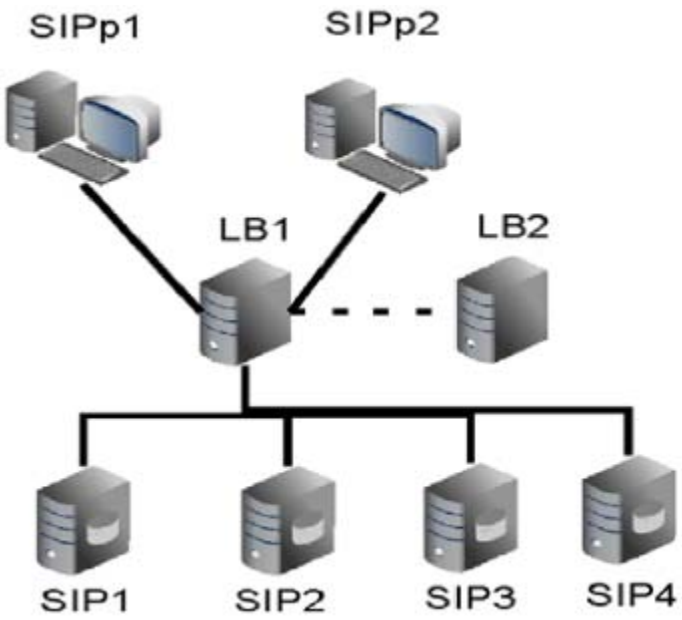

Figure 5. Test bed architecture

6. Conclusion

In the e-learning environment, availability and scalability challenges have to be carefully addressed in order to avoid overloading and failure of web applications and online services as VoIP technologies.

Therefore in this paper, we have realized a highly available and scalable SIP-based clustered architecture. Moreover, a distributed load testing architecture is presented in order to test the performance of the used failover and load balancing techniques. We implemented our approach by using recent tools such as Pacemaker which automatically initiates recovery high availability mechanisms, SIPp as a traffic generator tester and ldirectord and IPVS tools to dispatch SIP requests to SIP proxy servers. The experimental results have shown the importance of failover and load balancing techniques to produce a highly available SIP-based clustered system. In addition, the use of load testing technique as a mean of validation points out the real performance of the running system and helps to detect inconsistencies and failure.

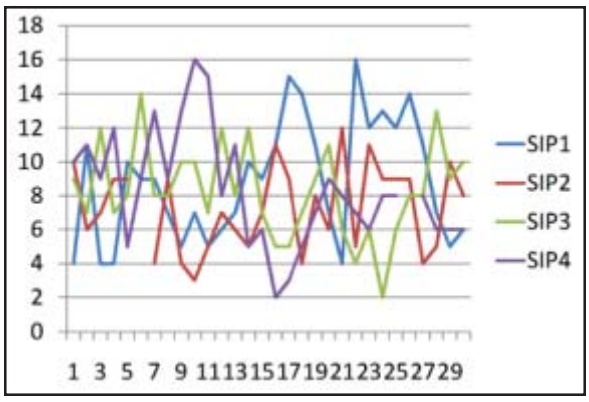

Figure 6. Case A: Call requests distribution using RR method

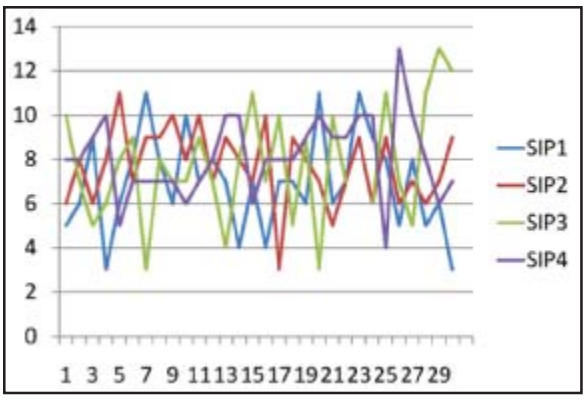

Figure 7. Case A: Call requests distribution using WLC method

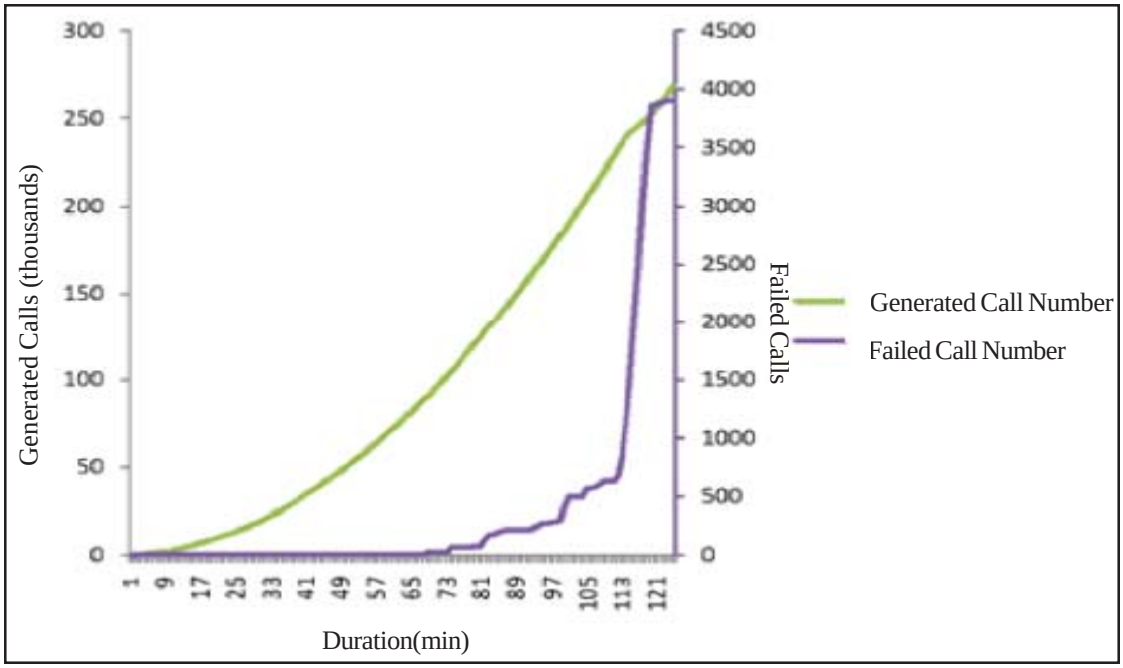

Figure 8. Case B: Failed calls due to SIP server failure

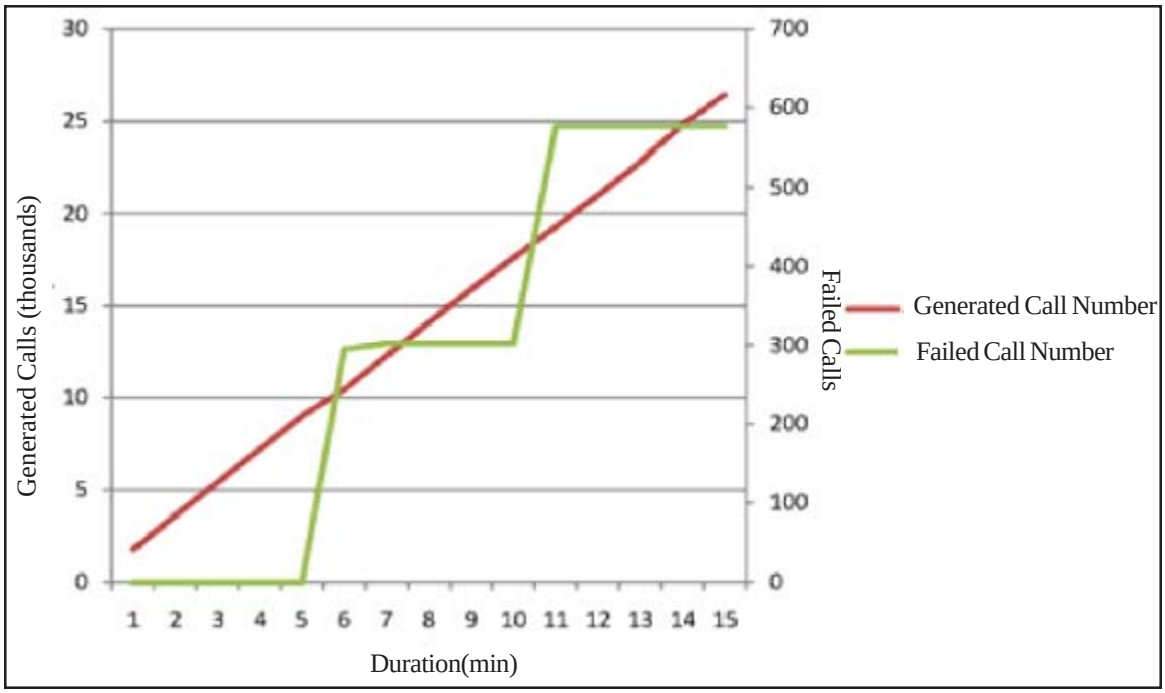

Figure 9. Case C: Failed calls due to dispatcher failure

As future work, we intend to experiment the proposed approach on real machines not virtual ones.

In addition, we plan to verify its availability and scalability when we deal with a mobile network. Also, we aim to propose a more general testing approach by adopting the model based testing paradigm.

References

[1] Adorni, G., Coccoli, M., Fadda, C., Veltri, L. (2007). Audio and video conferencing tools in learning management systems. *In*: Proceedings of the Third International Conference on Automated Production of Cross Media Content for Multi-Channel Distribution, p.123–129, Washington, DC, USA. IEEE Computer Society.

[2] Alam, M., Bose, S., Rahman, M., Abdullah Al-Mumin, M. (2007). Small office pbx using voice over internet protocol (voip). *In*: Advanced Communication Technology, The 9th International Conference on, 3, 1618 –1622.

[3] Alwi, N., Fan, I. S. (2009). Information security management in e-learning. *In*: Internet Technology and Secured Transactions, 2009. ICITST 2009. International Conference for, p.1 –6.

[4] Din, G., Tolea, S., Schieferdecker, I. (2006). Distributed load tests with ttcn- 3. *In*: TestCom, pages 177–196.

[5] Karimi, A., Sarram, M., Ghasemzadeh, M. (2010). Two stage architecture for load balancing and failover in sip networks. Middle East Journal of Scientific Research, p. 88–92.

[6] Kumar, A. (2006). An overview of voice over internet protocol (voip). Online Academic Journal.

[7] Network, F. A new paradigm for sip high availability and reliability. [on line], Available: http://www.f5.com/solutions/technology/pdfs/sip wp.pdf.

[8] Panagiotakis, S., Kalogiannakis, M., Ripoll, N.,Vassilakis, K. Towards synchronous tele-education: Solutions, trends and experience gained. [on line], Available: http://elexforum.etqm.ae/Proceeding/PDF/Towards/Tele-Education.pdf.

[9] Singh, A., Schulzrinne, H. (2004). Failover and load sharing in sip telephony. *In*: Proceedings of the International Symposium on Performance Evaluation of Computer and Telecommunication Systems (SPECTS).

[10] Singh, K., Schulzrinne, H. (2007). Failover, load sharing and server architecture in sip telephony. Comput. Commun., 30, 927–942.

[11] Wu, W. M., Wang, K., Jan, R, H., Huang, C, Y. (2007). A fast failure detection and failover scheme for sip high availability networks. *In*: Proceedings of the 13th Pacific Rim International Symposium on Dependable Computing, p. 187–190, Washington, DC, USA. IEEE Computer Society.