Journal of Engineering Science and Technology Review 7 (3) (2014) 1 – 6

JOURNAL OF
Engineering Science and Technology Review

www.jestr.org

Research Article

Two Stage Secure Dynamic Load Balancing Architecture for SIP Server Clusters

G. Vennila* and MSK. Manikandan

Department of Electronics and Communication Engineering, Thiagarajar College of Engg., Madurai, INDIA

Received 7 July 2014; Accepted 1 August 2014

Abstract

Session Initiation Protocol (SIP) is a signaling protocol emerged with an aim to enhance the IP network capabilities in terms of complex service provision**.** SIP server scalability with load balancing has a greater concern due to the dramatic increase in SIP service demand. Load balancing of session method (request/response) and security measures optimizes the SIP server to regulate of network traffic in Voice over Internet Protocol (VoIP). Establishing a honeywall prior to the load balancer significantly reduces SIP traffic and drops inbound malicious load. In this paper, we propose Active Least Call in SIP Server (ALC_Server) algorithm fulfills objectives like congestion avoidance, improved response times, throughput, resource utilization, reducing server faults, scalability and protection of SIP call from DoS attacks. From the test bed**,** the proposed two-tier architecture demonstrates that the ALC_Server method dynamically controls the overload and provides robust security, uniform load distribution for SIP servers.

Keywords: Load balancer; honeywall; SIP; VoIP; overload; Least Session

1. Introduction

SIP is a VoIP signaling communication protocol used for establishing, modifying, controlling and terminating multimedia session such as video and voice call over IP based network. SIP based telephony is an alternative method to Public Switched Telephone Network (PSTN) due to its greater flexibility and lower costs [1-2]. Due to rapid growth and increasing demand of SIP deployments in VoIP network, it's necessary to handle overload and customer service demand.

Load balancing in SIP server become a powerful solution to various problem like security, scalability, reliability and management[3]. Increasing SIP signaling traffic causes overload in server, measuring factors such as throughput, response time and CPU utilization are significantly affected. This paper presents and analyzes load balancing algorithm for distributing load by eradicating signaling attack and anomaly load using honeywall which are unlikely to the SIP server. We develop a set of rules for above in the honeywall thus, work task of the load balancer enhanced furthermore. As part of an overall security approach, the proposed two tier architecture maintains SIP application performance and availability, thus allowing time for segregation and blocking of attack traffic.

Proposed ALC_Server algorithm is implemented in the load balancer which forwards the SIP request of the corresponding call to its corresponding server thus, eliminating the problem of incomplete sessions after heavy load. The main goal of ALC_Server load balancer is that route a new call towards the SIP server that has lest number of active call which has least number of *BYE_ACK* method. In case any server in the VoIP network has same number of least call and methods, we make use of FIFO fashion. We compare our load balancer with Least Session Method (LSM) and measure various parameters such as throughput, CPU Utilization and response time.

1.1 SIP Overview

SIP is a standard protocol for multimedia conferencing defined by an Internet Engineering Task Force (IETF)[2]. SIPs are location-independent with negotiate session characteristics and support other protocols, identify and carry out multimedia session. SIP session negotiation is accomplished with the help of Session Description Protocol (SDP) and Real time Transport Protocol (RTP) is responsible for multimedia transactions over IP network[1]. This RTP in conjunction with Real-time Transport Control Protocol (RTCP) provides session synchronization and other media transaction statistics. SIP is a transaction based protocol similar to the HTTP method. Each SIP transaction consists of a client request that invokes a specific method on the server and generates corresponding response.

A complete SIP transaction is established between two user agents that close after a period of time called as SIP *call* or *dialog*. SIP transactions are considered as either *INVITE* or *Non-INVITE,* an *INVITE* transaction establishes a long conversation which includes an acknowledgement (*ACK*) without failure of closing response (e.g *200 OK*). A new transaction begins, when the BYE message is generated to its corresponding INVITE request. An *INVITE* transaction consumes more processing time than BYE method [3]. Figure 1 shows the SIP client A and B initiate and receive sessions.

The SIP client A generates an *INVITE* request and sends it to the SIP client B. The main principle of a SIP server is to forward requests to clients closer to its destination. The SIP proxy servers are more light-weight compared to PSTN because it can only route call without maintaining any

* E-mail address: vennilatg@tce.edu

session details. The request may traverse via one or more servers. Media session is exchanged and then the call is terminated by sending a *BYE* message. After completing SIP transaction, SIP clients uses cryptographic algorithms to encrypt/decrypt the voice packets.

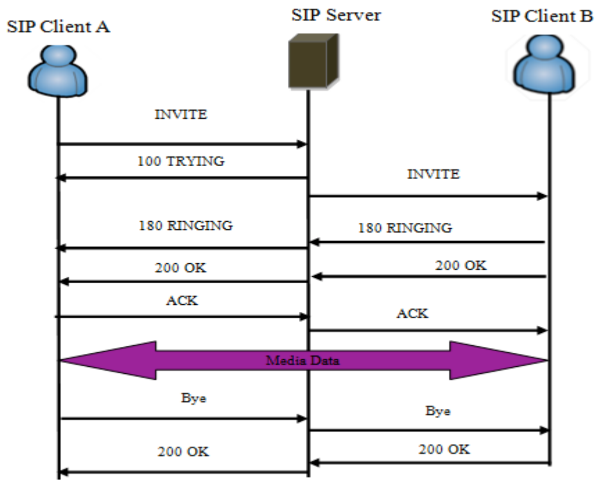

Fig. 1. An example SIP message flow

The rest of this paper is structured as follows: Section 2 describes related work. Section 3 illustrates proposed two tier architecture. Experimentation and performance results are shown in section 4 and 5. Section 6 concludes the paper.

2. Related works

Load distribution and balancing in SIP servers are some of the areas in VoIP network that are extensively in focus today. In particular, the problems of optimal load distribution in distributed computing environment is studied using queuing models [4].The overload control mechanism using dispatchers for SIP server clusters are discussed in [5-6]. SIP based failover and load distribution method for routing a call to the SIP server and various prevention methods are given in [7-8].

In SIP based cluster architecture, three issues are considered to avoid SIP failover with a single point of failure, health monitoring of proxy server and load balancing of SIP proxy servers [9]. SIP server performance gets degraded due to retransmission and anomaly load [10]. To prevent this problem window based overload control algorithm is proposed [11]. Some of the overload control mechanism cannot handle overload condition effectively. To detect overload at the downstream server and control retransmission rate from upstream server a novel algorithm is presented in [12]. A Distributed End-to-End Overload Control (DEOC) mechanism provides high throughput and responds faster to the abrupt variation of the load generation [13]. SIP server performance varied depending on how the server is configured. The measuring parameters of server like throughput, response time and CPU utilization also gets affected due to authentication scheme used in the server, whether the transaction is stateful or stateless and type of protocol used (TCP, UDP) are discussed in [14]. Over load control methods are applied in either end to end or hop-by-hop scheme. Hop-by-Hop overload control is simple and balance loads in VoIP networks with many SIP clients. Over load control loop method applied in an end-to-end system controls overload in the entire path of the SIP request, from user agent client to user agent server. But this method needs prior information about load from all SIP servers and update load status on the potential possible path to a destination [15].

SIP configured in both Back to Back User Agent (B2BUA) and proxy modes, simulation results are presented in [16]. Attacks in VoIP systems causes damage and are more imaginative. The security status of VoIP systems are observed with the help of a honeynet environment. The result describes existing prevention techniques utilized to recognize and analyze attacks in production environment [17]. A comparison of existing algorithms like round robin, random, SITA-E and dynamic schemes with demonstrating results is explained in [18]. If a newly generated request arrives randomly to the system, the servers become significantly loaded. For this situation, a main objective of load sharing to transfer request from heavily loaded to lightly loaded servers using ALC_Server scheme is discussed in this paper. Thus, there is no SIP server in idle or waiting state. To the best of our knowledge, a secure uniform load distribution with honeywall and performance comparison with existing algorithm (LSM) towards multiple servers has not been considered in the earlier works.

3. Proposed two tier SIP architecture

In this section, we propose to combine two methods in two tier architecture to improve scalability, security and reliability by adding an additional stage which do not affect the media session.

3.1 Secure first tier honeywall architecure

Honeywall is the first element to capture the load from the SIP clients. Honeywall provides a transparent gateway to the honeynet system. A honeypot is virtually installed in the SIP server to trace the attacker. This honeynet topology is completely untraceable by an attacker. The proposed honeywall architecture offers three functionalities such as data capture, data control and data analysis [19]. *Sebek* performs as a data capturing tool, intended to capture SIP load on a server, it secretly stores the results in log directory. The result directory logs are used to study the behavior of attack traffic like type of attacks and protocols used by an attacker.

This logged activity is sent out in the form of *sebek* packets. S*ebek* packet data taken from the *pcap* files with the help of wireshark and *sebek* scripts are analyzed. *Snort* inline tool is used to control and analyze the SIP call flow rate based on rules written in this mode. SIP packets are compared against inline rule-sets and actions are taken to decide whether to allow or drop the packets with the help of *iptables*. We used *walleye* for data analysis and its interface is remotely accessible from the honeywall. The data analysis interface is the most powerful user interface for analyzing both real-time and previously archived SIP network packet. This log result includes *pcap* data, packet payload, real time SIP call flows, inbound and outbound data traffic, action taken out by the *snort*, *sebek* based data log and honeywall activity summary. Figure 2* depicts honeywall implementation in VoIP architecture. This first tier architecture consists of a honeynet and SIP clients. A honeypot is a trap set to the attacker to analyze the attacker methodology and alerts the SIP servers regarding the malicious packets that bypass the load balancers. A

Honeynet is a network of honeypots organized in a VoIP environment.

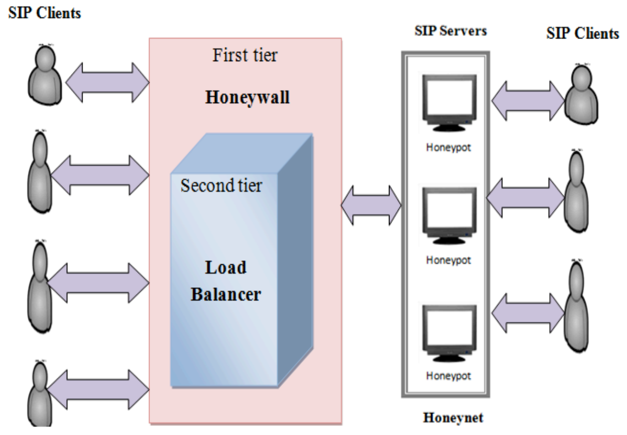

Fig. 2. Secure two tier SIP architecture

Here, honeywall is the major contribution for reducing the load towards the load balancer acts like a security wall that prevents the entry of unwanted flooding towards SIP server clusters. This architecture creates a high control and monitoring of all kinds of SIP clients and VoIP network activity.

3.2 Second tier load balancer architecure

Load balancing is significantly more important due to overload and anomaly traffic. Our load balancer has a choice to select the server during reception of new *INVITE* messages and *Non-INVITE* methods can only be forwarded to the respective servers handling the corresponding session. The implemented ALC_Server algorithm dynamically measures server parameters such as capacity, speed, number of active sessions and work assigned to a server based on sending request to a server and then receiving response from the server. To evaluate the health of the server, load balancer always observes response from every SIP server. The SIP server is configured to utilize the proposed algorithm which makes decision to select the server with the least number of active calls and *BYE_ACK* method to ensure that the load of the active requests are balanced by the server.

In the two tier architecture, SIP clients generates request to the load balancer. Before that, honeywall gateway reduces SIP traffic towards the server and the proposed ALC_Server algorithm selects a proper SIP server to handle all the incoming requests. Each response from SIP servers at first go through honeywall, then the load balancer, it is then forwarded to the appropriate SIP clients. By monitoring these transactions, the load balancer algorithm computes server completed transactions and updates the work assigned to each SIP server in the network. Figure 3 illustrates the pseudocode and working flow model for the proposed system.

The captured SIP packets from honeywall are forwarded towards the load balancer. If the received session request is a new *INVITE* method, the load balancer assigns it to the new SIP server based on least number of active calls and *BYE_ACK* method. Otherwise, ALC_Server scheme forwards the packets to the corresponding servers handling the existing sessions thereby call synchronization before and after the load is achieved. The health of SIP servers are checked using the updation table entries for the SIP request and response methods.

```
Assign c= counter; l= load balancer log ; t= table entry
S= existing call server
S1= new server
/*server selection based on    ALC_Server_Method */
If session_request = INVITE
   /* new session is allocated to the server S1 */
     assign session to S1
      S1 = STATUS_New INVITE
       counter c++
        /*add entry to the table*/
          update t
           forward t to l
else
   /*if session_request is Non_INVITE*/
      assign session to S
       S=STATUS_existing session
        counter c--
          update t
             forward t to l
```

Fig. 3. Pseudocode for ALC_Server algorithm

3.3 Existing algorithm

LSM method routes a new call towards the SIP server that has least active number of session instead of least active calls. Here, session defines that the each SIP request and responses are initiated by the client and server. The main limitation of this method is that each session has different cost, consumes more resources and takes longer idle period to complete a call [3]. For example, an *INVITE* request takes longer period to complete a call. But the *BYE* request consumes fewer resources and takes short period of time for a call. At the same time existing requests are not forwarded to corresponding handling server. This method automatically achieves less percentage of call completion rate and throughput.

4. Experimental test bed

Figure 4 demonstrates experimental test bed of honeywall with load balancer implementation. The test bed consists of a load balancer algorithm and honeywall implemented in the server, SIPp load generator and OpenSER SIP server. Oprofile tool is used to monitor the CPU profiles, performance and utilization where as Nmon presents all the important tuning information on screen.

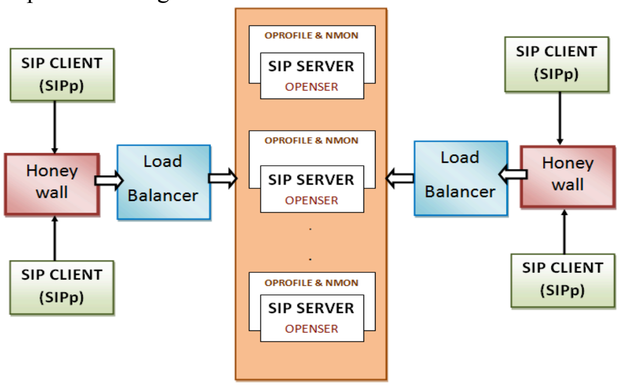

Fig. 4. Experimental testbed

4.1 Work load generator

We use open source SIPp traffic generator tool to generate a maximum of 9000 calls/sec**.** It is a XML configurable packet generator, composed of event driven architecture. SIPp contain different scenarios like User Agent Client (UAC) and User Agent Server (UAS) which launches multiple calls with *INVITE* and *BYE* methods. We make use of UAC scenario model consisting of an *INVITE* request method in which the OPENSER SIP server responds with *100 TRYING*, *180 RINGING* and *200 OK* response methods. UAC loads are increased every 30 seconds by 10 and with a pause time of 30s to 1 min. After pausing a call to UAC model, the SIP clients immediately sends a *BYE* message to terminate the session. Then the server returns a *200 OK* response code. We implement two architectural methods which include,

a. Load balancer with honeywall
b. Load balancer without honeywall

4.2 SIP Server
OPENSER back-end SIP proxy server is installed in redhat linux environment. We use five back-end servers to analyze and evaluate the overall performance of our algorithm. The request rate starts initially with 5 calls per second (cps) and gradually increases every second. After 10 seconds request rate is increased by 50 cps. Network analyser tool (*Wireshark*) analyzes SIP packet flow which is established in client and server machines. Initially, load is distributed uniformly, all SIP clients obtained increased rate of received calls after 1 additional cps until the experimental rate is reach.

4.3 Honeywall
The honeywall is the primary element for entry and exit of VoIP network traffic. The honeywall actively monitors, analyzes and controls the incoming traffic to the load balancer. The main idea of implementing a honeywall is to reduce the workload of the load balancer by eliminating suspicious traffic at the initial stage itself. Honeywall logs provide us evident results of the various kinds of attack, traffic and the techniques used. Honeywall is configured [19] and *Hwctl* control command line is utilized for changing and updating the value of the honeywall variable. In general, Honeywall has three interfaces (*eth 0, 1, 2*); we used interface *eth 2* for handling SIP management traffic. The honeywall then transmits the incoming load to the level II load balancer architecture.

5. Performance Analyses
In this section, we quantitatively evaluate the performance of proposed two-tier architecture for scalability with security using the experimental testbed.

5.1 Throughput
Throughput is defined as the number of requests completed per second by the SIP server. We compute the maximum throughput of a SIP client node to be 6804 cps for honeywall with pause and 6513 cps for honeywall without pauses. Two parameters with and without pauses ensure that ALC_Server load balancer provides promising results even with uninterrupted overloading of the server.

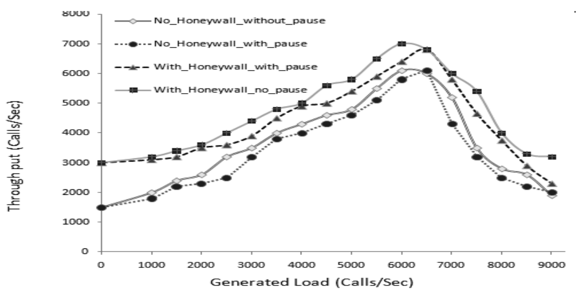

Fig. 5. ALC_ Server load balancer throughput comparison with offered load towards the server

This is significantly proved from the figure 5 which gives maximum throughput for honeywall without pauses. The same scheme without honeywall provides throughput of about 5620 cps without pause and 5467 cps with pause. The reason behind this drop is because the honeywall limits the incoming floods towards the SIP load balancer according to the proposed rules inside the honeywall to block the incoming load.

5.1.1 Peak throughput
Figure 6 shows that peak throughput of two different load balancers when increasing number of nodes in the test bed environment.

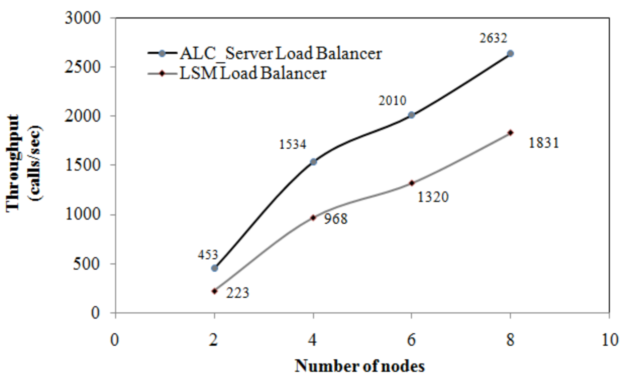

Fig. 6. Peak throughput

We implement LSM algorithm in SIP server which achieves maximum peak throughput of 1831 for the 8 nodes. For the same number of nodes ALC_Server load balancer achieves highest peak throughput of 2632 calls/sec because of that reduced response time is obtained.

5.2 Response time
Response time is defined as the time duration between when a request (*INVITE/BYE*) is sent and the successful 200 OK is received. For response time calculation y axis is represented in logarithmic scale. Our load generator can produce an aggregate request rate of 9000 cps with respective pauses to ensure that the response time is reduced by limiting the load request by the honeywall. We observed the response time between 3000 to 9000 cps and there was an increase in response time slightly from 3000 cps without pauses when honeywall is used. This shows that reduced response time than the system without honeywall. There is a slight degradation in response times from 4500 cps in without honeywall and pause as shown in figure 7.

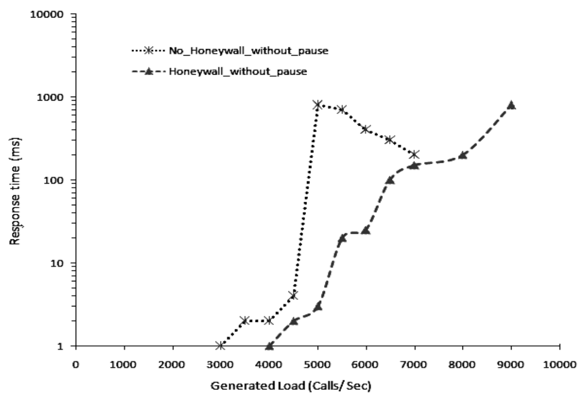

Fig. 7. Average response time versus generated load

5.3 CPU Utilization

A comparative study of different loads carried out in the test bed for CPU utilization is shown in figure 8. The graph shows the percentage level of CPU utilization over different loads. The x-axis represents the load generation (cps), and the y-axis represents the percentage of utilization. We observed a maximum 96.97 % of CPU is utilized around 6000 cps using honeywall without pause when SIP specific servers are used. It is obvious that many servers may be powerful than others and its CPU utilization varies depending on the load and type of request that limits the resources. When the OPENSER SIP server is overloaded, the CPU utilization is close to 100%.

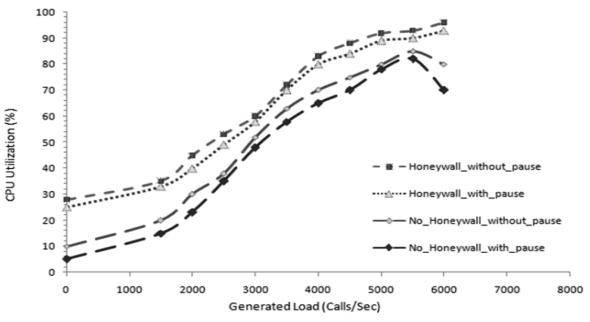

Fig. 8. CPU utilization comparison with offered load

5.4 SIP Call completion rate

From SIP client's perspective, the call completion rate is important. Figure 9 depicts number of calls generated vs completion rate.

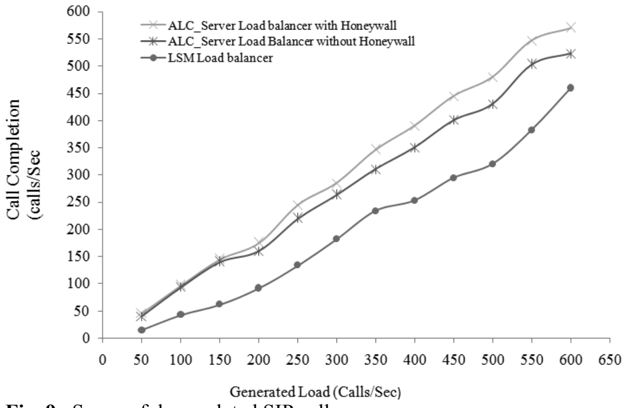

Fig. 9. Successful completed SIP calls

The processing time of LSM method varies with SIP call length. Because each SIP calls have different number of transaction and consumes more server resources. So the successful number of call completion rate in LSM is lower than ALC_server method. Proposed load balancer handles maximum of 570 calls in honeywall implementation, 523 calls without honeywall implementation and 460 calls are completed in LSM method out of 600 calls.

6. Conclusion

Our proposed two tier architecture of VoIP network evidently increases reliability, security and scalability. The experimental test bed results help in managing the incoming IP traffic and dropping the malicious packets before load distribution across multiple SIP servers. On the other hand our algorithm provides better response time, maximizes throughput and CPU utilization. The dramatic reduction in response time is achieved by ALC_Server algorithm in the load balancer along with honeywall implementation. Operation of honeywall at the earlier stage greatly reduces the work of the load balancer spending its time in processing and balancing the unwanted malicious flooding packets targeting the SIP server. Thus honeywall with load balancer provides improved efficiency in terms of security and traffic management in a VoIP honeynet environment.

References

1. Schulzrinne, H., and Rosenberg, J., "The Session Initiation Protocol", *IEEE Communication Magazine*, vol.38, pp.134-141, 2000.
2. Rosenberg, J., "SIP: Session Initiation Protocol", RFC3261.
3. Hongbo Jiang, Arun Iyengar, Erich Nahum, Wolfgang Segmuller, Asser N. Tantawi and Charles P.Wright, "Design, Implementation, and Performance of a Load Balancer for SIP Server Clusters", *IEEE/ACM Transactions on Networking*, vol.20 (4),pp.1190-1202, 2012.
4. Kameda, H., Li, J., Kim, C., and Zhang, Y., "Optimal Load Balancing in Distributed Computer Systems", *Springer-Verlag*, vol. 32(2), pp.445- 465, 1997.
5. Bryhni, H., Klovning, E., Oivind Kure, "A comparison of load balancing techniques for scalable web servers", *IEEE Network*, vol. 14, 2000.
6. Suryanarayanan, K., Christensen, KJ., "Performance evaluation of new methods of automatic redirection for load balancing of apache servers distributed in the Internet", *International IEEE Conference on Local Computer Networks*, USA, 644-651, 2000.
7. Kundan Singh and Henning Schulzrinne "Failover and Load Sharing in SIP Telephony", *Journal of Computer Communications*, vol. 35(5), pp. 927-942, 2007.
8. Xu, C., and Lau, F., "Load-Balancing in Parallel Computers", Theory and Practice, Kluwer Academic, 1997.
9. Alireza, K., Mehdiagha, S., and Mohammad, G., " Two stage architecture for load balancing and failover in SIP network", *Middle-East Journal of Scientific Research*, vol.6, pp.88-92, 2010.
10. Azhari, SV., Homayouni,M., Nemati,H., Enayatizadeh, J., Akbari, A., " Overload control in SIP networks using no explicit feedback: A window based approach," *Journal of Computer Communications*, vol.35, pp.1472–1483, 2012.
11. Hilt, V., and Widjaja, I., "Controlling overload in networks of SIP servers", *IEEE International Conference on Network Protocols (ICNP 2008)* , pp.83-93,2008.

12. Liao, J., Wang, J., Li, T., and Zhu, X., "A distributed end-to-end overload control mechanism for networks of SIP servers," *Journal of Computer Networks*, vol. 56(12),pp. 2847-2868, 2012.
13. Hong, Y., Huang, C., and Yan, J., "Controlling retransmission rate for mitigating SIP overload", *IEEE International Conference on Communications (ICC)*, pp.1-5, June 2011.
14. Nahum, EM., Tracey, J., and Wright, CP., "Evaluating SIP server performance", *SIGMETRICS, Performance Evaluation Review*, vol. 35, pp. 349-350,2007.
15. Widjaja, I., Hilt, V., and Schulzrinne, H., "Session Initiation Protocol (SIP) Overload Control," Internet Draft, Sipping Working Group, January 2008.
16. Miroslav, V., Jan, R., "Methodology for SIP Infrastructure Performance Testing", *WSEAS Transactions on Computers*, vol.9, 2010.
17. Gruber, M., Fankhauser, F., Taber, S., Schanes, C., and Grechenig, T., "Security status of VoIP based on the observation of real-world attacks on a honeynet", *3rd IEEE International Conference on Information Privacy, Security, Risk and Trust*, pp.1041-1047, 2011.
18. Harchol-Balter, M., Crovella, M., and Murta, CD., "On choosing a taskassignment policy for a distributed server system," *Journal of Parallel Distributed Computer*, vol. 59(2), pp. 204–228, 1999.
19. http://www.honeynet.pk/honeywall/roo/page3.html