ResearchGate

See discussions, stats, and author profiles for this publication at: https://www.researchgate.net/publication/261318018

Overload control in Asterisk proxy: Implementation and evaluation

Conference Paper · May 2013

DOI: 10.1109/IranianCEE.2013.6599695

CITATIONS
0

READS
450

2 authors:

Ahmadreza Montazerolghaem
University of Isfahan
52 PUBLICATIONS **303** CITATIONS
SEE PROFILE

Mohammah-H Yaghmaee
Ferdowsi University Of Mashhad
246 PUBLICATIONS **3,274** CITATIONS
SEE PROFILE

All content following this page was uploaded by Ahmadreza Montazerolghaem on 16 May 2015.

The user has requested enhancement of the downloaded file.

Overload control in Asterisk Proxy: implementation and Evaluation

Ahmadreza Montazerolghaem
Department of Computer Engineering
Ferdowsi University of Mashhad
Mashhad, Iran
Ahmadreza.montazerolghaem@stu-mail.um.ac.i

Mohammad Hossien Yaghmaee moghaddam
Department of Computer Engineering
Ferdowsi University of Mashhad
Mashhad, Iran
yaghmaee@ieee.org

***Abstract*—The extent and diversity of systems provided by IP networks have lead various technologies to approach integrating various types of access networks and converting to next generation network. On account of features as being in text form, end-to-end connection, independence from the type of transmitted data, and supporting various forms of transmission, is an appropriate choice for signaling protocol in order to make connection between two IP network users. These advantages have made SIP be considered as a signaling protocol in IMS, a proposed signaling platform for next generation networks. Despite having all these advantages, SIP protocol is in lacking of appropriate mechanism for addressing overload. In this paper the window-based overload control mechanism which does not require explicit feedback is developed and implemented on Asterisk open source proxy and evaluated. The results of implementation show that this method could practically maintain throughput in case of overload. As we know this is the only overload control method which is implemented on a real platform without using explicit feedback. The results show that the under load server maintains its throughput at the maximum capacity.**

Keywords— SIP; Overload control; Asterisk Proxy;

I. INTRODUCTION

SIP protocol is the signaling protocol in application layer which is used to start, manage, and finish the meeting between two or more application. The major components of a SIP network are user agents, server proxies, and registrars. User agent is the terminal component in SIP session. The proxy task is routing and redeploying signaling between user agents.

SIP server is an application one. The overload problem in SIP server is distinguished with ones in other HTTP servers for at least three reasons: firstly the messages of SIP meeting pass several SIP proxy servers to reach destination which itself could make overload between two SIP proxy servers. Secondly SIP has several retransmit timers which are used for dealing with package loss, especially when the package is sent via UDP transmission protocol, and this could lead to overload on SIP proxy server. Thirdly SIP requests are used as real time session signaling, so have a high sensitivity. Overload in SIP-based networks occur when the server does not have sources necessary for answering every received call. Reviews accomplished in overloaded SIP proxy server show that increasing request rate results in sudden increase in delay in establishing connection and dropping proxy throughput and therefore increase in unsuccessful call rates. Therefore the aim in overload control in SIP is maintaining the throughput of overloaded server near its capacity. Generally there are two local and distributed methods for overload control. In local control when SIP proxy server reaches its capacity threshold, it starts to reject requests; SIP estimates this threshold by calculating CPU consumption or queue length. But request rejection mechanism in order to finish meeting imposes cost itself and when server is overloaded it is compelled to allocate fraction of sources to reject requests, which in turn decreases efficiency in SIP proxy server. In distributed method, upstream servers control the load of downstream servers through rejecting requests and try to maintain it under their capacity.

In this paper the overload control method is implemented on an appropriate platform and since this method does not need any feedback, it is very robust. This method is one of window-based techniques which is developed and implemented on Asterisk open source proxy and evaluated. In the following in parts III the implementation platform is introduced. In part IV the details of load control algorithm which is developed in this open source software is presented. Part V includes the results and evaluation of efficiency of implemented algorithm and finally part VI includes conclusion and topics of future works.

II. RELATED WORK

Many researches about the efficiency of SIP proxy server have been accomplished. Paper [1] deals with overload control methods in SIP proxy server and uses OPNET software for measuring throughput. In papers [2] and [3] SIP is practically implemented along with TCP and UDP transmission protocol and OpenSER is used to obtain efficiency results. Articles [4]

and [5] mention to window-based distributed method and combination of signal and window-based method, respectively. SIPstone [6] is series of benchmark in which various criterions are proposed for evaluating proxy server powers, redirect server and registrar in answering SIP requests. In [7] another benchmark is presented for measuring the effect of operating system, hardware configuration, database, and selected transmission layer on SIP efficiency. In [8] practical experiments are accomplished on four types of proxy implementation which are different in both thread management and memory allocation method. The results of these experiments show that the effective parameters in proxy efficiency could be classified in two parts: parameters related to protocol such as message length, length variability, and irregularity of excess load, and parameters related to the type of server implementation e.g. how to allocate sources of operating system to transactions. Also in [9] similar studies about the effect of operating system and type of proxy implementation on SIP efficiency are done. In [10] it is studied about the efficiency of signaling of SIP in establishing VOIP connections using JAIN SIP API and by considering the effect of call duration and call rate on the delay of connection establishment between two end-to-end user agents. In [11] the effect of delay of user's answer on SIP server's efficiency is analyzed by introducing a tool called SIPperformer. In [12] issues such importance of security cost in configuration which uses authentication and the effect of proxy being stateful or stateless and the protocol type of transmission layer on proxy's efficiency is studied by placing only one proxy between user agents. A cluster of researches is concentrated on the evaluation of SIP efficiency under various access technologies. For example in [13] the effect of transmission delay and package loss on W-CDMA link is surveyed by using a proxy. In [14] the delay in signaling of SIP in establishing IMS meetings is evaluated for different WiMax channels with different speeds. Also compression techniques for SIP messages are used in order to reduce the volume of SIP packages along with their transmission delay. Selection of transmission layer protocol is also influential in the efficiency of signaling of SIP. In [15] various options in selection of transmission layer protocol for SIP are surveyed qualitatively. In [16] the effect of deploying various transmission layer protocols, especially effect of window control mechanism in TCP on throughput and delay in connection establishment is evaluated. In [17] it is shown that despite the general perception in which the more common utilization of UDP than TCP is considered on account of the low processing excess load in the former, it is probable that unfavorable efficiency in TCP utilization is due to implementation manner of proxy.

III. INTRODUCING IMPLEMENTATION PLATFORM

In order to survey the efficiency of SIP proxy when facing overload, the simplest traditional topologies are used [11, 3 and 9]. In this topology, which platform is illustrated in Fig. 1, a central proxy inquires every connection made between parties. This model is usually used for studying about the destructive effects of overload on proxy. If the number of users that starts to call in a short duration of time exceeds proxy's capacity, it faces with overload. The overload which the proxy in this topology faces with is in the form of client-server and the methods that are presented to get away from it are local inevitable.

The second topology which is used to evaluate and apply overload control methods is a SIP trapezoidal structure which is shown in Fig. 2. In this topology, the messages of each sides of call are exchanged via two middle proxies. In this topology it is assumed that M transmitter or upstream proxies (e.g. M=1) make an overload in a destination (downstream) proxy by sending many call-making requests. The overload which the proxy in this topology faces with is in the form of server-server. In this form of overload a limited number of upstream proxies send a huge volume of traffic to a downstream proxy and leads it to face with overload. This topology is applicable wherever any users get service from their local service provider proxy. The capacities of upstream proxies are considered so that they do not face with overload during experiments. By the way for simplicity only one upstream proxy is used. Asterisk software and Spirent Abacus 5000 tester device are used for implementing proxy servers and user agents, respectively. The upstream server is a PC with INTEL Dual Core 2.8 GHZ processor and 4 GB memory and the downstream server is a PC with INTEL 1.8 GHz processor and 2 GB memory. Both servers uses version 6.3 of Linux CentOS as their operating system. We used Spirent Abacus 5000 device to create traffic with different transmission rates and various distributions. This device is used for different tests including interoperability, performance, scalability, and testing audio and video quality on IP networks. This production is able to test the efficiency and extensibility of tested proxy by producing hundreds to thousands of calls.

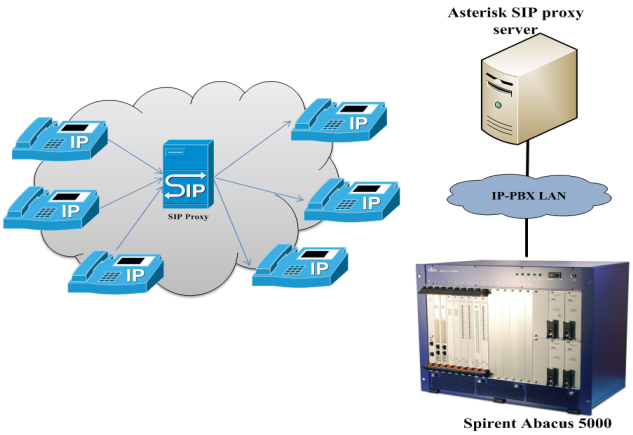

Figure 1. Single-Proxy Topology

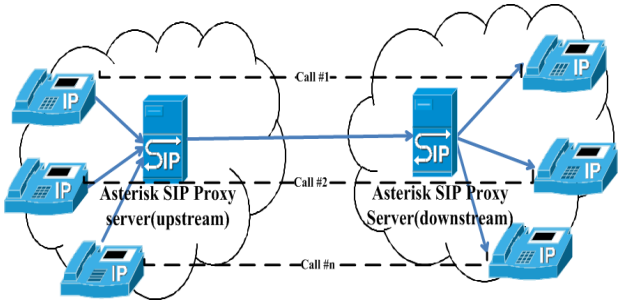

Figure 2. Dual-Proxy Topology

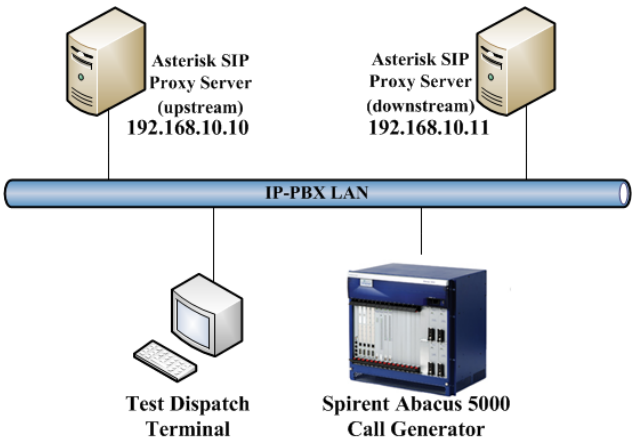

Figure 3. Setup Dual-Proxy Topology

IV. Overload Control Algorithm

The main idea in window-based methods is limiting the maximum requests waiting for replication in proxy. The procedure is that every transmitter proxy per every destination proxy keeps a timer of number of sent messages which replications have not been received. Sending request to destination proxy will stop as the number of packages which replications have not been received reaches a given amount which is called window length. The key issue in this algorithm is the window lengths which should be regulated through feedback in downstream proxy. Therefore it could be concluded that we can prevent overload through limiting window length [18]. In this algorithm the upstream server keeps a window of active transactions. New calls will be accepted if only the window has any empty space. When a transaction finishes, its delay is calculated and added to the list of recent transactions' delays; then the average of recent delays is calculated and if the amount of this average is more than multiplication of alpha by momentary delay, the magnitude of window will be equal to one and the window potential size, which we call WINth afterwards, reduces to half of the window size. The threshold of WINth accelerates the proceeding of window increase so that if the window size is lower than WINth, window size will be increase by one unit; and if it is higher than that, it will increase by amount of 1/W [21]. The pseudo-code of how window size will change during establishing a connection is as follows.

If $Zavg \geq Zth + \varepsilon \times \delta$ *then*

$Windowst+1=1$

Else if $Windowst < WINth$ *then*

$Windowst+1=Windowst + 1$

Else if $Windowst \geq WINth$ *then*

$Windowst+1=Windowst + 1/Windowst$

Z_{avg} is the average of delays in recent transactions and parameter Z_{th} describes the acceptable delay threshold. δ is the variance of delays and ε is a parameter for regulating acceptable variations of average delay about the given threshold Z_{th} and is considered equal to 3 practically [18]. Indeed Z_{th} is our conjecture about the delay in finishing transactions before occurring overload and is determined manually. The WINth parameter is the threshold of slow start. This parameter is initialized to half of maximum window size just before its complete close. In other words when the condition $Z_{avg} \geq Z_{th} + \varepsilon * \delta$ is satisfied, WINth will be equal to $Windows_t/2$ [14, 7].

V. Evaluation of Efficiency

In this section the efficiency of overload control algorithm is reviewed. Reports produced by Spirent Abacus 5000 device is used to check the time and type of sent and received messages by users. Also the reports of asterisk software are used to measure the status of progression of calls and transactions that occurred in proxy; and also oprofile software is used to measure the processing load of proxy. There are various criterions to determine the efficiency of SIP [19] amongst which in this study we concentrated on the delay of connection establishment (the interval between sending INVITE from UAC to receiving OK from proxy), resend rate, and proxy throughput (the number of successful calls per unit of time). First of all we will check the efficiency of Asterisk proxy in a case that applications are inbound (of one proxy) and establishing connection between them is achieved via one proxy. In the next part two proxies are assumed to be located between the parties, and the efficiency of algorithm presented in part IV will be studied on upstream proxy on account of the results of the last part.

A. Evaluation of Efficiency of Asterisk Proxy in Inbound (of One Proxy) Calls

As mentioned before, according to Fig. 1, signaling load is produced by Spirent Abacus 5000 device. Conversation production rate is started from the amount of as low as 100 calls per second and goes up to as high as 1200 calls per second, having Poisson distribution. In this case only one proxy considers call routing and no overload control mechanism is contrived on this asterisk proxy.

Fig. 4 shows the average call establishment time versus call request rate; call establishment time is defined as the interval between sending the first INVITE from meeting starter to time of receiving OK message. As it can be seen in this figure, the average call establishment time before received call rate reaches about 700 calls per second is inconsiderable and lower than tens milliseconds. So the capacity of the proxy is 700 calls per second. As received call rate approaches to proxy's capacity (700 cps), the average delay goes beyond 10 seconds.

The reason of call process procedure in proxy getting slow is that the amount of received call rate is beyond the processing capacity of the proxy and therefore proxy's sources are conjugated for analyzing and dissecting new call requests. Besides that, the flow of requests of new calls eventuates to overflowing of received queue and losing the packages related to ongoing calls. As a result, users whose requests have remained unanswered or the progression of whose call establishment have remained incomplete, proceeds to resend their messages. On the other hand, the proxy itself because has not seen some of received reply packages which are missed

through overflowing of queue or have not been checked yet because of proxy being busy, spends section of its capacity to resend requests which have been sent already. This procedure continues until proxy's throughput falls to near zero. The diagram of proxy's throughput in terms of call request rate which is shown in Fig. 5 proves this subject clearly. For example for rates higher than 800 cps, the magnitude of throughput is practically negligible. Note that this diagram shows the rate of calls that have started successfully in unit of time. Fig. 6 shows resend rate for INVITE requests made in user side. As expected, no request is resent before received call rate reaches proxy's capacity. But upon reaching received call rate to proxy's capacity, resend rate increases abruptly and intensifies the load imposed to proxy (Fig. 7) considerably. Fig. 7 shows amount of load imposed to proxy in terms of call request rate. The amount of imposed load is calculated as the ratio of processing cost of each call to its expected processing cost in case of non-overload.

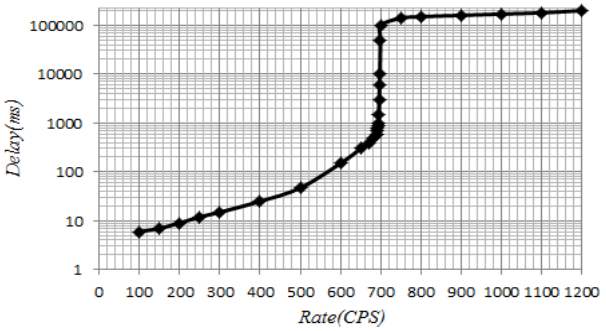

Figure 4. the delay of call establishment in the case of single proxy

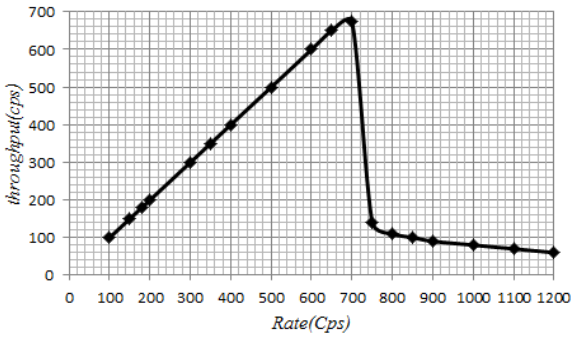

Figure 5. proxy's throughput in the case of single proxy

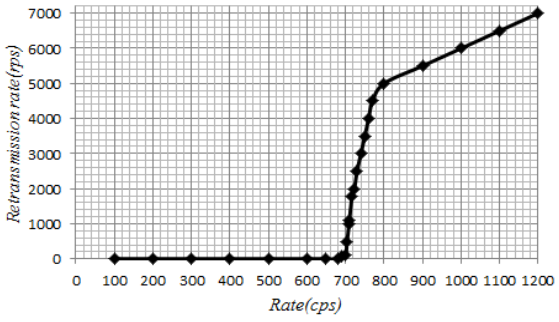

Figure 6. INVITE message resend rate in the case of single proxy

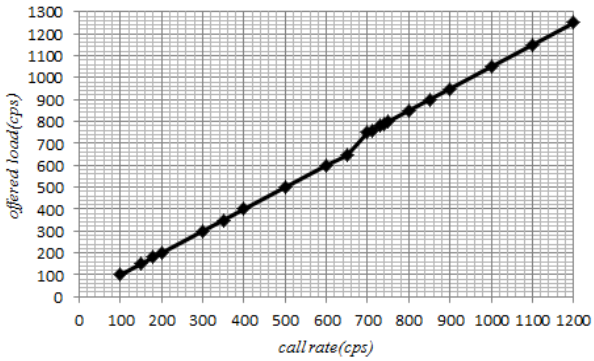

Figure 7. amount of load imposed to proxy

The cost of each call, which includes required time for dissection of its related packages, regulating timers, and creating and eradicating state for its related transactions, increases as load rate increases; because in case of overload, as a result of resend phenomena, the number of packages which is created and dissected per each call, along with number of timers that are regulated and reset increases. Although the number of packages that are received and dissected are related to size of queue too; because if the size of queue is small, many packages are missed and there is no need to be processed in proxy. Therefore as cost of each call increases, the amount of load which each proxy should process is much more than the load rate that is imposed directly by excess calls. The reason of sudden fall in proxy's throughput in load rates a little more than 700 cps is this too. For example if the cost in non-overload case for each call is 5 ms, in case of overload may increase to 6 milliseconds. So imposed load to proxy is like as being 1.2 times of the rate of its received calls.

B. Evaluation of Efficiency of Asterisk Proxy in outbound calls with two proxies

When calls are outbound, the rout of signaling is between the proxies related to each domain.

According to Fig. 3, signaling load is produced by two UAS and UAC user agents, both of which role is played by Spirent Abacus 5000 device. Conversation production rate starts from low amount and continues to heavy rates of about 1800 cps. On account of the results of last section, the capacity of downstream proxy under load in platform of experiment in Fig. 3 is set to 700 cps, link delay and probability of package loss are negligible generally, and magnitude of δ is considered 1.2 [21].

Fig. 8 shows the throughput as a function of rate of received call requests in case of existence and non-existence of overload control method, which represents that proxy's throughput could maintain at its maximum capacity in case of existence of overload control mechanism. As it is shown in Fig. 8, using overload control mechanism, the upstream proxy is able to maintain its throughput up to about 1500 cps, which is about twice as capacity of downstream proxy, whereas if overload control algorithm is not used, upstream proxy's throughput would be approximately equal to the one related to downstream proxy (700 cps).

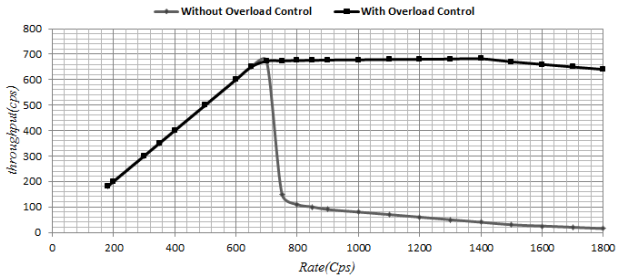

Figure 8. Throughput in case of existence and non-existence of overload control mechanism

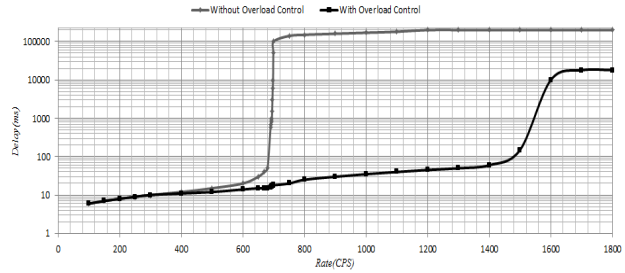

Figure 9. average delay of call establishment in presence and Absence of overload control mechanism

The size of window increases along with receiving new requests, so does delay. When the average of resulted delays is lower than multiplication of δ in momentary delay, size of window reduces to one and WINth reduces to half of the size of window and then starts to increase again. As it is shown in Fig. 8 and 9, this makes upstream proxy, in which overload control mechanism is implemented, maintain its throughput on about twice of downstream proxy's capacity, and also increases the average time of call establishment in this proxy to about 1500 cps linearly and with a growth rate far much lower than the case in which the overload control mechanism is not used. In rates higher than downstream server's capacity (700 cps), the huge amount of received requests stimulate CPU sensor and therefore many calls are rejected. Sudden rejection of calls leads to many Ack packages reach proxy in a very short interval and therefore full the queue of proxy so that there is no place in this queue for the packages of answers related to ongoing calls. Missing of answer packages is a stimulation to activate resend mechanisms in both server and client which deteriorates the situation. So calls accepted in proxy in this status are accepted with a very long delay. The diagram shown in Fig. 10 illustrates resend rate for INVITE and BYE requests from user side, individually. As it is expected, when we use overload control mechanism in upstream server, resend rates of messages decrease considerably. Overload leads to loss of OK packages related to the passed calls. So the proxy is required to resend INVITE requests related to lost packages. In this case, increase of resend rate makes proxy spend much of its time on resending requests related to ongoing calls and therefore throughput rate of proxy falls considerably. Processing the abundant packages which exist in proxy's queue, delays the passes calls more and increases the resend rate in caller's side. Note that if the capacity of proxy's queue is high enough, additional requests are rejected before being expired and resent from user's side; on the other hand not losing OK packages related to the passed calls in this case leads to decrease in average time of successful call establishment and therefore there is no need for proxy to resend INVITE messages; so throughput will decrease lesser. As it can be seen in Fig. 11, proxy's queue is approximately empty before occurring overload, since every message is drawn out of queue and processed upon reaching to proxy. Although in overload conditions many packages are consistently waiting in the queue to receive service. Nevertheless, applying overload control method, we could decrease memory usage considerably and delay its exponential growth rate up to about 1500 cps. The diagram in Fig. 12 shows CPU usage in proxy in terms of call request rate. In this diagram the horizontal axis represents amount of load received by proxy and vertical axis represents the average CPU utilization. These amounts are achieved by regular sampling (every second) of CPU engagement times and averaging them. As was expected, in load rates lower than proxy's capacity, percentage of CPU utilization is proportional to amount of load to proxy's capacity. For example when there is no overload control mechanism, in about 350 cps which is half of the proxy's capacity, utilization of proxy's CPU is approximately 0.5 too, also in a rate of 630 cps which call establishment request rate is 90% of proxy's capacity (700 cps), utilization of proxy's CPU is 0.9. In loads greater than proxy's capacity which leads to overload, CPU utilization reaches 100%. Using overload control mechanism, we could decrease CPU utilization considerably and delay the exponential growth of CPU utilization to 100% to about 1500 cps. In this case, in call rate of 1500 cps - twice of downstream proxy's capacity (700 cps) - CPU utilization reaches 100%.

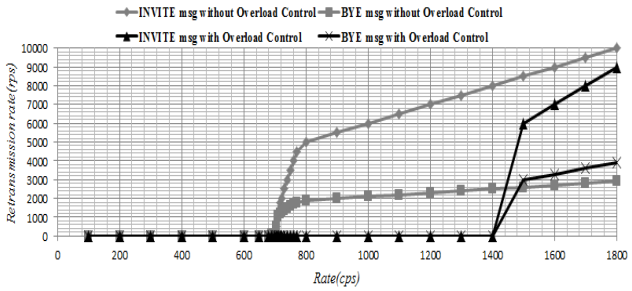

Figure 10. resend rate of INVITE and BYE messages in presence and absence of overload control mechanism

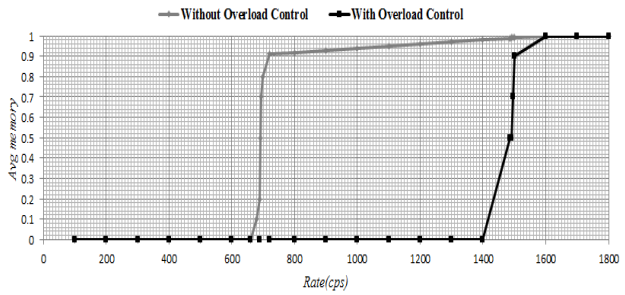

Figure 11. the average memory utilization in presence and Absence of overload control mechanism

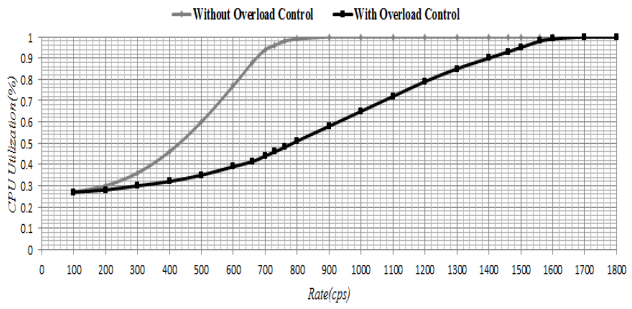

Figure 12. average CPU utilization in presence and Absence of overload control mechanism

VI. CONCLUSION

The studies accomplished in this paper show that SIP protocol is not efficient enough in facing with congestion, so that when call request rate increases, the delay of call establishment increases suddenly, proxy's throughput falls, and consequently resend rates and unsuccessful calls increase. In this paper window-based control method is developed, implemented, and tested on a real platform and also the efficiency of SIP proxy in case of overload is studied by using window-based distributed overload control method, which is developed on Asterisk open source proxy. Studying the charts of throughput, delay, and resend rate of INVITE and BYE messages in Asterisk proxy shows that the algorithm is able to maintain the throughput at up to twice of the downstream proxy's capacity. This is clearly observable in average memory and CPU utilization charts.

ACKNOWLEDGMENT

This work is supported by IP-PBX sample equipment verification laboratory of Ferdowsi University of Mashhad, morally and materially.

REFERENCES

[1] V. Hilt and I. Widjaja, "Controlling overload in networks of SIP servers," presented at the Network Protocols, 2008. ICNP 2008. IEEE International Conference on, 2008.

[2] C. Shen and H. Schulzrinne, "On TCP-based SIP server overload control," presented at the Principles, Systems and Applications of IP Telecommunications, 2010.

[3] E. M. Nahum, J. Tracey, and C. P. Wright, "Evaluating SIP server performance," SIGMETRICS Perform. Eval. Rev., vol. 35, pp. 349-350, 2007.

[4] V. Gurbani, V. Hilt, and H. Schulzrinne, "Session Initiation Protocol (SIP) Overload Control," 2012.

[5] M. Ohta, "Performance comparisons of transport protocols for session initiation protocol signaling," presented at the Telecommunication Networking Workshop on QoS in Multiservice IP Networks, 2008. IT-NEWS 2008. 4th International, 2008.

[6] H. Lindholm, T. Vähäkangas, and K. Raatikainen, "A Control Plane Benchmark for Telecommunications Signalling Applications," *Sort,* vol. 20, p. 100.

[7] S. V. Azhari, M. Homayouni, H. Nemati, J. Enayatizadeh, A. Akbari, " Overload control in SIP networks using no explicit feedback: A window based approach," presented at Computer Communications, Volume 35, Issue 12, 1 July 2012, Pages 1472–1483.

[8] M. Cortes, J. R. Ensor, and J. O. Esteban, "On SIP performance," *Bell Labs Technical Journal,* vol. 9, pp. 155-172, 2004.

[9] S. Wanke, M. Scharf, S. Kiesel, and S. Wahl, "Measurement of the SIP parsing performance in the SIP express router," *Dependable and Adaptable Networks and Services,* pp. 103-110, 2007.

[10] S. S. Gokhale and J. Lu, "Signaling performance of SIP based VoIP: A measurement-based approach," presented at the Global Telecommunications Conference, 2005. GLOBECOM'05. IEEE, 2005.

[11] S. Montagna and M. Pignolo, "Performance evaluation of load control techniques in sip signaling servers," presented at the Systems, 2008. ICONS 08. Third International Conference on, 2008.

[12] C. Chi, D. Wang, R. Hao, and W. Zhou, "Performance evaluation of SIP servers," presented at the Communications and Networking in China, 2008. ChinaCom 2008. Third International Conference on, 2008.

[13] D. Pesch, M. I. Pous, and G. Foster, "Performance evaluation of SIP-based multimedia services in UMTS," Computer Networks, vol. 49, pp. 385-403, 2005.

[14] M. Homayouni, S. V. Azhari, M. Jahanbakhsh, A. Akbari, A. Mansoori, and N. Amani, "Configuration of a sip signaling network: An experimental analysis," presented at the INC, IMS and IDC, 2009. NCM'09. Fifth International Joint Conference on, 2009.

[15] V. K. Gurbani and R. Jain, "Transport protocol considerations for session initiation protocol networks," Bell Labs Technical Journal, vol. 9, pp. 83-97, 2004.

[16] K. K. Ram, I. C. Fedeli, A. L. Cox, and S. Rixner, "Explaining the impact of network transport protocols on SIP proxy performance," presented at the Performance Analysis of Systems and software, 2008. ISPASS 2008. IEEE International Symposium on, 2008.

[17] Y. Wang, "SIP overload control: a backpressure-based approach," ACM SIGCOMM Computer Communication Review, vol. 40, pp. 399-400, 2010.

[18] D. Y. Hwang, J. H. Park, S. W. Yoo, and K. H. Kim, "A window-based overload control considering the number of confirmation Massages for SIP server," presented at the Ubiquitous and Future Networks (ICUFN), 2012 Fourth International Conference on, 2012.

[19] A. Abdelal and W. Matragi, "Signal-based overload control for SIP servers," presented at the Consumer Communications and Networking Conference (CCNC), 2010 7th IEEE, 2010.

[20] Y. Hong, C. Huang, and J. Yan, "Impact of Retransmission Mechanism on SIP Overload: Stability Condition and Overload Control," Journal of Networks, vol. 7, pp. 52-62, 2012.

[21] M. Homayouni, S. V. Azhari, H. Nemati, A. Akbari, "Controlling Overload in SIP Proxies: An Adaptive Window Based Approach Using No Explicit Feedback," presented at Global Telecommunications Conference (GLOBECOM 2010), Dec. 2010.

View publication stats