IRACST – International Journal of Computer Networks and Wireless Communications (IJCNWC), ISSN: 2250-3501
Vol.4, No3, June 2014

Overload Detection and Controlling Techniques in SIP Servers

A Survey

Kiran Kumar Guduru
Lead Engineer
Samsung R&D Institute Bangalore
Bangalore, India

Dr. Usha J
Professor, Dept of M.C.A
R. V. College of Engineering, Mysore Road
Bangalore, India

***Abstract*—The Session Initiation Protocol (SIP) is an application level signaling protocol used for locating end users, establishing, modifying and closing multimedia sessions between end users. This protocol has been adopted by 3GPP as a signaling protocol for IP-Multimedia Subsystem (IMS) networks. The usage of IP telephony is increasing rapidly, which indeed increases the load on the SIP servers. This paper gives an overview about different load controlling techniques applied on SIP servers to increase the through put and analyses the corresponding results of those works. It analyses the increase in throughput by detecting and controlling the overload in SIP servers using implicit overload control technique based on response delays and self overload control techniques based on FIFO queues simple Priority queues and priority queues with round robin selection.** *(Abstract)*

Keywords-SIP, Overload, Congestion, Queueing models.

I. INTRODUCTION

The usage of internet telephony is increasing rapidly with the experience of providing low price long distance calls and usage of multimedia communication sessions. SIP is a signalling protocol, majorly used in IP signalling in telecom industry. With the advantages of SIP signalling over the other IP signalling protocols, SIP is considered as the signalling protocol for IP signalling, by 3GPP.

With the increase in the usage of IP signaling in telecom industry, network servers are experiencing huge loads beyond the load which they can support with the increase in the number of users, and with the corresponding increase in the usage of IP telephony by each individual user, SIP servers are becoming overloaded. This gives raise to the need for detecting and controlling overload in SIP servers to protect them from congestion. Overloading of servers increases the request processing time and reduces the throughput of the servers. This increased processing time, when using UDP for transport, may result in unnecessary signal retransmissions, which indeed increases the load on the server.

This paper analyses the controlling of overload using different queuing models including simple FIFO queue, priority queue and priority queue with round robin selection using buffers for identifying the self-overload control of proxy servers and techniques for analysing overload in the network, implicit overload control, using round trip time delay values of SIP request response messages.

The delay incurred in processing a signal includes both transmission delay in a transport link and processing delay at each server through which a SIP signal traverses until it reaches its destination. Such types of delays are categorized as SIP signalling metrics [15]. The transport delay varies with different transport protocols used for transporting SIP signalling.

SIP allows using different transport protocols like UDP [7], TCP [8] and SCTP [9] which also influences on the signalling performance. SIP prefers UDP as its transport protocol [4] for carrying signalling messages. UDP is an unreliable transport protocol. So SIP ensures application level control for retransmitting signalling messages in order to achieve reliability.

The following sections gives the over view of SIP signaling followed by analyzing the work done using different queuing models and simulation environments to control the overload and identifying the overload in network using round trip time delays, and comparing the results of their outputs.

II. SIP SIGNALING

SIP is an application level signalling protocol. It is used for identifying the end users using registration and presence techniques. It employs messages for establishing, modifying and closing the sessions. It supports many types of messages for enabling features like messaging and multimedia sessions. A simple sip session consists of six messages namely Invite, 180 Ringing, 200 OK, Ack for establishing the call and Bye and 200 OK for disconnecting the call. Apart from this a 100 Tyring response may be sent for acknowledging Invite message. SIP does not carry any media related data but carries only metadata that gives the information about the underlying media session to be established.

SIP Invite message takes more processing time, which involves database query for identifying the end user location and forking of Invite requests to multiple proxies, when compared to that of other messages. Responses for SIP messages other than Invite requests should be immediate and should not take more time for giving response. Since smooth

continuation of existing sessions are given more priority for processing than that of establishing new sessions, Invite requests are given less priority when compared to that of other requests and responses.

The following Fig. 1 shows the basic session setup and teardown of session using SIP signalling messages. It consists of a SIP source User Agent, two proxies and a Destination User Agent, forming a SIP trapezoid.

Every SIP request has its corresponding response. A SIP request and its corresponding 200 OK response pair are termed as a successful SIP transaction. A SIP Invite transaction may contain more than one response, with one or more 1xx provisional responses. A SIP Invite transaction along with its corresponding Acknowledgement message is called as a SIP dialog. ACK message in continuation to a SIP invite request is treated as a separate transaction. For every failure response corresponding to an initiated request, client should send an ACK message. This ACK message that is sent in continuation to a failure response is treated as the same transaction of the previous request.

SIP protocol stack consists of four layers namely Transport, Transaction, Dialog and User Agent. A transaction initiated from client is called client transaction and that of towards server is called server transaction. Invite messages are used for establishing a new SIP session. BYE message is used for terminating an ongoing SIP session.

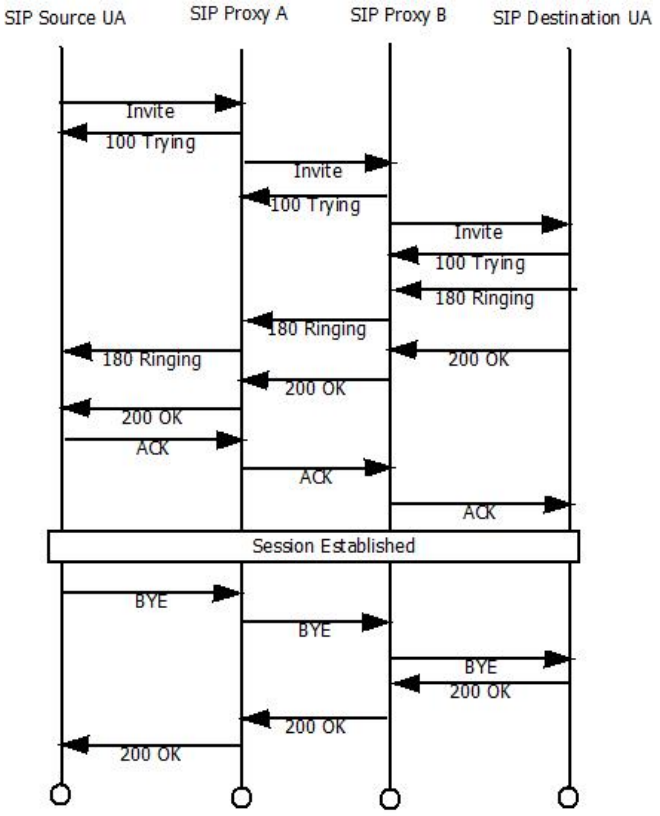

Fig. 1 SIP message exchange.

SIP requests received up to the transaction layer can be buffered, queued and processed, if more number of requests or responses is received at a SIP proxy, leading to overload.

SIP responses other than that for Invite requests are expected immediately be the client. If the response for a request takes more time than that of the specified threshold, since SIP is using UDP transport, user agent client or server assumes that the request has been corrupted or dropped in the network and retransmits the requests. This retransmission interval has been specified as 500ms after sending the first request and then onwards it doubles for the next retransmission until it reaches 32 seconds. If the response is not received until 32 seconds, then the request is considered to be timed out.

The following section explains about different overload detection and controlling techniques in SIP signaling.

A. Maintaining the Integrity of the Specifications

The template is used to format your paper and style the text. All margins, column widths, line spaces, and text fonts are prescribed; please do not alter them. You may note peculiarities. For example, the head margin in this template measures proportionately more than is customary. This measurement and others are deliberate, using specifications that anticipate your paper as one part of the entire proceedings, and not as an independent document. Please do not revise any of the current designations.

III. SIP Overload Detection And Control Techniques

Overload control in SIP network can be detected in two ways namely implicit overload control, which identifies the overload state of the network downstream, and self-overload control, which identifies the overload state of the proxy in question.

A. SIP Implicit Overload Control

Implicit Overload Control is the process of identifying the overload state of the downstream network from that server node. Overload control in SIP network can be identified by analysing the request response round trip times at a SIP proxy. In this method the response delay for corresponding requests, from the time the requests has been sent, are calculated over a period of time. These request response delays are specified as final response delay (fRD) [3]. If the average response delay over a specified period of time is less than a particular threshold, then the network is assumed to be in good state and the messages are passed to the same network nodes. If the average response delay for a specified period of time is more than that of the specified threshold, then the network is assumed to be congested and the corresponding action like redirecting the excess requests to other network nodes, if possible, or rejecting the excess requests through 503 service unavailable responses as specified in [4] are applied. In this mechanism the overload information of the network is not intimated to any other SIP nodes as that of in [2].

A 95 % percentile algorithm has been proposed as the threshold value [3]. According to this algorithm 95% of all the transactions need this value for fRD or below this value to complete successfully. After calculating 95% percentile values for the last m seconds, if the fRD values are more than that of expected, then the network is assumed to be congested and the outgoing transactions are limited to a particular number and the excess transactions, particularly Invite transactions which establishes new sessions are rejected through 503 error response.

B. SIP Self Overload Control

Self-Overload Control is the process of identifying the overload state of the same server node in the network. In Self Overload Control method, discussed in this paper, the number of transactions received per second in a proxy server is queued, after processing from transaction layer and before entered into user agent layer of SIP protocol stack, and then processed using different queuing models. The analysis includes queues of different types that include normal FIFO queues and priority based queues, depending on the time and priority based queues where the processing if SIP messages is done by selecting the messages in queues in round robin fashion.

In First in First out (FIFO) queuing model, all the requests, including Invite and non-invite requests are queued into a single queue and processed without considering any priority. SIP Invite requests will take more time for processing where as non-invite SIP requests are processed immediately. Since less priority Invite requests takes more processing time at the expense of high priority non-invite request's processing time, the probability for a proxy server to receive retransmissions of non-invite requests increases. This unnecessary retransmission will still increase the load on the server and reduces the throughput.

In the time dependent priority queuing model two queues are maintained. SIP Invite requests are processed into a low priority queue and remaining all the other types of requests are processed into a high priority queue. All the requests in the high priority queue are processed first. The requests in low priority queue, i.e., Invite requests are processed only if the requests in high priority queue are empty. This will reduce unnecessary request retransmissions of non-invite requests at the expense of reducing the establishment of new sessions, if the server receives more number of requests than that of the expected threshold.

In the other model priority based queuing models were simulated using two queues, one for low priority Invite requests and the other for high priority non-invite requests. The requests are processed and saved with buffer size of 49 and 105. The request – response messages in queues are processed in round robin fashion; to avoid the low priority queue to starve in case of overloaded state. This will reduce the number of terminated calls when compared to that of normal priority based queuing model. But the retransmissions of non-invite requests are more when compared to that of normal priority based queuing model. If the buffers are completely filled, the requests will be rejected. The queue sizes are in the ratio of the corresponding request types i.e., 1:6 where 1 is for less priority Invite requests and that of 6 is for high priority non-invite requests.

IV. COMPARATIVE ANALYSIS OF SIMULATION MODEL

The simulation model for controlling SIP overload using Implicit Overload Control consists of two sipp load generators which acts as user agent client and user agent server respectively and a SIP proxy server which transmits the SIP signals between client and server. The proxy server calculates the fRD for the requests transmitted in the last m seconds and calculates the 95% percentile over those response delays. If the resultant values are more than the specified threshold then the new requests received after a certain threshold is terminated.

The simulation model for controlling SIP overload using Self Overload Control contains n client SIP proxies with one proxy in each of n domains, shown as domain 0 to domain n-1 in Fig. 2 was setup using network simulator 2 (NS2). Each domain, out of the n domains, consists of m SIP clients which initiates the calls / sessions at regular intervals of time. These servers act as client servers that generate the load, with m calls per second, and repeats generating new calls after termination of old calls, at specified intervals of time. All the requests from these servers are redirected to destination domain n which contains n . m user agents. All the requests in Domain n are redirected through a single SIP proxy server and so expected to be bottlenecked. Fig. 2 shows the simulation model for simulating self overload control.

In this simulation model, if all the user agent clients invoke the requests at the same time, then the destination proxy receives n . m requests per second. If Ts is the time required for establishing a session, if the receiver answers the call immediately after receiving the ring back tone, considering the answer call time to be zero, and Ti is that of time elapsed before establishing a new session, then SIP signalling message arrival rate at the SIP proxy server in domain n is

$$\lambda = \frac{7 \cdot m \cdot n}{Ts + Ti}$$

Where n . m represents total number of SIP requests received per second, at peak load and Ts + Ti represents the time elapsed before a new call is established. With the increase in the duration of the session and the time interval between two sip requests that are generated by each SIP source, the number of SIP requests arriving at the expected bottle neck proxy will reduce. The values of Ts and Ti are set to vary exponentially with random values having an average of 30 seconds and 10 seconds respectively. If mu-sip is the service rate of SIP proxy server, then usage rate of bottlenecked SIP proxy server is defined as

$$\rho = \frac{\lambda}{\mu sip}$$

$$\rho = \frac{7 \cdot m \cdot n}{(Ts + Ti)\mu sip}$$

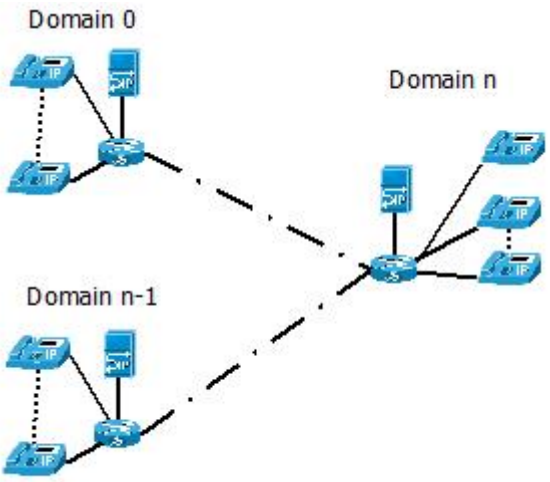

Fig. 2 Simulation model.

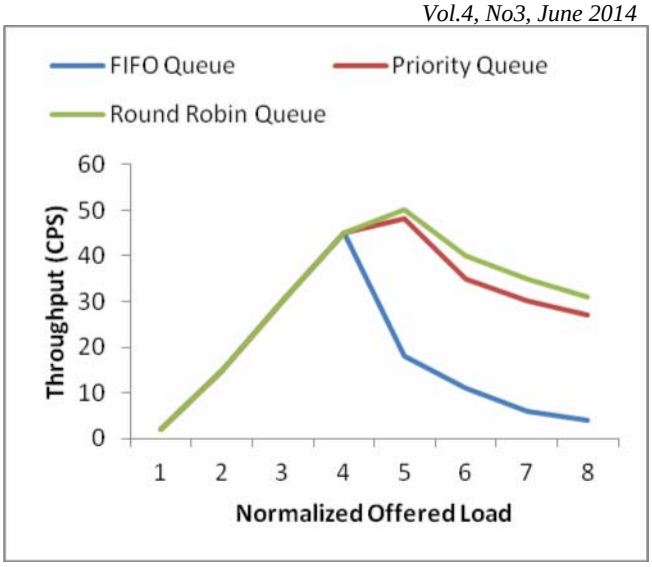

Fig. 3 Variation of throughput with offered load.

Assuming the capacity of SIP server to be 195 busy hour calls and maximum throughput of the SIP proxy is 55 calls per second, the offered load to proxy server n is calculated as

$$\rho = 0.06 X n . m X 10^{-3}$$

V. Results And Observations

From the simulation results it is observed that the throughput of the SIP proxy server increases linearly with the increase in the offered load. After reaching the threshold limit, at approximately 45 calls per second, the throughput of the server decreases drastically and reaches zero, vary faster, when using single FIFO queue whereas when using priority based queue, for both time based priority queues and round robin selection process queues with buffers, it decreases slowly and gradually but server does not breaks down like that when using FIFO queue model.

Following figure, Fig. 3 shows the relationship between offered load and throughput using FIFO queues, priority based queues and priority based queues with round robin selection schemes. In Fig. 3, X-axis represents normalized offered, ρ, load and Y-axis represents throughput in calls per second.

In Fig. 3, green line indicates throughput of SIP sessions, when the load is processed through round robin based priority queues. Red line indicates, throughput of SIP sessions, when the load is processed using time based priority queues. Blue line indicates the throughput of SIP calls that resulted when using FIFO queues.

The throughput of a server that implements to identify the implicit congestion detection, with 95% percentile algorithm is similar to that of a self-overload control algorithm with priority queue implementation. The throughput of SIP server without implementing congestion detection algorithm is similar to that of self-overload control server that implements FIFO queue for controlling the overload. The throughput of the SIP server has been increased by using 95% percentile congestion detection algorithm where as it is decreased without using the congestion detection algorithm. The final response delay and indeed the call establishment time have been stable even though the load on the SIP server has been increased by using the congestion detection algorithm, otherwise the final response time has been increasing with the increase in the input load. Retransmissions were also decreased using implicit congestion detection and control algorithm.

The rate of successful calls has been increased when using priority based queues when compared to that of FIFO queues. The average call setup delay is observed to be less with priority based queues when compared to that of FIFO based queues. The call setup delay using time based priority queues has been decreased by an average amount of 0.67 seconds when compared to that of FIFO queues.

A. Benefits

The call success rate is 6% more when using priority queues with round robin selection when compared to that of priority queues using time based selection. The through put of the priority queues with round robin selection is 26.4% more when using buffer sizes as 49 instead of 105 and the termination of calls increased from 83% to 80% respectively. 66% of terminated sessions in case of buffer size with 49 has Call Setup Delay (CSD) less than 10% and for that of buffer size with 105 it is 63.5%.

Observations show that implicit overload control algorithm that calculates the response delays and simple priority based

queues gives the same output. So it is better to implement implicit overload control in one server which can control the overload in all the servers downstream from it by calculating the response delays, instead of implementing self overload control algorithms in all the servers.

B. Limitations

The CPU time for processing simple priority queues is less than that of round robbing selection based priority queue, but the number of unsuccessful calls that were terminated has been increased when using priority queues when compared to that of round robin queue selection algorithm. The existing works missed to analyze the extra processing time incurred in implementing the self overload control algorithms and implicit overload control algorithms respectively. The queuing models that implements self overload control algorithm increases the extra processing time and thereby increases the retransmissions.

VI. CONCLUSION

The throughput of SIP servers reduces with the increase in congestion experienced by the server. The overload in SIP servers can be identified by using implicit overload control and self overload control algorithm. The controlling of overload in SIP servers which is identified using implicit overload control methods will be reduced using SIP metrics calculated on the basis of response times like fRD. The controlling of overload in SIP servers which is identified using self overload control will be reduced by employing different queuing models.

Priority queues that process messages using round robin sequence gives more efficiency when compared to that of simple priority queues and FIFO queues respectively. The call terminations also reduced by using priority queues when compared to that of FIFO queues or without using implicit overload control algorithms. Call setup delay is also reduced by using priority queues. Retransmissions of non-invite sip requests were reduced by using priority queues to detect and control self overload control. From the simulations it is observed that the through put of the SIP servers is more without using 503 error responses for terminating the calls when compared to that of using 503 error responses for terminating the calls.

REFERENCES

[1] Rosario G. Garroppo, Stefano Giordano, Stella Spagna, Saverio Niccolini, “Queuing Strategies for local overload control in SIP server” , IEEE GLOBECOM communications, 2009, ISSN: 978-1-4244-4148-8.

[2] Masataka Ohta, “Overload Protection in a SIP Signaling Network”, IEEE, 2006, ISSN: 0-7695-2649-7.

[3] Christoph Eggr, Marco Happenhofer, Peter Reichl, “SIP Proxy High-Load Detection by Continuous Analysis of Response Delay Values”, IEEE, 2011.

[4] J. Rosenberg, H. Schulzrinne, G. Camarillo, A. Johnson, J. Peterson, R. Sparks, M. Handley, E. Schooler, “SIP: Session Initiation Protocol” IETF RFC 3261, 2002.

[5] Kiran Kumar Guduru, Dr. Usha J, “Controlling Overload in Networks with SIP Redirect Servers”, International Journal of Advanced Research in Computer Science and Software Enginnering, Volume 3, Issue 11, November 2013, ISSN: 2277 128x.

[6] Kiran Kumar Guduru, Usha J, “Techniques for Reducing Network Congestion in Telecom Signalling Networks”, International Conference on Future Computig, 2012, Narosa Publications, ISBN: 978-81-8487-271-2.

[7] J. Postel, “User Datagram Protocol”, IETF RFC 768, 1980.

[8] Jon Postel, “Transmission Control Protocol”, IETF RFC 793, 1981.

[9] R. Stewart, K. Morneault, H. Schwarzbauer, T. Taylor, I. Rytina, K. Kalla, L. Zhang, V. Paxson, “*Stream Control Transmission Protocol”*, IETF RFC 2960, 2000.

[10] J. Rosenberg, H. Schulzrinne, P. Kyzivat, “Caller Preferencesfor Session Initiation Protocol (SIP)”, IETF RFC 3841, 2004.

[11] Volker Hilt, Indra Widjaja, Holmdel, Murray Hill, “Design Consideration for SIP Overload Control”, IETF RFC 6357, 2008.

[12] M. Lulling, J. Vaughan, “Reliability and Congestion Control for VOIP Signaling Transport”, Proceedings of the 5th WSEAS international conference on Applied Computer Science, 2006, ISBN: 960-8457-43-2.

[13] Pavel O. Abaev, Yulia V. Gaidamaka, Alaxander V. Pechinkin, Rostislav V. Razumchik, “Simulation of Overload Control in SIP Server Network”, Proceedings 26th European Conference on Modelling and Simulation.

[14] Sergio Montagna, Maurizio Pignolo, “Performance evaluation of Load Control Techniques in SIP Signaling Server”, Third International Conference on Systems, IEEE, 2008, ISSN: 978-0-7695-3105-2.

[15] D. Malas, A. Morton, “Basic Telephony SIP End-to-End Performance Metrics”, IETF RFC 6076, January 2011.

AUTHORS PROFILE

Kiran Kumar Guduru born on 10/01/1985 in Nellore obtained his B.Sc (MPC) and MCA from Sri Venkateswara University. He has over four years of experience in Research and Development and is working as a Lead Engineer in Samsung R&D Institute India - Bangalore. He is persuing his Ph.D from Visveswaraya Technological University.

J. Usha born on 03/09/1972 at Bangalore obtained her B.Sc (PCM) and MCA from Bangalore University, has over fifteen years of experience in teaching and is working as a Professor in the Department of MCA at R. V. College of Engineering, Mysore Road, Bangalore. She obtained her Ph.D from University of Pune in 2011. She is a member of IEEE, CSI, and ISTE.