EPH - International Journal Of Science And Engineering
ISSN (Online): 2454 2016
Volume 04 Issue 03 September 2018

DOI: https://doi.org/10.53555/ephijse.v4i3.177

SUPPLY QUALITY SERVICE VOIP THROUGH CALL SIGNALING OVERLOAD CONTROL

Ghusoon Alabbas[1*], Seyed Amin Hosseini Seno[2]
[*1]*Department of Computer* - Al Muthanna University.
[2]*Department of Computer* - College of Engineering - Ferdowsi University.

****Corresponding Author:***
Email: ghusoonalabbas@gmail.com

Abstract:-

SIP (session initiation protocol (is an application layer protocol to create, maintain and ending the multimedia sessions. These protocols have no suitable mechanism for encountering with the overload, so when a server is overloaded, it`s effectiveness is reduced and eventually server's real throughput reaches zero. To control the overload, various methods have been proposed, but there are some basic challenges presented in presented methods that lead to poor performance while there is an overloaded server in the network. The current study focusing on server – server's overload, a distributed overload control method, based on dynamic window and implicit feedback is proposed to control overload. In order to extend the originally proposed algorithm which uses the violation of the calling delay from a specific threshold as an incentive for reducing the sending load to downstream proxy, adaptive methods are proposed. The performance of the proposed algorithms has been evaluated.

Keywords:- *Session Initiation Protocol, Asterisk Proxy, overload control.*

1. INTRODUCTION

SIP is a text-based protocol which is very similar to HTTP (Hyper Text Transfer Protocol) and that is based on request and response. The SIP has a signaling task in a multimedia communication; it means the task is creating a session of the features changes and the completion of the current session, SIP does not carry any data. After communicating, the session data that can be real-time data such as voice, video, etc. Directly and without interference, SIP agents communicate between the communicating parties. Data transfer is usually done by RTP (Real-time Transfer Protocol) which is a standard protocol for the transmission of real-time data, [1].

SIP-based network consists of two types of logical entities, called User Agent (UA) and Server Agent (SA). Its user agents are divided into two groups, the user agent client (UAC) and user agent server (UAS) which are the end users (communication parties) and send requests and send replies respectively. Servers are logically divided into several categories, the most important of which are registrar servers that are responsible for registering and storing user information, and proxy servers that are responsible for routing messages between end users, [2&3].

When overload occurs in the SIP server that arrival rate requests to one of the SIP servers is higher than its capacity. Sources can include processor processing capacity, memory, bandwidth, and so on, when the server is under an overload, it`s performance decreases severely, to the extent that it`s useful throughput is zero, [4, 5 & 6].
Although this problem is very important for all calls, but it is more important for emergency calls, [7].

SIP protocol has its own mechanism to deal with the overload is send a 503 Service Unavailable response message to reject new requests. This is done in the application layer and take place after receiving INVITE requests. But the cost of the mechanism embedded in the SIP protocol to deal with overload, is to send a response message to reject new requests, is comparable to the cost of accepting a request. Since UDP has little overhead, it's not necessarily possible for all SIP users to use TCP.
Given that the SIP overload control mechanism is inefficient, methods must be provided to diagnose and prevent SIP overload. In this paper, we provide a method for effective control of overload. The proposed method consists of two algorithms. In the first algorithm, the threshold is set to delay the response from the downstream server. The load sent to the downstream server is controlled by this threshold. In the second algorithm, this threshold value is obtained in an adapted manner with the network conditions.

2. Methodology

In this section, we propose a method for controlling overload. In its design, it has been tried to minimize overhead and additional complexity for the server under overload.
In the proposed approach, the upstream proxy maintains for each downstream proxy a dynamic window of transactions on stream which sent to the downstream proxies, and continuously monitors their completion delay by the monitoring unit. Accepting and sending a transaction for a new call depends on the availability of empty space in the window. Upon completion of signaling for a transaction, in addition to being empty its location in the window, the window space is also updated according to the average delay completion of the recent transactions, which is given as the output of the monitoring unit and feedback to the application unit.
Upon completion of the signaling for each transaction, if the average delay completion of the recent transactions is smaller than a certain threshold, the window size will increase, otherwise the window size will be reduced to one. From then on, at a slow start phase, similar to TCP, using the incremental task procedure, at the end of each transaction, the size of the window of a unit increases. This process continues until the size of the window size will be half the maximum of its previous value (before reducing to one). From now on, with the start of the phase of congestion avoidance and by following the incremental increase procedure, the size of the window will increase by one unit with completion of all of its transactions, in other words with completion of each transaction, $\frac{1}{W}$ will be added to its size. It should be noted that leads to justice the use of a policy of incremental increase and reduction task and it is an optimal policy [20]. Equation (1) shows how to resize the window by completing each transaction.

$$W_{t+1} = \begin{cases} 1, \ delay_{Average} \geq delay_{Thereshold} + \alpha \\ w_t + 1, \ w_t < SlowStart_{Thereshold} \\ w_t + \frac{1}{w_t}, \ w_t \geq SlowStart_{Thereshold} \end{cases} \quad (1)$$

Where $delay_{Average}$ the average recent transactions delay and the parameter $delay_{Thereshold}$ specifies the acceptable delay threshold, α is the delays variance. In fact, $delay_{Thereshold}$ is speculation about the delay completion of transactions in a non-overload mode and is determined manually. $SlowStart_{Thereshold}$ Parameter is slow start threshold. This parameter is set to half the maximum window size, just before its complete closure. In

other words, when the condition $delay_{Average} \geq delay_{Thereshold} + \alpha$ is established, $SlowStart_{Thereshold}$takes the value of $\frac{w_t}{2}$.

The basis of this proposed approach is knowledge of the standard delay threshold for INVITE transactions from upstream proxies. Therefore, the method works well by adjusting the value of this threshold and where the normal delay of transactions in a network is known, or in the case of a network is lack the link delay. Although the amount of this delay threshold is possible by using statistical analysis and the history of proxy's interaction in the network, but this question may arise that how does the standard delay value be determined before the acquisition of this history or at the beginning of the interaction of proxies? The delay in completion of transactions, or the return time that a proxy is upstream, is the time between sending the first transaction initiator request and receiving a response that will lead to call completion Time (CCT).

According to the above discussion, the proposed adaptive method is presented below. It should be noted that here the same process is used, and the setting trigger for the window size has changed. Equation (2) shows how to change the window size by using the adaptive algorithm.

$$W_{t+1} = \begin{cases} 1, \ delay_{Average} \geq CCT_{estimation} + \alpha \\ w_t + 1, \ w_t < SlowStart_{Thereshold} \\ w_t + \frac{1}{w_t}, \ w_t \geq SlowStart_{Thereshold} \end{cases}$$

(2)

$$SlowStart_{Thereshold} = \frac{w_t}{2}, \ delay_{Average} \geq CCT_{estimation} + \alpha$$

Where $CCT_{estimation}$ specifies the estimated delay in the normal state, which is obtained by averaging the CCT samples in non-overlapping intervals. $delay_{Average}$, is the actual average of delay k recent transaction, which is calculated with each run of algorithm. The coefficient α is considered also in order to take into account the acceptable changes around the estimated value, and its value is considered as an innovative one of 1/2. The adaptive method of proposed algorithm is presented as an improvement on the proposed algorithm, which the optimal delay threshold was given as the input parameter to the algorithm, the acceptable delay for making call dynamically and simultaneously with sending of messages to the downstream proxy was estimated and used.

3. Evaluation

In this paper, we used two powerful Asterisk and SIPp tools to implement SIP servers and users. Initially, in the form of two proxy's topology, the proposed overload control method is evaluated and compared with one of common overload control algorithms. The downstream proxy capacity is 200cps, and the upstream proxy is much quicker to ensure that it is not overloaded. In addition, the lower downstream has a queue length of 1000 packets. The tests in this section are performed in a state where the network delay is negligible (the delay of each link is about 0.5 ms). DTH is constant and is equal to 50ms. Each experiment was performed at an average rate of 200 seconds and the duration of the conversation is random variable with exponential distribution and with an average of 2 seconds. Proxy throughput graph based on the rate of incoming call requests is shown in Figure (1).

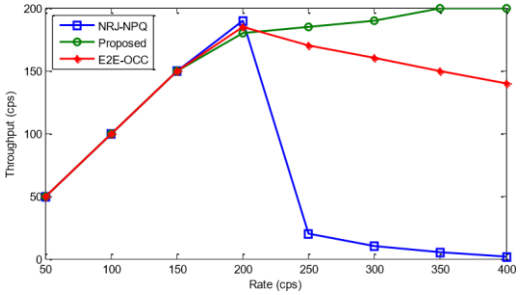

Fig. (1): Server throughput in the presence of the proposed method compared to E2EOCC and server without overload control

As shown in this Figure, proxy throughput in the overload conditions by applying the proposed method has reached its maximum and it does not drop. While applying local control at high amounts of loads will cause a significant drop in throughput.

Figure (2) shows the average call delay for loads of 50cps-400cps. As shown in this Figure, the delay of making calls through the use of the proposed method is considerably less than that of a proxy that has no control mechanism or E2E-OCC applied to it.

Because the amount of load passed to the destination proxy in the upstream proxy is limited and adjusted. As a result, none of the calls that are accepted in the upstream proxy will be lost.

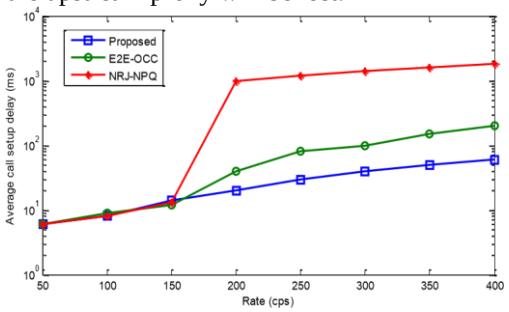

Fig. (2): Average call time in the presence of the proposed method compared to the E2EOCC and the server without overload control

The optimal use of the proposed method with a constant delay is due to the right choice of threshold value. This value is dependent on factors such as the speed of the downstream proxy service and the delay of the network and can be given as input by the network operator to each SIP network proxies. This section experiments with fixing maintenance of the network topology and its low delay examines the effect of DTH value on proxy performance. DTH values in these experiments are 50, 100, 300, and 900 milliseconds. Choosing the correct DTH value will have a significant effect on the amount of throughput. As shown in Fig. 3, considering a value of 900ms for DTH will cause the average delay in making calls in overload conditions greater than 500ms (T1 value). When we allow traffic to be sent to the downstream proxy, the queue delays increase the amount of retransmission timer, it causes the retransmission mechanism to be active in both the upstream and downstream proxies, thus, part of the proxy resource is used to process duplicate packets and handle packets retransmission. The drop of the graph in Figure (3) for DTH = 900ms is due to this reason.

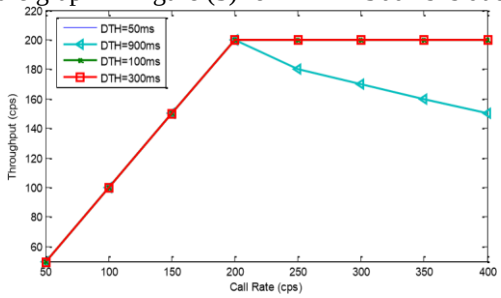

Fig. (3): Effect of the DTH parameter value on throughput of the proposed method

Figure (4) shows the average call time for DTH different states. As expected, whatever the value of this parameter is lower, the delay in making calls is also lower. Because, as soon as reaching the average delays to the DTH threshold, the proposed method is stimulated and reduces the window size. In this way, the amount of load sent to the downstream proxy is reduced, so that the amount of delay will not exceed the desired level.

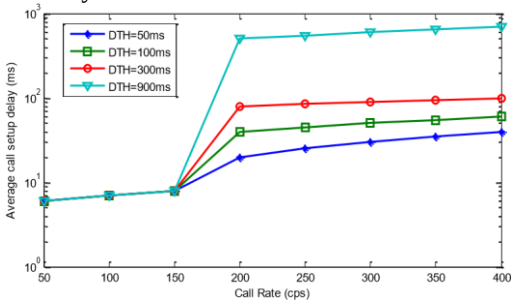

Fig. (4): Effects of the DTH parameter value on the average call delay

The graphs in Figure (5) compare the effect of applying an adaptive algorithm in two modes of Dlink = 0.5, 50ms with the results of applying E2E-OCC in terms of the amount of throughput at different load rates. As shown in Figure (5), the proxy throughput by applying adaptive algorithm in both network delay mode is greater than throughput in the E2E-OCC mode, that some of these differences returns to adjust the amount of CPU utilization in the algorithm E2E-OCC which causes the throughput value does not exceed a certain level and the other value appears at the rate of loads and delay of the higher links which is due to the high fluctuations of CPU utilization and receipt with delay feedback in the upstream proxy, which both make it difficult to adjust the amount of load sent to the proxy under load.

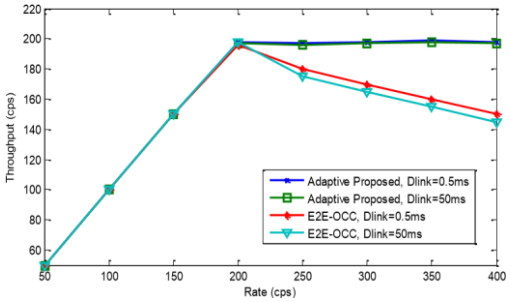

Fig. (5): comparison of throughput with the applying of adaptive algorithm and E2E-OCC in different Dlinks

4. Discussion

Algorithms based on the length of the queue and algorithms based on the occupancy rate of the processor are the most well-known methods in this field.

When the queue length is lower than the threshold, all requests are accepted, and by crossing the queue size from thresholds, all requests are rejected until the queue size reaches the threshold, [5].

The forward method was improved by adding a state between them and possible rejection of requests with respect to the queue length, [8], it has been shown that the old method, [5] is unsustainable due to the rejection of all requests during overload and accepting all requests when exiting the overload.

A combination of the two forwarded methods [5&8], with the allocation of the highest priority to non-INVITE messages and subsequent priorities is for the first to the seventh time for INVITE, respectively. In processor utilization-based methods known as OCCs, [9] a desirable amount is determined for processor utilization, with a violation of it, the proxy enters the overload phase and begins to reject additional calls [4&10].

However the authors compared two queuing-based algorithms and processor occupancy rates, and has shown that the method based on the processor occupancy rate has lower performance. Of course, it should be noted that in OCC methods caused the server does not reach its maximum throughput, and, on the other hand, instability server and its throughput and non-responsive quickly to reducing overload is another disadvantage of these methods, [10].

A team of authors presented four different algorithms which proxy continuously estimate their current responsiveness and inform the upstream proxies as window length or send rate, [7].
At the same time four different algorithms have been tested, the first algorithm is an improved version of the Retry-After mechanism in the SIP, the second and the third are two rate-based algorithms, which are based on the amount of processor occupancy and queue delay, respectively, [11].

It has been shown that the third one works better than the second one, the fourth algorithm is a sender-based window based method that uses the sender to get the Trying response to increase and no response and to get the 503 code to reduce the window. The window-based method is better than the other three algorithms.

Researchers introduced a method that uses four criteria of service time, database response, queue delay, and latency of the Trying response for overload detection. The author's argument is that the use of multiple criteria is better than a criteria for the detection of overload, [12].

Workers in the field of communication provided a hybrid method that uses a rate-based method [7] for a distributed algorithm. But the method of estimating the available capacity in [7] has slightly improved. It has shown that this method gives better performance, [13].

Controlling the rate of resend by upstream to reduce the effects of overload on light and temporary overloads was studied, the idea is that during temporary overloads, there may be no need to reject calls, and can be used to reduce the blocking rate and improve performance without rejecting calls, [14].

An implicit window-based method that uses delayed call as a measure for the detection of overload was provided, it has been shown that this method works better than the end-toend OCC [4] method because the OCC due to the nature of its feedback is constantly fluctuating in CPU performance and resulting fluctuating throughput, [15].

The rate of receiving 503 and the expiration rate of requests was used, that is, a combination of explicit and implicit methods, this method consists of three sets of reducing task, increasing task and linear increment. In these ways, there is still a problem of explicit feedback, [16].

A sender-based method with explicit response using received 503 messages was highlighted , comparing its results with a local method and a step-by-step method, both based on processor occupancy rates, and has shown that in the two different topologies, the proposed algorithm yields better results. The reason for this is that in this method there are no disadvantages of OCC and local methods. But the disadvantages of this approach include the use of explicit response and the downstream server occupation in overload control, [17].

A method, in which the source server inserts messages for each destination into a separate queue and, according to the queue size for a destination, performs an overload control action for that destination, [18].

Providing a method based on implicit feedback that uses latency criteria for completion of transactions. Each second it checks the calculated latency in the previous 5 seconds. If 95% of transactions have a delay of less than 0.5 seconds, they are considered normal and add up to the maximum number of active transactions. Otherwise it will decrease this number. It is claimed that in this method the values given for the parameters increase the stability and non-oscillation, [19].
In this paper, a distributed overload control method is presented, based on a dynamic window and implicit feedback for overload control. This method adjusts the load amount sent to it in the upstream proxy. If any proxy or network administrator is aware of the conventional delay of making calls with network proxies, by placing this parameter as the input of the algorithm, it will be able, even in the case of an overload, expect to get the maximum possible performance of a network of SIP proxies.

5. Conclusion

In this paper, with a focus on server-server overload, a distributed overload control method, based on dynamic window, and implicit feedback is provided for overload control. This method adjusts the load sent to it in the upstream proxy by based on the criteria delay for making call. In order to generalize the initial proposed algorithm that uses violation call delay from a specified threshold as an incentive to reduce the load sent to the downstream proxy, adaptive methods have been proposed. The proposed methods in this paper, with advantages such as setting the sent load rate with the destination server's response speed, the end-to-end performance, the independence of the processing speed of the destination proxy, the independence of the modelling parameters, the elimination of sudden jumps in the load rate of input, and regardless of how the implementation of the proxy, they can keep proxy throughput with a good approximation around proxy capacity without suffering the least cost for the proxy under load.

References

[1]. Rosenberg, J., Schulzrinne, H., Camarillo, G., Johnston, A., Peterson, J., Sparks, R., Handley, M. and Schooler, E., 2002. SIP: session initiation protocol (No. RFC 3261).

[2]. Karapantazis, S. and Pavlidou, F.N., 2009. VoIP: A comprehensive survey on a promising technology. Computer Networks, 53(12), pp.2050-2090.

[3]. Shen, C. and Schulzrinne, H., 2010, August. On TCP-based SIP server overload control. In Principles, Systems and Applications of IP Telecommunications (pp. 71-83). ACM. [4] Hilt, V. and Widjaja, I., 2008, October. Controlling overload in networks of SIP servers. In Network Protocols, 2008. ICNP 2008. IEEE International Conference on (pp. 83-93). IEEE.

[4]. Ohta, M., 2006, March. Overload control in a SIP signaling network. In Proceeding of World Academy of Science, engineering and technology (pp. 205-210).

[5]. Azhari, S.V., Homayouni, M., Nemati, H., Enayatizadeh, J. and Akbari, A., 2012. Overload control in SIP networks using no explicit feedback: A window based approach. Computer Communications, 35(12), pp.1472-1483.

[6]. Shen, C., Schulzrinne, H. and Nahum, E., 2008. Session initiation protocol (SIP) server overload control: Design and evaluation. In Principles, systems and applications of IP telecommunications. Services and security for next generation networks (pp. 149-173). Springer, Berlin, Heidelberg.

[7]. Yang, J., Huang, F. and Gou, S., 2009, September. An optimized algorithm for overload control of SIP signaling network. In Wireless Communications, Networking and Mobile Computing, 2009. WiCom'09. 5th International Conference on (pp. 1-4). IEEE.

[8]. Joe, I. and Lee, J., 2012, January. An overload control algorithm based on priority scheduling for sip proxy server. In Proceedings on the International Conference on Internet Computing (ICOMP) (p. 1). The Steering Committee of The World Congress in Computer Science, Computer Engineering and Applied Computing (WorldComp).

[9]. Montagna, S. and Pignolo, M., 2008, April. Performance evaluation of load control techniques in sip signaling servers. In Systems, 2008. ICONS 08. Third International Conference on (pp. 51-56). IEEE.

[10]. Noel, E. and Johnson, C.R., 2009, September. Novel overload controls for SIP networks. In Teletraffic Congress, 2009. ITC 21 2009. 21st International (pp. 1-8). IEEE.

[11]. Chentouf, Z., 2011, April. SIP overload control using automatic classification. In Electronics, Communications and Photonics Conference (SIECPC), 2011 Saudi International (pp. 1-6). IEEE.

[12]. Garroppo, R.G., Giordano, S., Niccolini, S. and Spagna, S., 2011. A prediction-based overload control algorithm for sip servers. IEEE transactions on network and service management, 8(1), pp.39-51.

[13]. Hong, Y., Huang, C. and Yan, J., 2010, December. Mitigating sip overload using a control-theoretic approach. In Global Telecommunications Conference (GLOBECOM 2010), 2010 IEEE (pp. 1-5). IEEE.

[14]. Homayouni, M., Jahanbakhsh, M., Azhari, V. and Akbari, A., 2010, April. Overload control in SIP servers: Evaluation and improvement. In Telecommunications (ICT), 2010 IEEE 17th International Conference on (pp. 666-672). IEEE.
[15]. Abdelal, A. and Matragi, W., 2010, January. Signal-based overload control for SIP servers. In Consumer Communications and Networking Conference (CCNC), 2010 7th IEEE (pp. 1-7). IEEE.
[16]. Liao, J., Wang, J., Li, T., Wang, J., Wang, J. and Zhu, X., 2012. A distributed end-to-end overload control mechanism for networks of SIP servers. Computer Networks, 56(12), pp.2847-2868.
[17]. Wang, Y., 2010, August. SIP overload control: a backpressure-based approach. In ACM SIGCOMM Computer Communication Review (Vol. 40, No. 4, pp. 399-400). ACM.
[18]. Egger, C., Happenhofer, M. and Reichl, P., 2011, September. SIP proxy high-load detection by continuous analysis of response delay values. In Software, Telecommunications and Computer Networks (SoftCOM), 2011 19th International Conference on (pp. 1-5). IEEE.
[19]. Low, S.H., Paganini, F. and Doyle, J.C., 2002. Internet congestion control. IEEE control systems, 22(1), pp.28-43.