International Journal of Recent Technology and Engineering (IJRTE)
ISSN: 2277-3878, Volume-8 Issue-3, September 2019

Overload Control with Hybrid SIP Signaling - Using Smart Reliability and Priority Queuing Strategies

Kiran Kumar Guduru, Usha Jayadevappa

Abstract*: Session Initiation Protocol is an IP based application level signaling protocol. SIP employs request retransmission for enabling reliability over UDP transport. This research work proposes a Hybrid SIP System that integrates the congestion control techniques from the proposed Smart Reliable SIP and Overload control using Advanced Smart Priority (ASP) Queuing Strategy. The proposed system reduces retransmissions and queuing delays in SIP servers. Smart Reliable SIP (SRSIP) implements the modified Selective Repeat ARQ protocol for enabling reliability of SIP requests without violating the SIP core functionality. SRSIP reduces the number of retransmissions. ASP Queuing Strategy prioritizes the SIP requests based on the retransmissions. The proposed system reduces the queuing delays for request processing at the downstream server and the retransmissions over the network. The Hybrid models integrates the features of both Smart Reliable SIP and ASP queuing strategy resulting in a system with almost zero retransmissions and reduced queuing delay for request processing.

***Keywords* : Network Congestion, Overload Control, Session Initiation Protocol, Queuing Strategies..**

I. INTRODUCTION

Session Initiation Protocol (SIP) [1] is an application level signaling protocol, used for controlling multimedia sessions and for communicating chat messages. SIP signaling has more advantages when compared to other IP based signaling protocols. So, 3GPP adopted SIP as signalling protocol for IP-Multimedia Subsystem (IMS) networks [2] and Voice over LTE (VoLTE) [3].

The rapid growth of smart phone usage and internet telephony spurred the usage of IP based audio / video sessions. SIP signaling is used for managing low cost long distance calls over Internet. Messaging applications like What sApp are enhancing the application environment to provide support for voice and video call sessions apart from the existing chat sessions [4]. SIP signaling network traffic increased with the increase in the usage of IP based multimedia applications resulting in network congestion. The observed congestion might be either in data channel between SIP entities or with in SIP servers which are experiencing SIP request load beyond their respective tolerable levels.

SIP supports transport protocols like UDP [5], TCP [6] and SCTP [7]. Initial versions of SIP [8] supports signaling over UDP transport only. To support the backward compatibility and to support the stateless proxy features, SIP prefers UDP as its default transport protocol [1]. UDP is an unreliable transport protocol. SIP empowers self-reliability using request retransmissions when employing UDP as transport protocol. SIP requests are retransmitted after the T_1 timer [1] for first time. The timer value for request retransmissions is doubled until the timer value reaches 32 seconds. 500ms is the default value for T_1. So SIP clients retransmit unacknowledged requests at the time intervals of 0.5 sec, 1.5 sec, 3.5 sec, 7.5 sec, 15.5 sec and 31.5 seconds. If the corresponding response for a request is not received after 32 seconds, then that request is considered to be timed out and dropped.

If the end to end network state is congested resulting in Round Trip Time (RTT) values higher than 500ms, then SIP client assumes that the requests might be either dropped or corrupted over the network and retransmit the requests (duplicated request). Duplicated SIP request retransmissions increase load on the network resulting in unnecessary congestion. The duplicated retransmission requests increases the queue lengths in the downstream servers, resulting in increased queuing delay for processing the requests. This catastrophe of congestion in SIP network system due to SIP signaling retransmissions demands the need for robust SIP system that reduces the retransmissions over network and the queuing delays in the downstream servers.

The slump in SIP application servers due to unwanted request retransmissions might result in exhaustive server outage comparable to other VoIP server blackouts, similar to Skype server crash [9][10] and WhatsApp server crash due to the immense request burst during the new year evening [11][12][13]. Skype server crash took hours to get back to normal state [14]. Graphs reported in online media [11][12], reveals that, WhatsApp experienced huge request bursts for multiple times during a short period of time. Y. Hong et. al [15] emphasized the impact of SIP retransmissions during overload control. D. Y. Yavas et. al [16] evaluated the effect of retransmissions using Markov Modulated Poisson Process and analyzed that the impact of retransmission on SIP servers are higher than that of request drops.

Manuscript published on 30 September 2019
* Correspondence Author
Kiran Kumar Guduru*, Mobile Protocols and Platforms department, Samsung Electronics, Bangalore, India. Email: kiran.guduru@samsung.com
Usha Jayadevappa, MCA department, R.V. College of Engineering, Bangalore, India. Email: usha.j@rvce.edu.in

Such hazards motivated us to suggest a SIP system, which can reduce the number of SIP re-transmissions over the network and can withstand from huge bursts of requests and process the requests with minimum queuing delays.

In this research work, we analyzed the reduction in retransmissions and reduction in Queuing Delay using the proposed SRSIP signaling system, the proposed HSIP signaling system and compared with the existing SIP congestion control techniques like RSIP signaling system [17], Advanced Smart Priority (ASP) Queuing Strategy and FIFO Queuing Strategy. SRSIP reduces the SIP retransmissions by modifying the retransmission timer window, through continuous evaluation of RTT times and the estimated processing delay information received from the downstream SIP server. HSIP signaling system comprises the features of SRSIP and ASP Queuing Strategy to provide an optimized SIP system that works with least possible retransmissions and queuing delays. The proposed SRSIP and HSIP systems can be enabled by including extra parameters to SIP Via Header similar to that of in SIP overload control [18], without modifying the SIP core functionality.

In the following of this paper, section 2 provides the detailed literature review. Section 3 identifies and formulates the problem from the existing literature. Section 4 provides the working of Smart Reliable SIP signaling and analyzes the reduction in retransmissions during request drops and demand burst conditions. Section 5 details about the ASP Queuing Strategy by analyzing the reduction in queuing delays. Section 6 describes the working of Hybrid SIP (HSIP) signaling and analyzes the reduction in retransmissions and queuing delay. Section 7 discusses the detailed analysis of simulation results in various scenarios like demand burst condition, server slowdown and validates the same by comparing with the theoretical results, followed by conclusion in section 8.

II. RELATED WORK

SIP is an application level signaling protocol used for controlling IP based multimedia sessions like messaging, video, voice etc. Inflation of smart phone usage spurred the usage of multimedia communications. These IP based multimedia communications incorporates operator enabled services like VoLTE services and third party downloadable or web applications similar to Skype, WhatsApp etc. The growing utilization of IP routed multimedia communications challenges for exceptional levels of reliability from signaling protocol like SIP. Reliability of SIP system includes signaling reliability [1], server reliability [19][20][21] and reliability during mobility [22]. In this research work, we proposed a technique for improving the SIP system reliability, by improving the signaling reliability through SRSIP signaling and server reliability through ASP Queuing Strategy.

SIP signaling enables application level request retransmissions to empower reliability of messages. The impact of SIP reqeust retransmissions were analyzed by Y Hong [15]. Authors obtained a stability condition for SIP application servers while employing FIFO queuing. Probability of SIP retransmissions were analyzed using Markov-Modulated Poisson process [23]. They illustrated the effect of retransmissions on SIP system performance. Authors pointed that increased retransmission probability due to short demand burst or slump in server processing capability during server maintenance period may result in server overload or even a server crash. Authors, through their research, proposed an algorithm to reduce retransmissions [24]. This algorithm keeps track of ongoing SIP transaction queue along with their respective retransmission timers. If the size of queue or the response times overreach certain threshold, the server recognizes overload state in the downstream network path and employs the corresponding control method like rejecting the new requests.

Overload state in SIP servers can be controlled using overload control techniques and by enabling reliability for SIP signaling.

A. SIP Overload Control

Overload state of a SIP server minimizes the SIP server performance and may even result in unwanted retransmissions [23]. Following Overload control techniques were illustrated in the literature, to control the overload state in SIP servers.

1) SIP Explicit Overload Control

In this method, the SIP server gathers the overload information of the downstream SIP servers explicitly and controls the load on the overloaded server by redirecting extra load to other non overloaded servers. SIP standardization defined an overload control (OC) technique [18] using piggybacking method. Overload state (capacity to process the SIP requests in an arbitrary duration) of downstream SIP server is added in the first Via Header of response. Based on this value, upstream server will redirect the requests to other servers, beyond the overload limit of the downstream server. M. Khazaei et.al [25] proposed a holonic - multi agent system for gathering and controlling the overload information of SIP servers. The holons will be controlled in hierarchical bases based on the regions. M. Jahanbakhsh et.al [26] proposed a stability condition for SIP overload between two SIP proxies using Lyapunov stability function. Authors derived a proportional integral derivative for performing overload control.

2) SIP Implicit Overload Control

Identifying the overload state of other SIP servers from the behavior of signaling metrics is termed as SIP Implicit overload control [20]. C. Egger analyzed response delays continuously to identity the overload condition of downstream SIP servers [27].

3) SIP Self Overload Control

In this method SIP servers identify the overload state of itself and takes the preventive actions to return back to the stable state. Accessible server resources like available memory, processing time and CPU usage etc., can be used to identify the overload condition in a SIP server. D. Sisalem et. al [28] examined considered CPU usage, memory and network buffer for tracking the overload condition in SIP server.

Different queuing models, like First-in-First-out (FIFO), request type based priority queues, were analyzed through analytical methods and simulation methods, for recognizing and controlling overload condition in SIP servers [18][20][27]. D. Y. Yavas et.al proposed a fluid-flow modeling [29] and priority based modeling [30] for processing SIP requests with a queuing system that gives higher priority for initial requests instead of retransmitted requests. However if the original requests are dropped in the network, retransmitted requests will starve in low priority queues for longer durations. So, we proposed higher priorities for retransmitted requests instead of original requests in the proposed method.

We proposed Smart Priority Queuing Strategy [31] for processing SIP requests that gives highest priority to retransmitted requests. This method is further improved using ASP Queuing Strategy that prioritizes SIP requests depending on the number of times the SIP request has been retransmitted. If the SIP requests are being retransmitted because of increased processing (queuing) delay in the server, then the older duplicated requests waiting in the process queue will be removed and a new retransmitted request will be enqueued in the immediate next higher priority queue. The dropping of duplicated requests that are generated due to increased processing delays at the server, will avoid needless processing of retransmitted SIP requests and transmitting them towards downstream servers. We derived the SIP server stability condition using Advanced Smart Priority Queuing method and compared it with the FIFO queuing method.

B. Reliable SIP Signaling

Enabling reliability for SIP signaling through alternative methods instead of request re-transmissions, will avoid unnecessary retransmissions to downstream network. M. Lulling et. al proposed Reliable Signaling Datagram Protocol (RSDP) [32], which is especially designed for VoIP signaling. RSDP uses a 12 byte independent header for acknowledging the receipt of datagram packet. These RSDP Acknowledgement messages may incur additional network traffic. Packet drops are handled by triggering retransmissions after RSDP window timers. But these retransmissions results in backward compatibility issues. H. Zheng and S. Wang studied the inconsistency in SIP signaling due to mobility and proposed a Chain-Based SIP signaling [22]. Hop by hop reliability for UDP signaling was proposed by Reliable UDP (RUDP) [33]. But still it is not useful for SIP, with the reasons detailed in Section 3.

III. PROBLEM FORMULATION

Real time experiences like huge request bursts, which may lead to massive initial SIP request retransmissions, and limitations of existing systems for reliably transmitting SIP signaling messages over unreliable UDP transport emphasize the need for a reliable SIP signaling system with reduced retransmissions. The problem for SIP request retransmissions has been identified from the existing literature and proposed a solution with the Hybrid SIP system, as follows.

Retransmission of SIP requests from upstream servers, while the original requests are waiting in the queues of downstream servers, results in duplicated retransmissions. Queuing delays in the downstream servers are reduced with ASP Queuing Strategy. Optimizing the ASP Queuing system by reducing SIP retransmissions will eliminate the system time required for en-queuing the retransmitted requests and deleting the old requests from lower priority queues. This optimization further reduces the queuing delays in downstream servers.

Even after enabling the overload controlling techniques [18][19][20][27][31] SIP is still retransmitting request messages to achieve signaling reliability. Overload control techniques are able to reduce the load and reduce the processing delays on the downstream servers. But, they are not able to reduce the request retransmissions. Considering the live incident of receiving huge request bursts, multiple times, by WhatsApp server on the New Year eve [11][12], these initial request retransmissions (that are received before processing the older request) will surge the needless load on downstream SIP server and might even give rise to server outage. Such live scenarios demand the need for a reliable SIP signaling system with negligible or no retransmissions.

Infeasible RUDP: SIP signaling systems may install Reliable UPD protocol [33] for enabling reliability. SIP servers can accomplish hybrid reliability at transport layer using RUDP. But the amount of installations of RUDP is negligible in live networks. Implementation of RUDP is based on sliding window protocols [34]. So, RUDP provides reliability for transmitting packets as well as in-order delivery of packets [33][34]. SIP signaling packets transmitted between servers are disjoint and have no dependency for in-order delivery. Buffering mechanisms enabled for achieving in-order delivery may store some requests in RUDP buffer resulting in unwanted transmission delay. So, SIP cannot employ RUDP as transport protocol to enable reliability.

SIP needs an independent reliable system that can be maintained by it-self to attain reduced retransmissions that ideally zeros retransmissions. RSIP signaling reduces retransmissions and also backward compatible with the existing SIP servers. But we can still observe SIP retransmissions. These retransmissions arise due to the lack of overload information of the downstream SIP server.

In this research work we proposed a Hybrid SIP signaling system that comprises of the proposed SRSIP signaling and ASP Queuing Strategy. SRSIP signaling eliminates the retransmissions by enabling reliability with the application level acknowledgements. SRSIP is designed to completely eliminate the SIP retransmissions except for SIP request and response drops. ASP Queuing Strategy reduces the queuing delay at the downstream SIP servers, by prioritizing the requests based on the number of retransmissions and dropping the duplicated requests.

IV. SMART RELIABLE SIP SIGNALING

Advances in the network transmission technologies reduced the packet drops in UDP networks. Early literatures [35] [36] identified that average back bone loss for UDP packets may vary from 0.01% to 6.94%.

During demand burst condition, the packet loss rate may reach up to a maximum of 16.98%, with an average of 9.97% during the longest burst [35]. Latest Literature illustrates that the maximum probability for packet drops in UDP networks is 5% [37][38]. Online internet traffic report [39] depicts that the UDP packet drops are approximately zero. D. Y. Yavas et. al [16] evaluated the effect of retransmissions using Markov Modulated Poisson Process and analyzed that the impact of retransmission on SIP servers are higher than that of request drops. From this we concluded to the hypothesis that the huge retransmissions in SIP systems are due to processing delays and network delays rather than that of packet drops. So we proposed Smart Reliable SIP (SRSIP) to reduce the retransmissions to the maximum possible extent that arise due to processing delays at the downstream server, server slow down and network delays. The remaining of this section describes the working of SRSIP signaling. Section 4.1 describes the usage and requirement of additional headers for existing SIP via header, for enabling the SRSIP signaling. Section 4.2 details the algorithm for SRSIP signaling. Section 4.3 explains the mechanism for handling request drops and response drops. Section 4.4 and 4.5 analyzes the reduction in retransmissions during request drops and demand burst conditions using SIP retransmission mechanism and queuing theory respectively.

The proposed SRSIP signaling enables hop-by-hop reliability for SIP signaling through application level acknowledgements. The acknowledgements for received requests are added to the existing SIP responses, that are transmitted between two SIP servers. SRSIP signaling will not add any new headers to SIP requests or responses. The SRSIP signaling method adds the following four parameters to top most SIP Via Header similar to that standardized by Sip Overload Control [40] and RSIP [17]. The behavior of 'seq', 'mack' and 'nack' parameters are similar to that of RSIP. This method defines a new parameter 'delay', to provide the processing delay information of downstream SIP server.

A. Via Header Parameters for SRSIP Signaling

This work defines the following four parameters for SIP Via Header. These parameters ensure reliability between adjacent SIP entities apart from providing information related to processing delays in of downstream SIP server.

- **seq:** This parameter should be included in the top most Via Header of SIP request, generated by SIP client. This parameter represents unique sequence numbers of the SIP requests. The unique sequence number starts with an arbitrary value, and is incremented by unity in every subsequent request.
- **mack:** This parameter should be included by the downstream SIP server in the top most via header of the SIP response. It represents the maximum value of sequence numbers received by the downstream server, till that time.
- **nack:** This parameter should be included by the downstream SIP server in the top most via header of the SIP response. This parameter represents the missing sequence numbers, if any requests are not received by the downstream server. For example, if a request with sequence number '2' is not received by the downstream server, then 'nack = 2'; will be added to the top most SIP Via Header of the next response.
- **delay:** This parameter should be included by the downstream SIP server in the top most Via header of the SIP response. Downstream SIP server should calculate the approximate time (in milli seconds) required for processing the requests and add that value to the 'delay' parameter of top most Via Header, in the next SIP response. For example, if the estimated delay value is 500 ms, then 'delay = 500' should be added to the top most Via Header.

Downstream SIP servers can consider the parameters like CPU usage, queue lengths, processing time of each request etc to calculate the delay values. In this research work we considered the queue lengths and request processing time for calculating the delay values.

B. Algorithm for Enabling SRSIP Signaling

The proposed Smart Reliable SIP (SRSIP) signaling employs modified selective repeat protocol for enabling hop by hop reliability for SIP system. Algorithm 1 represents the procedure for enabling SRSIP. In order to reduce the retransmissions and ensure hop by hop reliability, SRSIP considers Estimated Processing Delay (E_d) and Moving Average Round Trip Time (M_{rtt}) along with the default SIP retransmission timer. E_d is the processing delay at downstream SIP server, which is represented by 'delay' parameter of SIP Via Header.

In reality, network conditions and transmission speeds will be fluctuating [41]. Network fluctuations may be due to low signaling strength during mobility, noise, demand burst conditions [42], server slow down for some time [15], congested network link between servers etc. So, we considered M_{rtt} from the recent three round trip time values as shown in equation 1.

$$M_{rtt}(i) = 0.6R_i + 0.3R_{i-1} + 0.1R_{i-2} \quad (1)$$

Where $M_{rtt}(i)$ represents the moving average of i^{th} Round Trip Time R_i. M_{rtt} values are calculated similar to that of in [43] [17], based on Analytical Hierarchy Process [44]. From Analytical Hierarchy Process, sum of Random Consistency Index (RCI) for (6:3:1) is 1.82 and that of for (5:3:2) is 1.7 [44]. Since the factors influencing network speeds are highly fluctuating, priorities for R_i,R_{i-1},R_{i-2} are considered with highest RCI values, with the ratios of 6:3:1, instead of other low RCI values.

To enable SRSIP, client sends the sequence numbers in monotonically increasing order in the top most Via Header of SIP requests. Downstream server 300 sends the 'mack', 'nack', and 'delay' values in responses back to the client server. This mechanism enables Client Server to gather enough information regarding the request drops and processing delays in the downstream server.

Algorithm 1 illustrates the procedure for enabling SRSIP. UAC sends the SIP request with unique sequence number towards downstream SIP server and triggers the timer.

The value of timer is set to the maximum of either T_1 or $M_{rtt}(i)$ or E_d. Setting the retransmission window timer to the maximum calculated value differentiates the current research with the existing mechanisms and ensures the reduction in retransmissions and there by reduces the probability of retransmissions. If acknowledgement is received for the request, then T_{ret} is adjusted to the maximum of $(2 \times T_{ret})$ or $(2^j \times T_1)$. Where 'j' represents the number of times the request has been retransmitted. The value of j varies between 1 and 6, where 6 represent the maximum retransmissions allowed by SIP specification [1].

Algorithm 1 SRSIP

1: **procedure** SRSIP
2: **while** *Server is Running and Remote Peer Supports SRSIP* **do**
//Runs for ever to receive SRP responses from downstream network
3: UAC → Send requests with unique sequence numbers
4: T_{ret} ← max{ T_1, $M_{rtt}(i)$, E_d}
5: //$M_{rtt}(i)$ represents the moving average round trip time
6: //E_d is estimated processing delay received from downstream server
7: //T_{ret} represents the next retransmission timer
8: **if** *Received Ack for SIP Request* **then**
9: T_{ret} ← max{ $2 \times T_{ret}$, $2^j \times T_1$},
10: such that $2^j \times T_1 < M_{rtt}(i)$ & $2^j \times T_1 < E_d(i)$
11: ∀ 1 <= j <= 6 & *Maximize* j
//j represents the number of times the request has been retransmitted
12: **end if**
13: **if** Window *Timer expired* **then**
14: **if** *ack message received* **then** //response message dropped
15: UAC → Retransmit SIP request
16: //Sequence number must be updated
17: T_{ret} ← max{$2 \times T_{ret}$, $2^j \times T_1$, E_d},
18: such that $2^j \times T_1 < M_{rtt}(i)$ &$2^j \times T_1 < E_d(i)$
19: ∀ 1 <= j <= 6 & *Maximize* j
20: **else** //request dropped or no responses from UAS
21: UAC → Retransmit SIP request
22: //Sequence number must be same as that in original request
23: T_{ret} ← max{$2 \times T_{ret}$,E_d}
24: **end if**
25: **end if**
26: **if** SIP *Response message received* **then**
27: *Stop all timers and terminate the SIP transaction*
28: **end if**
29: **end while**
30: **end procedure**

If response is received before timer expiry, then all timers are stopped and the SIP transaction is closed. If the timer expires before the response is received, then it results in two cases. In first case, if the ack is received for the request, it indicates that the request has been received by the downstream server. In this case the UAC assumes that the response is either dropped or delayed. UAC retransmits the request by updating the sequence number and starts the retransmission window timer with the maximum of either $(2 \times T_{ret})$ or $(2^j \times T_1)$ or E_d, after ensuring that the value of $(2^j \times T_1)$ is less than that of $M_{rtt}(i)$ and E_d. In second case, if ack itself is not received before timer expiry, then the UAC assumes that the request has been dropped and retransmits the request with original sequence number. The value of retransmission timer T_{ret} is set to the maximum of either $(2 \times T_{ret})$ or E_d. This process ensures the maximum timer value for retransmissions and reduces the initial retransmission bursts, which will trigger retransmissions at the expiry of 0.5 seconds and 1.5 seconds, during the absence of SRSIP.

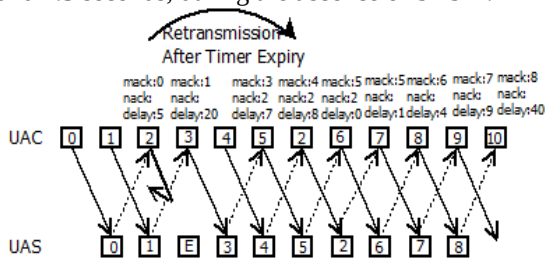


Figure 1: Modified Selective Repeat Protocol for SIP Signaling, Handling SIP request drops to ensure Reliability

C. Reduction in Retransmissions

Processing time of downstream SIP servers will be more during the overloaded scenarios like demand burst and server slow down. These processing delays are alternatively represented as queuing delays. With the increase the processing delay, downstream SIP servers cannot process the requests within the SIP window timer. So, the Client server assumes that the requests might have dropped in the network and retransmits the requests, resulting in unnecessary duplicate request re-transmissions. Even though the response delays can be identified from M_{rtt} values [17], the initial M_{rtt} values during demand burst conditions cannot provide accurate processing delays in the server. During this kind of scenarios, the estimated processing times calculated by the downstream servers provide a better input for calculating the retransmission timer window than that of the calculated values using M_{rtt}. During scenarios like server slowdown, downstream server may not calculate the exact delay values. During such scenarios, M_{rtt} will provide the more appropriate values. So, for proposed RSIP, we considered the maximum of M_{rtt} and E_d along with the default SIP retransmission timer, to reduce the retransmissions to the maximum extent. With this mechanism, even in worst case scenarios like server slow down, we can expect the SIP responses with a maximum of one SIP retransmission.

1) Handling Request Drops

SIP servers follow the retransmission mechanism, described in Algorithm 1. During SIP request drops, sequence numbers of the retransmitted SIP request should not be modified and must be same as they were present in the initial SIP request. Figure 1 shows the process for handling SIP request drops. In figure 1, SIP request with sequence number 2, 'seq = 2' is dropped in the network; UAC continuous to retransmit the SIP request with same sequence number after timer expiry.

UAS then receives the retransmitted SIP request with sequence number 2, 'seq = 2' and acknowledges the same in the next response.

2) Handling Response Drops

SIP doesn't retransmit response messages, by default, except for 200 OK responses of SIP Invite request [1]. So, SIP responses do not convey sequence numbers to ensure response reliability. Enabling SIP response retransmissions similar to that of SIP requests requires the modification in core functionality of SIP.

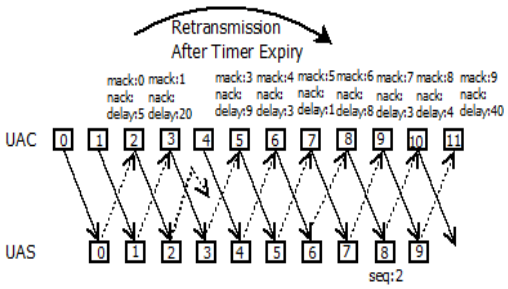


Figure 2: Modified Selective Repeat Protocol for SIP Signaling, Handling SIP response drops to ensure Reliability

This is a major modification in SIP protocol and may even result in issues towards backward compatibility, with the existing SIP servers that were not enabled with SRSIP. So reliability for SIP response messages is ensured similar to that of the existing SIP system, but using an optimized window timer, which is based on the M_{rtt} and E_d without violating SIP core functionality.

For SIP response drops, UAC retransmits SIP requests after the timer expiry. If the retransmitted SIP request holds the same old sequence number, similar to that when handling request drops, it may lead to ambiguity in downstream SIP server for processing and sending the acknowledgement. Besides this, the updated new sequence numbers in SIP requests ensures the reliability of the retransmitted requests. So the SIP requests are retransmitted with new updated sequence numbers instead of the old sequence numbers.

The process for handling response drops is depicted in figure 2. In figure 2, response for the request with 'seq = 2' is dropped in the network. So the request is retransmitted with the updated sequence number. Now the downstream server processes the retransmitted request similar to that of the original request and transmits the corresponding response to UAC.

D. Reduction in Retransmissions during Packet Drops

SIP sessions are managed at the servers by processing SIP messages. For simplicity, we assume that the SIP messages are transmitted in discrete time slots, instead of continuous message flow. And the length of each time slot is assumed to be 500 ms, which is equal to the minimum retransmission timer T_1.

Let us consider, the probability of dropping a request is 'P'. If 'N' requests are transmitted during each time slot, then the total number of requests dropped during each time slot, D(n), is

$$D(n) = NP$$

SIP client server identifies the request drops through 'nack' parameter in responses. Since we enabled reliability, SIP client server need to retransmit only dropped requests. So total number of retransmissions during a time slot will be equal to the number requests dropped in the previous time slot.

$$r(n) = D(n-1) = NP$$

Lemma 1: The amount of SIP requests dropped during a time, 't' is

$$N_t = \sum_{n=0}^{t}\sum_{j=2}^{6} P^j(r_j(n))\delta t$$

Proof: SIP request retransmissions are triggered at the rate of $(2^j-1)T_1$ [1]. SIP allows at most 6 request retransmissions. So, total number of retransmissions during n^{th} time slot will become

$$r(n) = \sum_{j=1}^{6} r_j(n)$$

Where r_j denotes j^{th} time retransmission of the initial SIP request processed downstream at the time $(n - T_j)$,

$$T_j = (2^j - 1)T_1, \qquad \forall\ 1 \le j \le 6$$

The probability for a SIP request drop during j^{th} time retransmission will be P_j. So the amount of request retransmissions during n^{th} time slot, including those dropped in retransmissions is

$$r(n) = \sum_{j=1}^{6} (P^j . r_j(n)) \tag{2}$$

The value of 'j' starts with 2 because, the initial retransmissions triggered by the SIP system after T_1 time interval will be completely avoided by the SRSIP system, as explained in Algorithm 1. Probability distribution function for SIP request drops is represented by equation 2. Equation 2 confirms that, the amount of SIP request retransmissions minimizes to zero or to a negligible value, with the increase in 'j' value.

The total number of requests dropped during time 't', N_t, is

$$N_t = \int_0^t r(n)dn = \int_0^t \sum_{j=1}^{6} (P^j . r_j(n))\, dn \tag{3}$$

Equation 3 can be reduced using Riemann Sum [45] as

$$N_t = \sum_{n=0}^{t}\sum_{j=2}^{6} P^j(r_j(n))\delta t \tag{4}$$

Where δt represents the Riemann division unit, smallest considered unit of time to sum up the areas formed by Riemann integral. The value of δt = 0.5 seconds, which is equal to the minimum retransmission timer T_1 [1] and the length of discrete time slot in which the requests are received.

E. Reduction in Retransmissions during Request Bursts

VoIP servers are regularly experiencing immense request bursts. These request bursts were regularly observed during special occasions, for example SMS contests, emergency calls [46],

New Year eve [11] etc. During SIP request bursts, servers require additional time for processing requests and therefore require additional time for server recovery, similar to Skype server's recovery [14]. Due to increase in the server recovery time, the upstream SIP server can not receive the corresponding SIP response before the expiry of retransmission window timer. As the retransmission timer expires in the upstream SIP server, before receiving the response, upstream server starts retransmitting the SIP requests, which may give rise to irrecoverable congestion.

Requests received by the downstream server are acknowledged using the proposed SRSIP signalling. The upstream SIP server then updates the retransmission timer window to a maximum possible value, which is calculated using Algorithm 1, such that the request retransmissions arise due to increased RTT can be eliminated. This updated window timer value reduces the number of unwanted request retransmissions that are forwarded towards the congested downstream SIP server.

Corollary 1: Total number of retransmissions triggered by a SIP server during time 't', using SRSIP is

$$R_t = \sum_{n=0}^{t}\left(\sum_{j=2}^{6} P^j . r_j(\mathrm{n})\right)\delta t + \sum_{n=0}^{t}\left(\sum_{j=2}^{6} r_j(\mathrm{n})\right)\delta t$$

Proof: Let us consider the mean arrival rate of SIP requests at the downstream SIP server be λ(n) during 'n^{th}' time slot, and the total number of SIP requests processed during 'n^{th}' time slot be μ(n). Since SIP allows at most 6 retransmissions [1], the total number of requests retransmitted during the current time slot 'n', r(n), are

$$r(n) = \sum_{j=2}^{6}(r_j(\mathrm{n})), \qquad \forall\, T_j < 32 \tag{5}$$

Where r_j denotes j^{th} time retransmission for the original request arriving at time $(n - T_j)$, where

$$T_j = max\{(2^j - 1)T_1, M_{rtt}(\mathrm{n}), E_d\}, \qquad \forall\, 2 \le \mathrm{j} \le 6$$

At time $(n-T_j)$ let the original request arrivals be $\lambda(n-T_j)$, existing requests in the server queue be $q(n-T_j)$. Server can process $\sum_{k=1}^{T_j}\mu(n - T_j + \mathrm{k})$ requests during time T_j. Then j^{th} time retransmitted requests during current time slot 'n', $r_j(n)$ is

$$r_j(\mathrm{n}) = \left[q(n - T_j) + \lambda(n - T_j) - \sum_{k=1}^{T_j}\mu(n - T_j + \mathrm{k})\right] \tag{6}$$

Equations 5, 6 confirm that the first time request retransmissions, which leads to an immense request retransmission burst, will be completely avoided. Apart from that the proposed SRSIP system also avoids the consequent request retransmissions, based on the RTT of the existing requests and estimated processing delay at the downstream server.

The total number of retransmitted requests during time 't', R_t, is

$$R_t = \int_0^t r(n)dn = = \int_0^t \left(\sum_{j=2}^{6} r_j(\mathrm{n})\right)dn \tag{7}$$

Equation 7 can be reduced using Riemann Sum [45] as

$$R_t = \sum_{n=0}^{t} r(n)\delta t = \sum_{n=0}^{t}\left(\sum_{j=2}^{6} r_j(\mathrm{n})\right)\delta \mathrm{t} \tag{8}$$

Equation 8 represents the total number of retransmitted requests during time 't' that are expected at the downstream SIP server, in a SIP system without any dropped or corrupted requests. In a SIP system that has high probability of request drops, the total number of retransmitted requests during time 't' will be the sum of retransmitted requests because of the requests dropped, equation 4, and that of retransmissions due to demand burst represented by equation 8.

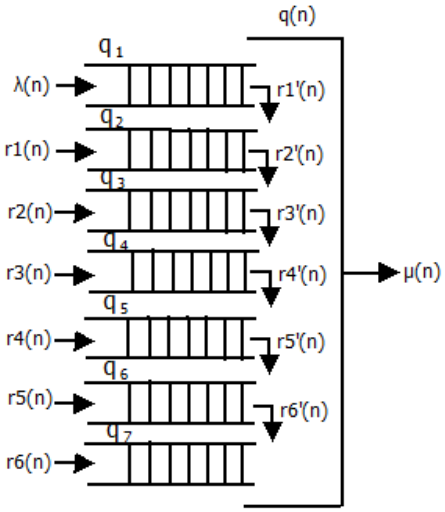


Figure 3: Advanced smart Priority Queuing Strategy

$$R_t = \sum_{n=0}^{t}\left(\sum_{j=2}^{6} P^j . r_j(\mathrm{n})\right)\delta t + \sum_{n=0}^{t}\left(\sum_{j=2}^{6} r_j(\mathrm{n})\right)\delta t \tag{9}$$

V. ADVANCED SMART PRIORITY QUEUING STRATEGY

Advanced Smart Priority (ASP) Queuing Strategy is a SIP Self-Overload control method. In ASP Queuing Strategy, SIP requests starving in queues for higher durations are given highest priority. The request starving in the queue for more time receives more number of retransmissions. So, SIP server prioritizes the requests for processing, based on the number of times the request has been retransmitted. This kind of request prioritizing reduces the request processing delay.

A SIP request can be retransmitted for a maximum of 6 times, at the intervals of $(2^j - 1)T_1$ seconds [1]. In ASP Queuing Strategy, the server maintains seven queues with increasing priorities. i.e., queue 2 will be processed with higher priority than queue 1; queue 3 will be processed with higher priority than queue 2, and so on up to queue 7. When the server receives the initial request, it will be pushed into queue1.

When the server receives the first retransmission for a request, then it will be pushed into queue 2 and the duplicated older request in queue 1 will be dropped. If the same request receives a retransmission before processing the request in queue 2, then it will be pushed into queue 3 and the duplicated request in queue 2 will be deleted. The process of deletion of duplicated requests in the lower priority queues reduces the processing time of the server and also reduces the unnecessary load on the downstream servers. Also, this process of inserting the retransmitted requests in higher priority queues and deleting the duplicates in lower priority queue will always maintain only one copy of the same request at the server at any time.

The Schematic process of the ASP Queuing Strategy is depicted in the figure 3. For simplicity, we assumed that the requests are received in time slots instead of continuous flow. $\lambda(n)$ represents the number of requests received during n^{th} time slot. $r_j(n)$ represents the j^{th} time retransmitted requests received during nth time slot. $r_{j`}(n)$ represents the j^{th} time retransmitted requests dropped during n^{th} time slot. $q(n)$ represents the total number of requests in all the queues. $\mu(n)$ represents the requests processed during n^{th} time slot.

A. Reduction in Queuing Delays

With ASP Queuing Strategy, all the SIP requests are queued in different queues, based on number of times the request has been retransmitted. Dropping of duplicated requests in the less priority queues reduces the queuing delay.

Corollary 2: The Average Queuing Delay in a SIP server using ASP Queuing Strategy is

$$T_n = \frac{1}{\mu(n) - \lambda(n)}$$

Proof: Let us consider the number of requests received at a particular discrete timeslot be λ and that of the processing capacity of server be μ, then the utilization of server ρ is

$$\rho = {}^{\lambda}/_{\mu} \tag{10}$$

In ASP Queuing Strategy, at time slot 'n', λ comprises of original requests and retransmitted requests $\lambda(n) + r(n)$ and that of μ comprises of requests processed towards downstream server and dropped requests i.e., $r`(n) + \mu(n)$.

ASP Queuing Strategy comprises of 7 queues. But we consider only one server for processing the requests. So the proposed ASP Queuing Strategy can be represented with M/G/1 queuing model [47]. According to Littles Law [48], expected number of packets in queuing system for M/G/1 queue is

$$E(n) = \lambda\tau$$

Where λ is the mean arrival rate and τ is the mean queuing delay experienced by a request. Mean queuing delay

$$\tau = \frac{E(n)}{\lambda} \tag{11}$$

But average number of packets in an M/G/1 queue [48] in equilibrium is

$$E(n) = \frac{\rho}{1 - \rho} \tag{12}$$

Now equation (11) reduces to

$$\tau = \frac{1}{\mu - \lambda} \tag{13}$$

Queuing delay with ASP Queuing Strategy at a discrete time slot 'n' is

$$\tau_n = \frac{1}{\left(r`(n) + \mu(n)\right) - \left(\lambda(n) + r(n)\right)} \tag{14}$$

In ideal conditions number of requests dropped due to starving in queues will be equal to the number of requests retransmitted, i.e., r(n) will be equal to r`(n), then the average queuing delay will be reduced to

$$T_n = \frac{1}{\mu(n) - \lambda(n)} \tag{15}$$

VI. HYBRID SIP SIGNALING

SRSIP system reduces the number of SIP request retransmissions to the minimum possible value. ASP Queuing Strategy reduces the queuing delay for request processing in the downstream server to the maximum possible value. So, we proposed a Hybrid SIP (HSIP) signaling model that integrates the features of SRSIP and ASP Queuing Strategy. HSIP incorporates the following congestion control techniques to reduce the retransmissions and queuing delay to the maximum possible value,

- Reliability: It enables Reliability for SIP System using SRSIP signaling.
- Self-Overload Control: It enables Self-Overload Control through ASP Queuing strategy.
- Overload Control: It enables overload control by conveying the Processing Delay of the downstream server.

A. Reduction in Retransmissions with HSIP Signaling

Enabling all the above congestion control techniques in the proposed HSIP System, ensure to increase the SIP system performance to the greatest possible value by reducing the retransmissions, queuing delay and increasing the processing speed.

Theorem 1: The total number of retransmissions triggered by a SIP server during time t using HSIP is

$$R_t = min\left\{\left\{\sum_{n=0}^{t}\sum_{j=2}^{6} P^j(r_j(n))\delta t + \sum_{n=0}^{t}\sum_{j=2}^{6} r_j(n)\,\delta t\right\}, \left\{\sum_{n=0}^{t}\sum_{j=1}^{6} P^j(r_j(n))\delta t + \sum_{n=0}^{t}\sum_{j=1}^{6} r_j(n)\,\delta t\right\}\right\}$$

Proof: Observations from the existing literature [18][19] [20][27][31] reveal that, a SIP system enabled with FIFO Queuing Strategy results in the least performance with the maximum retransmissions. So we consider the retransmissions in a SIP system with FIFO Queuing to be the largest value.

Let us consider a demand burst scenario similar to that of in corollary 2 for a SIP system enabled with FIFO Queuing Strategy.

In FIFO model, without SRSIP, the initial retransmissions will not be avoided. So, equations 4, 8, derived in lemma 1 and corollary 1 respectively, will become equations 16 and 17 respectively.

$$N_t = \sum_{n=0}^{t} \sum_{j=1}^{6} P^j \, r_j(\mathrm{n}) \, \delta \mathrm{t} \tag{16}$$

$$R_t = \sum_{n=0}^{t} \sum_{j=1}^{6} r_j(\mathrm{n}) \, \delta \mathrm{t} \tag{17}$$

So, the total number of retransmissions with FIFO Queuing Model with request drops

$$R_t = \sum_{n=0}^{t} \sum_{j=1}^{6} P^j \, r_j(\mathrm{n}) \, \delta \mathrm{t} + \sum_{n=0}^{t} \sum_{j=1}^{6} r_j(\mathrm{n}) \, \delta \mathrm{t} \tag{18}$$

From Corollary 1 the total number of retransmissions using SRSIP, which are considered to be the least possible value is represented in equation 9.

So, the total number of retransmissions triggered by a SIP server during time 't' using HSIP is

$$R_t = min \left\{ \begin{Bmatrix} \sum_{n=0}^{t} \sum_{j=2}^{6} P^j (r_j(\mathrm{n})) \delta t + \sum_{n=0}^{t} \sum_{j=2}^{6} r_j(\mathrm{n}) \, \delta t \end{Bmatrix}, \begin{Bmatrix} \sum_{n=0}^{t} \sum_{j=1}^{6} P^j (r_j(\mathrm{n})) \delta t + \sum_{n=0}^{t} \sum_{j=1}^{6} r_j(\mathrm{n}) \, \delta t \end{Bmatrix} \right\} \tag{19}$$

B. Reduction in Queuing Delay with HSIP Signaling

A SIP server enabled with ASP Queuing Strategy ensures the minimum Queuing delay with the highest request processing rate.

Theorem 2: The average queuing delay in a SIP server using HSIP is

$$T_n = min \left\{ \frac{1}{\mu(\mathrm{n}) - \lambda(\mathrm{n})}, \frac{1}{\mu(\mathrm{n}) - (\lambda(\mathrm{n}) + \mathrm{r}(\mathrm{n}))} \right\}$$

Proof: Queuing delay is identified to be maximum when employing FIFO Queuing Strategy [31]. Queuing delays using M/M/1 Queues is represented with equation 13. By substituting the request receiving rate, retransmission rate and request processing rate in equation 13, the queuing delay with FIFO queuing strategy at a discrete time slot 'n' is

$$T_n = \frac{1}{\mu(\mathrm{n}) - (\lambda(\mathrm{n}) + \mathrm{r}(\mathrm{n}))} \tag{20}$$

From corollary2, the least possible queuing delay for a SIP server can be attained with ASP Queuing Strategy. The queuing delay using ASP Queuing Strategy is represented in equation 15. Therefore from equations 15 and 20, the Average queuing delay in a SIP server using HSIP is

$$T_n = min \left\{ \frac{1}{\mu(\mathrm{n}) - \lambda(\mathrm{n})}, \frac{1}{\mu(\mathrm{n}) - (\lambda(\mathrm{n}) + \mathrm{r}(\mathrm{n}))} \right\} \tag{21}$$

VII. RESULTS AND DISCUSSION

The proposed HSIP system simultaneously reduces the SIP retransmissions from upstream servers and queuing delay for processing the SIP requests in downstream servers. Using simulation method, we evaluated and compared the reduction in SIP retransmissions and reduction in queuing delays using different congestion control techniques namely FIFO Queuing Strategy, ASP Queuing Strategy, RSIP Signaling, SRSIP Signaling and HSIP signaling.

The simulation system is enabled with MJSIP stack [49] and the SIP requests are generated using sipp [50] simulators. We modified the MJSIP stack, that can enable different queuing and signaling methods in SIP statefull proxy servers, based on configuration, to develop the servers under test. The results obtained during demand burst scenario and during fluctuating end-to-end Round Trip Times are discussed respectively in the following sub-sections. The simulations were carried out for 500 times in each scenario and the results were calculated from their respective averages.

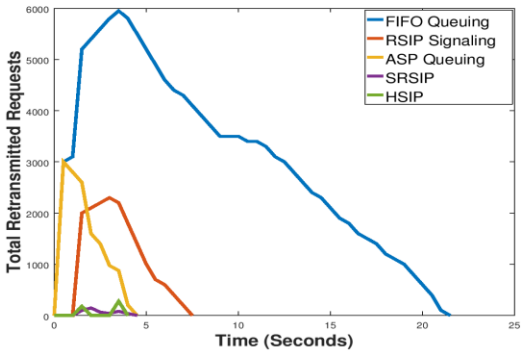


Figure 4: Retransmissions during Demand Burst Scenario

A. Demand Burst

SIP servers may receive requests in bursts during special events like new years, SMS contests, Indian Idol [46] etc similar to that happened for Skype [9] and WhatsApp [12].The SIP request load was generated using sipp simulators. Simulation environment is setup such that, the sipp simulators generated an initial load of 3500 requests. And one sipp simulator will continuously generate 100 requests for every 500ms. The sipp simulator is configured to generate load using Poisson distribution. Reduction in retransmissions during demand burst scenario is detailed below.

1) Reduction in Retransmissions

Reduction in retransmissions during demand burst scenario is depicted in Fig. 4. We analyzed the changes in number of SIP request retransmissions with time at the upstream SIP server. Number of SIP request retransmissions and the time for the SIP system to reach the normal state, is observed to be maximum when using FIFO Queuing Strategy. The retransmissions reached to a maximum of 6000 at 3.5 seconds due to the early / initial retransmissions. The SIP system took 23.5 seconds to get stabilized. Using ASP Queuing Strategy, the retransmissions reached to a maximum of 3000 but the system got stabilized within 4.5 seconds. Using RSIP signaling, the maximum retransmissions were observed to be 2300 but the system took 7.5 seconds to get stabilized.

Even though the number of retransmissions were reduced using RSIP, the time for stabilizing the system is greater than that of ASP Queuing Strategy. Using SRSIP system, the maximum number of retransmissions touched 145 and the system got stabilized within 4.5 seconds.

Using HSIP system only two peaks of request retransmissions were observed. The maximum retransmission are 240, which is greater than that of SRSIP but the overall retransmissions are 10% lesser than that of SRSIP signaling. Using HSIP, the system got stabilized within 4 seconds, which is the lowest of all the congestion control methods.

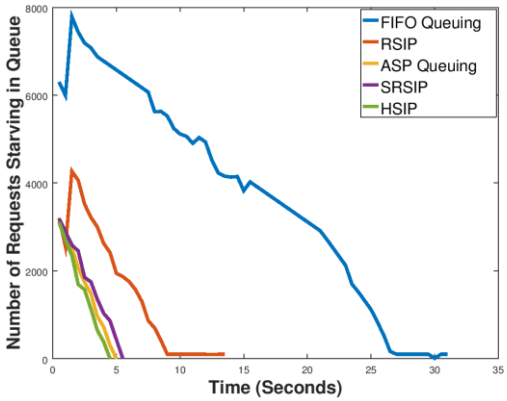


Figure 5: Number of SIP requests waiting in queue for processing during Demand Burst Scenario

2) Reduction in Queuing Delay

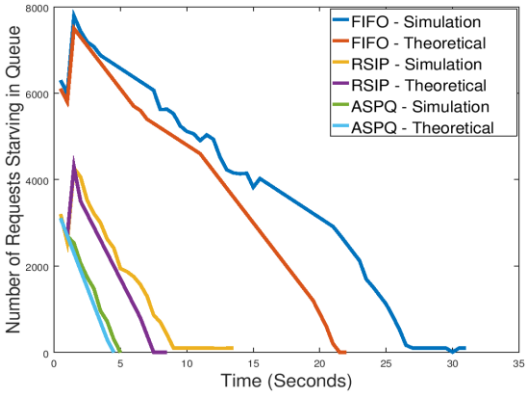
Queuing Delay is the amount of time the SIP requests are starving in the queue before getting processed in the downstream server. We analyzed the queuing delays at the downstream server and the time required for the server under test to get back to the stable state, using different methods, are represented in Fig. 5. The downstream SIP server took 27 seconds to get back to normal state using FIFO queuing Strategy where as it took 8 seconds, 5.5 seconds, 5 seconds and 4.5 seconds respectively when using RSIP signaling, ASP Queuing Strategy, SRSIP signaling and HSIP signaling. The queue lengths in simulation and theoretical methods reaches to zero with negligible time difference. The variations of queue lengths and maximum number of requests waiting in queues in both simulation and theoretical methods are observed at same timestamp, confirming that the simulation results are in agreement with the theoretical results.

3) Validation of Reduction in Queuing Delay

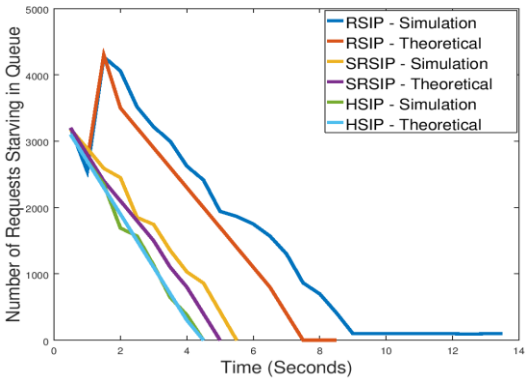
The equations derived for respective methods are populated with the numerical values, to obtain the theoretical results. The results obtained from the simulation methods are validated by comparing them with that of the theoretical results as shown in Fig. 6a and Fig. 6b. The variations of queue lengths and maximum number of requests waiting in queues in both simulation and theoretical methods are observed at same timestamp, confirming that the simulation results are in agreement with the theoretical results. The queue lengths in simulation and theoretical methods reach to zero at approximately same time.

VIII. CONCLUSION

SRSIP Signaling enables SIP with reliability along with overload control information. We proposed an algorithm with modified selective repeat protocol, to enable SIP system with smart reliability. SRSIP provides the estimated delay at downstream server to calculate the appropriate retransmission timer value and to reduce the number of retransmissions. It uses existing SIP Headers for acknowledging reliability and updating overload information, instead of mandating new Headers.



(a)FIFO, RSIP, ASPQ



(b)SRSIP, HSIP

Figure 6: Validation of Queuing Delay during Demand Burst

HSIP comprises the advantages of SRSIP and ASP Queuing strategy. SRSIP reduces the number of retransmissions. ASP Queuing Strategy reduces the Queuing delay at the downstream server. These two methods combine together to result in extreme reduction in retransmissions between servers and queuing delays at the downstream server. We analyzed the reduction in retransmissions using SRSIP signaling, reduction in queuing delays using ASP Queuing Strategy and reduction in queuing delays and retransmissions using HSIP signaling methods, with theoretical methods and validated the same using simulation method.

SIP system enabled with HSIP signaling resulted in 96% lesser retransmissions and 83% lesser queuing delays during demand burst scenarios.

REFERENCES

1. J. Rosenberg, H. Schulzrinne, G. Camarillo, A. Johnson, J. Peterson, R. Sparks, M. Handley, E. Schooler, SIP: Session

Initiation Protocol, IETF RFC 3261.
2. D. Malas, A. Morton, Basic Telephony SIP End-to-End Performance Metrics, IETF RFC 6076.
3. IP Multimedia Call control Protocol based on Session Initiation Protocol (SIP) and Session Description Protocol (SDP), 3rd Generation Partnership Project (3GPP)[3GPP TS 24.229, Version 5.25.0, Stage 3, Release 5, 09-2011].
4. A. Brown, WhatsApp - Highly-anticipated new feature FINALLY appears in popular chat app, [Express, UK, 27 October 2016] (2016).
5. J. Postel, User Datagram Protocol, IETF RFC 768.
6. J. Postel, Transmission Control Protocol, IETF RFC 793.
7. R. Stewart, K. Morneault, H. Schwarzbauer, T. Taylor, I. Rytina, K. Kalla, L. Zhang, V. Paxson, Stream Control Transmission Protocol, IETF RFC 2960.
8. M. Handley, H. Schulzrinne, E. Schooler, J. Rosenberg, SIP: Session Initiation Protocol, IETF RFC 2543.
9. J. Titcomb, Skype outage sees internet calls go down in many countries, [The Telegraph, 21 September 2015] (2015).
10. R. Ando, Skype hit by outage, says it is investigating, [Reuters news, 22 December 2010] (2010).
11. A. Brown, WhatsApp NOT Working: Facebook-owned messenger goes DOWN in New Years Eve OUTAGE, [Express, UK, 31 December 2015](2015).
12. W. Victoria, WhatsApp apologises as service crashes on New Years Eve: Users worldwide unable to connect as messaging app goes offline, [mailOnline, daily mail, UK, 31 December 2015] (2015).
13. A. Brown, WhatsApp Down: Users Complain Popular chat app is not working in UK, parts of Europe, [Express, UK, 23 May 2016] (2016).
14. R. Ando, UPDATE 2-Outage takes out Skype, recovery taking hours, [Reuters news, 22 December 2010] (2010).
15. Y. Hong, C. Huang, J. Yan, Impact of Retransmission Mechanism on SIP Overload: Stability Condition and Overload Control, Journal of Networks, No.1 7 (2012) 52–62.
16. D. Y. Yavas, I. Hokelek, B. Gunsel, An Analysis Tool to Evaluate Effect of Retransmissions on SIP Server Overloading, Seventh International Conference on Next Generation Mobile Apps, Services and Technologies.
17. K. K. Guduru, U. Jayadevappa, Reliable Session Initiation Protocol (RSIP) Signaling With UDP Transport Using Piggybacking Method, Telecommunication Systems Journal, Springer 68 (2018) 479–491.
18. V. Hilt, E. Noel, C. Shen, A. Abdelal, Design Consideration for SIP Overload Control, IETF RFC 6357.
19. P. O. Abaev, Y. V. Gaidamaka, A. V. Pechinkin, R. V. Razumchik, Simulation of Overload Control in SIP Server Network, in: in Proceeding of World Academy of Science, Engineering and Technology, 2012.
20. M. Ohta, Overload Control in SIP Signaling Network, in: International Journal of Electrical and Electronics Engineering, 2009.
21. R. G. Garroppo, S. Giordano, S. Spagna, S. Niccolini, Queuing Strategies for local overload control in SIP server, in: Proceedings of GLOBECOM communications, IEEE, 2009. doi:978-1-4244-4148-8.
22. H. Zheng, S. Wang, Reliable Session Initiation Protocol, VoIP technologies, Dr Shigeru Kashihara (Ed.), InTechdoi:10.5772/14414.
23. Y. Hong, C. Huang, Analysis of SIP retransmission probability using a Markov Modulated Poisson Process Model, in: Proceedings of Network Operations and Management Symposium, IEEE, 2010, pp. 179–186. doi:978-1-4244-5367-2.
24. Y. Hong, C. Huang, J. Yan, Controlling Retransmission Rate for Mitigating SIP Overload, in: Proceedings of International Conference on Communications (ICC), IEEE, 2011, pp. 1–5. doi:978-1-61284-231-8.
25. Khazaei.M, M. N, A dynamic distributed overload control mechanism in SIP networks with holonic multi-agent systems, Telecommunication Systems Journal 63 (2016) 437–455.
26. M. Jahanbakhsh, S. V. Azhari, H. Nemati, Lyapunov stability of SIP systems and its application to overload control, Computer Communication Journal 103 (2017) 1–17.
27. C. Eggr, M. Happenhofer, P. Reichl, SIP Proxy High-Load Detection by Continuous Analysis of Response Delay Values, in: Proceedings of 19th International Conference on Software, Telecommunications and Computer Networks, IEEE, 2011.
28. D. Sisalem, Sip Overload Control: Where are we Today, Trust worthy Internet, Springer Milan (2011) 273–287doi:978-88-470-1818-1.
29. D. Y. Yavas, I. Hokelek, B. Gunsel, On fluid flow modeling of priority based request scheduling for finite buffer SIP server, International Journal of Communication Systems 30 (2017) 1–18.
30. D. Y. Yavas, I. Hokelek, B. Gunsel, On modeling of priority-based SIP request scheduling, Simulation Modelling Practice and Theory 80 (2018) 128–144.
31. K. K. Guduru, U. Jayadevappa, Queuing Strategies for Self Overload Control in SIP Servers, in: International Conference on Contemporary Computing and Informatics IC3I, IEEE, 2014, pp. 1007–1011. doi:978-1-4799-6629-5/14.
32. M. Lulling, J. Vaughan, Reliability and Congestion Control for VoIP Signaling Transport, in: 5th WSEAS International Conference on Applied Computer Science, 2006, pp. 171–179.
33. T. Bova, T. Krivoruchka, Reliable UDP Protocol, IETF Draft (1999) 1–16.
34. Sliding window Protocol, https://en.wikipedia.org/wiki/Sliding_window_protocol, [Wikipedia] (2010).
35. M. Yajnik, J. Kurose, D. Towlsey, Packet Loss Correlation in the MBone Multicast Network, in: IEEE Globecom, 1996, pp. 94–99.
36. S. K. Chin, R. Braun, A Survey of UDP Packet Loss Charasteristics,in: Thirty-Fifth Asilomar Conference on Signals, Systems and Computers,2001, pp. 200–204.
37. Whats normal for latency and packet loss?, http://www.pingman.com/kb/42?nessoft_url=1, [Pingman tools, Article Number: 42] (2014).
38. M. Sanders, What are the chances of losing a UDP Packet, [Stack overflow] (2013).
39. Internet Traffic Report, http://www.internettrafficreport.com/30day.htm, [Global Packet Loss for 30 days] (2018).
40. V. Gurbani, V. Hilt, H. Schulzrinne, Session Initiation Protocol (SIP) Overload Control, IETF RFC 7339.
41. M. Karsai, N. Perra, A. Vespignani, Time-varying networks and the weakness of strong ties, Sci. Rep. 4doi:10.1038/srep04001.
42. K. K. Guduru, U. Jayadevappa, Overload Detection and Controlling Techniques in SIP Servers A survey, IRACST International Journal of Computer Networks and Wireless Communications, No 3 4.
43. K. K. Guduru, S. Dev, R. H. Naganur, Mitigating Power Consumption in Mobile Devices with Dynamic Triggering of XMPP Ping Requests, in: 84th Vehicular Technology Conference (VTC2016-Fall), IEEE, 2016.
44. E. Triantaphyllou, S. H. Mann, Usage of Analytical Hierarchy Process for Decision Making In Engineering Applications: Some Challenges, International Journal of Industrial Engineering: Applications and Practice 2 (1995) 35–44.
45. Thomas, G. B. J. Finney, R. L, Calculus and Analytic Geometry http://dx.doi.org/0-201-53174-7 doi:0-201-53174-7.
46. M. Homayouni, H. Nemati, V. Azhari, A. Akbari, Controlling Overload in SIP Proxies: An Adaptive Window based Approach Using no Explicit Feedback, in: IEEE Globecom, 2010.
47. A. Halilovic, M/G/1 Queuing System, http://ingforum.haninge.kth.se/armin/ALLA_KURSER/KOTEORI/EXER/repet12.pdf, [Queuing Theory Exercises.E12].
48. T. G. Robertazzi, Planning Telecommunication Networks http://dx.doi.org/81-219-2001-9 doi:81-219-2001-9.
49. A complete Java based Implementation of SIP Stack, http://www.mjsip.org/, [MJSIP Stack].
50. Open Source test tool / traffic generator for SIP Protocol, http://sipp.sourceforge.net/, [sipp] (2014).

AUTHORS PROFILE

Kiran Kumar Guduru, is working as a Chief Engineer in Samsung R&D Institute India – Bangalore He has over Nine years of experience in Mobile Protocols and Platforms. He completed his MCA from Sri Venkateswara University. He is a research scholar in R.V. College of Engineering Research Center, affiliated to Visvesvaraya Technological University.
His interests include research and development to reduce network congestion in Telecom Signaling.
He is a member of IETF RTCWEB working group, W3C WebRTC and Media Capture and Streams working groups.

Usha Jayadevappa, is working as a Director for Master of Computer Applications (MCA) Department in R. V. College of Engineering. She obtained her B.Sc(PCM) and MCA from Bangalore University. She has over Ninteen years of experience in research and teaching. She obtained her Ph.D from University of Pune in 2011.
She is a member of IEEE, CSI and ISTE.