Controlling Retransmission Rate For Mitigating SIP Overload

Yang Hong, Changcheng Huang, James Yan
Dept. of Systems and Computer Engineering, Carleton University, Ottawa, Canada
E-mail: {yanghong, huang}@sce.carleton.ca, jim.yan@sympatico.ca

***Abstract*—Recent server collapse in carrier networks (e.g., Skype outage) indicates that message retransmissions triggered by various SIP timers make the overload worse. The built-in overload control mechanism cannot handle overload conditions effectively. Since the retransmissions stimulated by the overload introduce more overhead rather than reliability, we suggest mitigating the overload by reducing the retransmission rate. We propose a novel algorithm to detect the potential overload at the downstream servers and control message retransmission rate from upstream servers to mitigate the overload at the downstream servers. We investigate two typical overload scenarios caused by demand burst and server slow down respectively. OPNET simulations demonstrate that (1) the proposed solution can help the overloaded downstream server to cancel the short-term overload quickly after it resumes its normal operation status; (2) without the overload control algorithm applied, the overload at the downstream server may propagate or migrate to its upstream servers.**

I. INTRODUCTION

Internet telephony is experiencing rapidly growing deployment due to its lower-cost telecommunications solutions for both consumer and business services. Session Initiation Protocol (SIP) [1] has become the main signaling protocol to establish the multimedia sessions for numerous Internet telephony applications such as Voice-over-IP, instant messaging and video conferencing. 3rd Generation Partnership Project (3GPP) has adopted SIP as the basis of its IP Multimedia Subsystem (IMS) architecture [2-4].

SIP introduces a retransmission mechanism to maintain its reliability [1, 5]. In practice, a SIP source uses timeouts to detect message losses. One or more retransmissions would be triggered if the corresponding reply message is not received in a predetermined time interval.

SIP RFC 3261 [1] suggests that the SIP retransmission mechanism should be disabled for hop-by-hop transaction when running SIP over Transmission Control Protocol (TCP) to avoid redundant retransmissions at both SIP and TCP layer [1]. However, nearly all vendors choose to run SIP over UDP (User Datagram Protocol) instead of TCP for the following reasons [4, 6-18]: (1) The overhead of state management such as three-way handshake prevents TCP from real-time application which is a critical requirement for SIP protocol; (2) The complex congestion control mechanism of TCP aims at preventing congestion caused by bandwidth exhaustion, thus it provides little help to SIP overload caused by CPU constraint; (3) TCP only provides reliability at transport layer, but SIP messages can still be dropped or corrupted while being processed at application layer.

As SIP network is reaching large scales and serving increasing number of users, periods of heavy load may happen regularly. When a SIP server is experiencing a heavy load, it may create an extremely long queuing delay and trigger unnecessary message retransmissions, thus consuming more memory and increasing CPU loads of the SIP server. This may cause the network to be severely overloaded and suffer from potential network collapse [4, 6-16].

RFC 5390 [17] identified the various reasons that may cause server overload in a SIP network. These include but not limited to poor capacity planning, dependency failures, component failures, avalanche restart, flash crowds, denial of service attacks, etc. In general, anything that may trigger a demand burst or a server slowdown can cause server overload and lead to server crash.

When a SIP receiving server detects the overload, it terminates some transactions by sending "503 service unavailable" messages to the sending servers. Such built-in overload control mechanism has two limitations: (1) The cost of rejecting a session is comparable with the cost of serving a session [10]; (2) A 503 response only terminates a transaction. The overloaded server may continue to receive subsequent requests for session establishment, thus exhausting its limited CPU resources to reply more 503 messages and adding to its overloaded state [6].

The contributions of this paper are: (1) Using difference equations to analyze the impact of the retransmission mechanism on the queuing dynamics of an overloaded downstream receiving server and its upstream sending server; (2) Using retransmission timer queue size to detect the overload and Proposing a novel heuristic algorithm to mitigate the overload by controlling retransmission rate; (3) Performing OPENT simulations under two typical overload scenarios to validate the efficiency of our overload control algorithm.

II. RELATED WORK

Recent collapses of SIP servers in carrier networks (e.g., Skype outage [19]) has motivated numerous overload control solutions (e.g., [6-16]). Three window-based feedback algorithms were proposed to adjust the message sending rate of upstream SIP servers based on the queue length [10]. Both centralized and distributed overload control mechanisms for SIP were investigated in [9]. Retry-after control, processor occupancy control, queue delay control and window based control were proposed to improve goodput and prevent overload collapse in [6]. However, these overload control proposals suggested that the overloaded receiving server advertises its upstream sending servers to reduce their sending rates.

Such pushback control solution would increase the queuing delays of newly arrival original messages at upstream servers,

which in turn cause overload at the upstream servers. Overload may thus propagate server-by-server to sources and block large amount of calls which means the revenue loss for carriers.

III. SIP Overview

To briefly describe the basic SIP operation, we only consider originating User Agent (UA), Proxy-server (P-server) and terminating UA. Fig. 1 depicts a typical procedure of a session establishment. To set up a call, an originating UA sends an "Invite" request to a terminating UA via two P-servers. The P-server returns a provisional "100(Trying)" response to confirm the receipt of the "Invite" request. The terminating UA returns a "180(Ringing)" response after confirming that the parameters are appropriate. It also evicts a "200(OK)" message to answer the call. The originating UA sends an "ACK" response to the terminating UA after receiving the "200(OK)" message. Finally the call session creates the media communication between the originating UA and the terminating UA. The "Bye" request is generated to close the session thus terminating the communication.

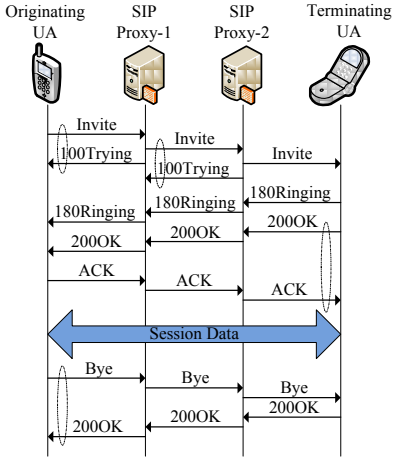


Fig. 1. A typical procedure of session establishment.

When an overload happens, the hop-by-hop Invite-100(Trying) transaction is the major workload contributor due to its role for call setup and its hop-by-hop retransmission mechanism [9]. Given the proportionate nature and the general similarity of the retransmission mechanisms between the "Invite" and "non-Invite" messages in a typical session [1], we will focus on the hop-by-hop Invite-100(Trying) transaction in this paper. For each hop, the sender starts the first retransmission of the original message at T_1 seconds, and the time interval doubles after every retransmission (exponential back-off), if the corresponding reply message is not received. The last retransmission is sent out at the maximum time interval $64 \times T_1$ seconds. Thus there is a maximum of 6 retransmissions. The default value of T_1 is 0.5s.

IV. SIP Overload Control Algorithm

When the message arrival rate to a server exceeds its message processing capacity, overload happens and its queue builds up. Then the long queuing delay would trigger unnecessary retransmissions from its upstream servers to make the overload worse. Therefore, it is necessary to investigate the impact of retransmissions on the queuing dynamics (e.g., [15, 20]), before we propose our novel algorithm to control retransmission rate.

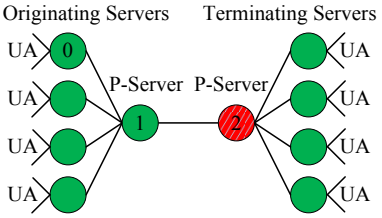


Fig. 2. SIP network topology with an overloaded downstream receiving Server 2 (which is marked with diagonal lines) and its upstream sending Server .1

A. Queuing dynamic of SIP retransmission mechanism

We consider a SIP network which consists of an overloaded downstream receiving Server 2 and its upstream sending Server 1 (as shown in Fig. 2). We use difference equations to describe the queuing dynamics of Server 1 and Server 2 respectively, by making the following assumptions according to SIP RFC [1]:

(a) The SIP RFC [1] does not specify the queuing discipline to be deployed by a SIP server. We assume that a SIP server maintains a First-In-First-Out (FIFO) queue for messages arriving at different time-slots. All request messages enter the tail of the message queue [9]. This FIFO queuing model reflects the common practice by most vendors today;

(b) Response messages enter the head of message queue, because delaying the processing of response messages may trigger unnecessary retransmissions to exacerbate the overload;

(c) To study the interactions between overloaded receiving Server 2 and its upstream sending Server 1, we assume that the upstream servers of Server 1 and the downstream servers of Server 2 have sufficient capacity to process all arrival messages without any delay, while Server 2 is the most overloaded server in the network;

(d) Practical buffer sizes vary with the actual service rates and system configuration plans. With the memory becoming cheaper and cheaper, typical buffer sizes are likely to become larger and larger. The buffer sizes for all servers are assumed to be large enough to hold all the incoming messages.

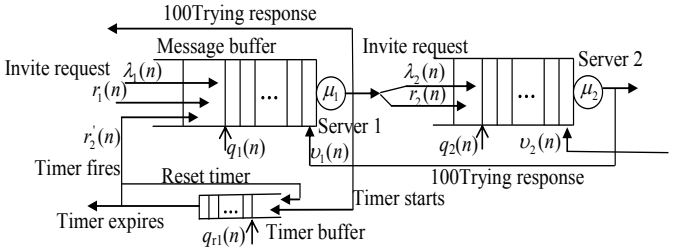


Fig. 3. Queuing dynamics of an overloaded Server and its upstream server.

Fig. 3 depicts the queuing dynamics of Server 1 and Server 2. There are two queues at each server: one to store the messages and the other to store the retransmission timers. The downstream servers of Server 2 have sufficient capacity to process all the arrival messages so that no retransmissions will be generated by Server 2 for its upstream servers. Thus Fig. 3 did not show the timer queue of Server 2.

We consider the overloaded downstream Server 2 first. We can obtain the queue size $q_2(n+1)$ for the message queue of

Server 2 at next time slot n+1 based on the information at the current time slot n, i.e.,

$$q_2(n+1)=[q_2(n)+\lambda_2(n)+r_2(n)+\upsilon_2(n)-\mu_2(n)]^+, \quad (1)$$

where at the current time slot n at Server 2, $q_2(n)$ denotes the queue size; $\lambda_2(n)$ denotes the arrival original request messages; $r_2(n)$ denotes the arrival retransmitted messages corresponding to λ_2(n); $\upsilon_2(n)$ denotes the response messages sent by downstream servers; $\mu_2(n)$ denotes the processed messages.

There are two queues at Server 1: one to store the messages and the other to store the timers. Like Eq. (1), we can obtain queuing dynamics for message queue of Server 1 as follows,

$$q_1(n+1)=[q_1(n)+\lambda_1(n)+r_1(n)+r'_2(n)+\upsilon_1(n)-\mu_1(n)]^+, \quad (2)$$

where at the current time slot n at Server 1, $q_1(n)$ denotes the queue size; $\lambda_1(n)$ denotes the aggregated arrival original request messages, which can be arbitrary stochastic process (e.g., Poisson, Pareto, Gamma or Lognormal distribution); $r_1(n)$ denotes the aggregated retransmitted messages corresponding to λ_1(n); $\upsilon_1(n)$ denotes the response messages corresponding to $\lambda_2(n)$; $\mu_1(n)$ denotes the processed messages; $r'_2(n)$ denotes the retransmissions generated by Server 1 corresponding to $\lambda_2(n)$.

Let us consider an overload scenario that Server 2 performs its routine maintenance and reduces its processing capacity. The original message rate is larger than the service rate, and the queue size q_2 increases slowly. After a short period of overload, the queuing delay of Server 2 is long enough to trigger the retransmissions r'_2 which enter the queue of Server 1. If the total new message arrival rate of λ_1, υ_1 and r'_2 is larger than the service rate μ_1, the queue size q_1 would increase (as described by Eq. (2)) and may trigger the retransmissions r_1. The retransmissions r_1 would enter Server 1 to increase the queue size q_1 more quickly, thus propagating the overload from Server 2 to Server 1. On the other hand, after queuing and processing delay at Server 1, the retransmitted messages r'_2 depart Server 1 and enter Server 2 as r_2 to increase the queue size q_2 more quickly (as described by Eq. (1)), thus making the overload at Server 2 much worse.

B. Overload Control Algorithm

As the retransmitted messages r'_2 may increase queue sizes at both Server 1 and Server 2 and bring the overload to both servers, our goal for mitigating the overload is to control the retransmission rate r'_2. To achieve our target, we calculate a retransmission probability p to determine whether to retransmit an original message when its retransmission timer fires or expires. The timers are ordered by their expiration periods in the retransmission timer queue.

Dynamics of retransmission timer queue q_{r1}

After Server 1 has forwarded the original messages λ_2 to the downstream Server 2, it inserts the corresponding retransmission timers into the retransmission timer queue q_{r1}. The response messages υ_1 remove the retransmission timers corresponding to λ_2 from the timer queue q_{r1}.

When the retransmission timer queue size q_{r1} is larger than its maximum threshold q_{r1max}, the overload at one or multiple downstream servers is anticipated. This indicates that Server 2 delays the processing of λ_2 and the generation of corresponding υ_1. Thus we use the retransmission timer queue size q_{r1} to detect the overload, and propose our novel overload control algorithm as depicted by Fig. 4.

Overload Control Algorithm

When each retransmission timer fires or expires
  Calculate retransmission probability p:
    if $q_{r1} < q_{r1min}$
      $p \leftarrow 1$
    else
      if $q_{r1min} \le q_{r1} \le q_{r1max}$
        $p \leftarrow 1-(1-p_{min})(q_{r1}-q_{r1min})/(q_{r1max}-q_{r1min})$
      else
        $p \leftarrow p_{min}$

Varying parameter:
  q_{r1}: Instantaneous timer queue size of Server 1

Dynamically tuning parameters:
  q_{r1min}: Minimum threshold for q_{r1}
  q_{r1max}: Maximum threshold for q_{r1}

Fixed parameter:
  p_{min} : Minimum retransmission probability

Fig. 4. Overload control algorithm for controlling retransmission rate

C. Setting of Overload Control Parameters

There are two dynamically tuning parameters and one fixed parameter in our overload control algorithm: minimum retransmission probability p_{min}, minimum threshold q_{r1min} for the retransmission timer queue size q_{r1}, and maximum threshold q_{r1max} for the retransmission timer queue size q_{r1}. We will discuss our parameter setting strategy as follows.

Setting p_{min}

SIP provides retransmission to maintain its reliability due to the message loss. The Internet Traffic Report indicates that current global packet loss statistic averaged 8% packet loss [21]. Since some messages may be corrupted while being processed at application layer, average ζ=10% message loss is assumed. Thus we suggest maintaining a minimum retransmission probability p_{min} as

$$p_{min}=10\%. \quad (3)$$

The non-retransmission probability of all 6 retransmissions for a lost message is $(1-p_{min})^6$, thus the minimum retransmission probability of a lost message becomes $p_\zeta=1-(1-p_{min})^6$ $=1-(1-10\%)^6\approx47\%$. This indicates that in order to mitigate the overload, maximum $(1-p_\zeta)$=53% average message loss cannot be recovered after 6 retransmissions when the overload is anticipated at the downstream servers. This will cause maximum $\zeta*(1-p_\zeta)$=10%*53%≈5% calls to be rejected.

Setting q_{r1min} and q_{r1max}

Since Server 1 only generates the retransmissions for the original messages whose retransmission timers fire or expire in the timer queue q_{r1}, and a queuing delay less than the 1st retransmission timer T_1 at Server 2 will not trigger any retransmissions, we suggest minimum threshold q_{r1min} as

$$q_{r1min}=\lambda'_{1avg}*T_1, \quad (4)$$

where λ'_{1avg} is the moving average original message departure rate of Server 1. In our scenario, the original message departure rate λ'_1 of Server 1 is equal to the original message arrival rate λ_2 of Server 2. We can use an exponential weighted moving average filter to obtain λ'_{1avg} as

$$\lambda'_{1avg}(n)=(1-w_\lambda)\lambda'_{1avg}(n-1)+w_\lambda\lambda'_1(n), \quad (5)$$

where w_λ is a filter weight. A good value for the averaging filter is recommended as 0.002 [22], i.e., w_λ=0.002. The initial value of λ'_1 is set as the mean service capacity C_1 of Server 1.

We define the ratio between maximum threshold q_{r1max} and minimum threshold q_{r1min} as α, i.e., $q_{r1max}=\alpha q_{r1min}$, and $\alpha \geq 1$. Based on our numerous experiments, we suggest setting α as 3, i.e., $q_{r1max}=3q_{r1min}=\lambda'_{1avg}*3T_1=\lambda'_{1avg}*(T_1+2T_1)$. This means that the retransmission probability p reaches its minimum value p_{min} when the queuing delay of the original messages in the timer queue is longer than 2nd retransmission timer.

Quite different from fixed packet queue thresholds for most active queue management algorithms (e.g., RED [22]) that provide end-to-end congestion control for data traffic, both minimum threshold q_{r1min} and maximum threshold q_{r1max} for retransmission timer queue are adaptively tuned by the moving average original message departure rate λ'_{1avg} to achieve hop-by-hop overload control for signaling traffic.

V. Performance Evaluation And Simulation

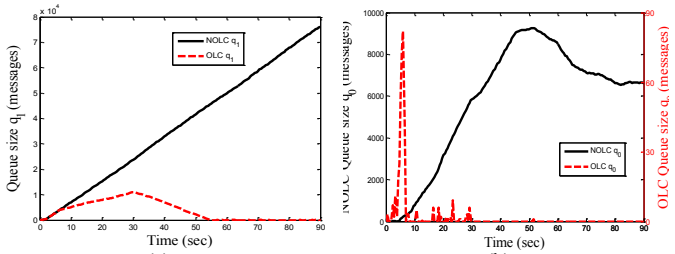
In order to validate our overload control algorithm, we perform OPNET simulation based on the SIP network topology depicted by Fig. 2. Four originating servers generated original request messages with equal rate, and then sent them to four terminating servers via two proxy servers (i.e., Server 1 and Server 2). Currently there is no measurement result for the workload in the real SIP networks. Since Poisson distribution has been widely adopted in existing research works (e.g., [10]), our message generation rates are Poisson distributed. The message service time of each server is exponentially distributed. Since processing a response message takes much less time than processing a request message, we denote β as the ratio of the mean processing time of a response message to that of a request message and set β as 0.5. We assume that the mean service capacity of a Proxy server is 1000 messages/sec measured based on the processing time of request message, i.e. $C_1=C_2=1000$ request messages/s. That is, the mean processing times for a request message and a response message are 1ms and 0.5ms respectively. The mean service capacity of an originating server or a termination server is equal to 500 request messages/sec. The total message service rate μ is bounded by the service capacity C at each server, i.e., $\mu \leq C$. The average message loss probability is 10%.

We consider two typical overload scenarios: (1) Overload at Server 1 due to a demand burst; (2) Overload at Server 2 due to a server slowdown. The simulation period is 90s. In each scenario, we conducted our simulations with overload control algorithm and without overload control algorithm separately. In all the simulation plots, we use "OLC"/"NOLC" to indicate that overload control algorithm "was"/"was not" applied to all servers in the SIP network.

A. Overload at Server 1

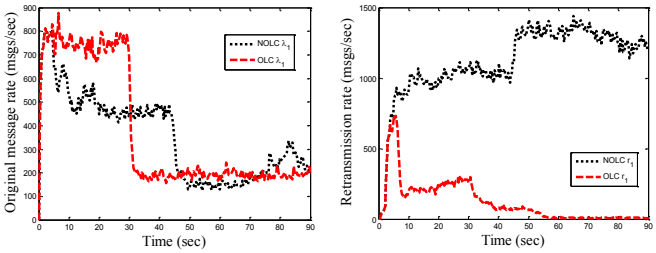
In this scenario, the mean message generation rate for each originating server was 200 messages/sec (i.e., $\lambda_1=4\lambda_0=800$ messages/sec, emulating a short surge of user demands) from time t=0s to t=30s, and 50 messages/sec (i.e., $\lambda_1=4\lambda_0=200$ messages/sec, emulating regular user demands) from time t=30s to t=90s. The mean service capacities of two proxy servers were $C_1=C_2=1000$ messages/sec.

Figs. 5 and 6 show the dynamic behaviour of overloaded Server 1 and one of its upstream originating servers. Without overload control algorithm applied, it is easy to see from Fig. 5(a) that Server 1 became CPU overloaded immediately and the overload deteriorated with time going, leading to the eventual crash of Server 1. Since the aggregate service capacity of four originating servers was larger than that of proxy Server 1, the queue size of each originating server decreased slowly (see Fig. 5(b)) after new original message generation rates decreased.



(a) (b)

Fig. 5. (a) Queue size q_1 (messages) of Server 1 versus time. (b) Queue size q_0 (messages) of an originating server versus time.



(a) (b)

Fig. 6. (a) Moving average original message rate λ_1 (messages/sec) of Server 1 versus time. (b) Moving average retransmission rate r_1 (messages/sec) for Server 1 versus time.

Our overload control algorithm made the queue size of Server 1 increase slowly during the period of the demand burst, and cancelled the overload at Server 1 within 25s after the new user demand rate reduced at time t=30s (see Fig. 5(a)).

B Overload at Server 2

In this scenario, the mean server capacities of the two proxy servers were C_1=1000 messages/sec from time t=0s to t=60s, C_2=100 messages/sec from time t=0s to t=30s, and C_2=1000 messages/sec from time t=30s to t=90s. The mean message generation rate for each original server was 50 messages/sec. When there were no queuing delays at all servers in the beginning, the SIP traffic enters Server 1 with λ_1=200 messages/sec, and then enters Server 2 with λ_2=200 messages/sec.

Without overload control algorithm applied, Figs. 7 and 8 demonstrate the following phenomena: (1) Server 2 became overloaded first, which was followed by a later overload at Server 1, i.e., the overload propagation which may cause the widespread server crashes. (2) After Server 2 resumed its normal service at time t=30s, Server 1 and Server 2 had the same service capacity. As Server 2 processed the residual messages in the queue and fed back response messages to Server 1 more quickly, the response rate of Server 1 increased sharply (see Fig. 8(a)). Server 1 had to process part of r_1 which would not enter Server 2, the total arrival rate at Server 2 was less than its service capacity. Therefore the queue at Server 2 started going down. Eventually the overload at Server 2 was

cancelled at time t≈64s, while the overload at Server 1 persisted due to its higher arrival rate of aggregated request messages (see Figs. 7 and 8(a)).

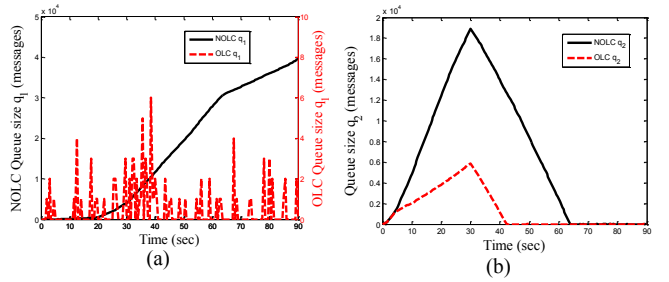


Fig. 7. (a) Queue size q_1 (messages) versus time. (b) Queue size q_2 (messages) versus time.

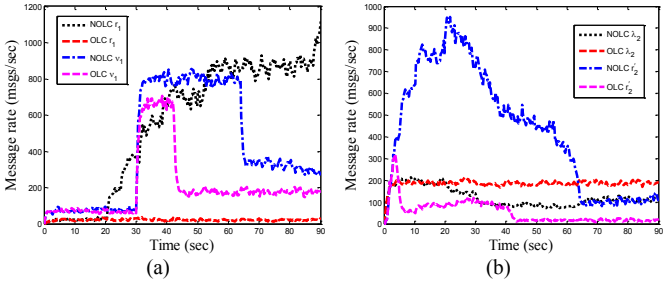


Fig. 8. (a) Moving average retransmission rate r_1 (messages/sec) for Server 1 and moving average response rate υ_1 (messages/sec) of Server 1 versus time. (b) Moving average original message rate λ_2 (messages/sec) of Server 2 and moving average retransmission rate r'_2 (messages/sec) for Server 2 versus time.

With our overload control algorithm applied, the retransmission rate r'_2 for Server 2 was restricted. The overload at Server 2 was mitigated and the queue size of Server 2 increased relatively slowly. In the mean time, Server 1 had enough capacity to process the limited retransmissions for Server 2, thus maintaining a small queue. Retransmission was triggered only for message loss recovery at Server 1 (i.e., r_1≈20messages/sec). After Server 2 resumed its normal service, it only spent 13s to cancel the overload and the buffer became empty at time t≈43s (see Fig. 7(b)).

VI. Conclusions

We have investigated the impact of SIP retransmission mechanism on queuing dynamics of an overloaded downstream receiving server and its upstream server. Our queuing analysis has demonstrated that unnecessary retransmissions not only exacerbate the overload, but also propagate the overload to server-by-server thus bringing down the whole SIP network (e.g., recent Skype outage [19]).

When the overload occurs at the downstream servers, our overload control algorithm reduces retransmission rate to mitigate the overload, while maintaining the original message rate to avoid excessive revenue loss.

Our OPNET simulation results have demonstrated that our overload control algorithm can cancel the short-term overload effectively regardless of whether the overload is caused by the demand burst or by server slow down.

One of main advantages of heuristic control approach is its simple structure of implementation. Control theoretic approach (e.g., [23, 24]) can reduce the impact of bandwidth congestion in TCP/IP layer on the retransmission timer queue, but the CPU cost of implementation is higher. When an overload lasts for a long period, our retransmission-based solution can be combined with pushback solution (e.g., [14]) to reject some calls by reducing the original message rates of SIP sources.

Acknowledgment

This work was supported by the NSERC grant #CRDPJ 354729-07 and the OCE grant #CA-ST-150764-8.

References

[1] J. Rosenberg, H. Schulzrinne, G. Camarillo, A. Johnston, J. Peterson, R. Sparks, M. Handley, and E. Schooler, "SIP: Session Initiation Protocol," *IETF RFC 3261*, June 2002.

[2] "3rd Generation Partnership Project". http://www.3gpp.org.

[3] S.M. Faccin, P. Lalwaney, and B. Patil, "IP Multimedia Services: Analysis of Mobile IP and SIP Interactions in 3G Networks," *IEEE Communications Magazine*, 42(1), January 2004, pp. 113-120.

[4] E. Noel and C.R. Johnson, "Initial simulation results that analyze SIP based VoIP networks under overload," in *Proceedings of 20th International Teletraffic Congress*, 2007, pp. 54-64.

[5] M. Govind, S. Sundaragopalan, K.S. Binu, and S. Saha, "Retransmission in SIP over UDP - Traffic Engineering Issues," in *Proceedings of International Conference on Communication and Broadband Networking*, Bangalore, India, May 2003.

[6] E. Noel and C.R. Johnson, "Novel Overload Controls for SIP Networks," *Proceedings of 21st International Teletraffic Congress*, 2009.

[7] R.P. Ejzak, C.K. Florkey, and R.W. Hemmeter, "Network Overload and Congestion: A comparison of ISUP and SIP," *Bell Labs Technical Journal*, 9(3), 2004, pp. 173–182.

[8] M. Ohta, "Overload Control in a SIP Signaling Network," in *Proceeding of World Academy of Science, Engineering and Technology*, Vienna, Austria, March 2006, pp. 205—210.

[9] V. Hilt and I. Widjaja, "Controlling Overload in Networks of SIP Servers," in *Proceedings of IEEE ICNP*, Orlando, Florida, October 2008, pp. 83-93.

[10] C. Shen, H. Schulzrinne, and E. Nahum, "SIP Server Overload Control: Design and Evaluation," in *Proceedings of IPTComm*, Heidelberg, Germany, July 2008.

[11] A. Abdelal and W. Matragi, "Signal-Based Overload Control for SIP Servers," in *Proceedings of IEEE CCNC*, Las Vegas, NV, January 2010.

[12] "SIP Express Router" http://www.iptel.org/ser/.

[13] T. Warabino, Y. Kishi and H. Yokota, "Session Control Cooperating Core and Overlay Networks for "Minimum Core" Architecture," in *Proceedings of IEEE Globecom*, Honolulu, Hawaii, December 2009.

[14] V. Hilt and H. Schulzrinne, "Session Initiation Protocol (SIP) Overload Control," *IETF Internet-Draft*, January 2011.

[15] Y. Hong, C. Huang, and J. Yan, "Analysis of SIP Retransmission Probability Using a Markov-Modulated Poisson Process Model," in *Proceedings of IEEE/IFIP Network Operations and Management Symposium*, Osaka, Japan, April 2010, pp. 179–186.

[16] E.M. Nahum, J. Tracey, and C.P. Wright, "Evaluating SIP server performance," in *Proceedings of ACM SIGMETRICS*, San Diego, CA, US, 2007, pp. 349–350.

[17] J. Rosenberg, "Requirements for Management of Overload in the Session Initiation Protocol," *IETF RFC 5390*, December 2008.

[18] Y. Hong, O. W. W. Yang, and C. Huang, "Self-Tuning PI TCP Flow Controller for AQM Routers With Interval Gain and Phase Margin Assignment," in *Proceedings of IEEE Globecom*, Dallas, TX, U.S.A, November 2004, pp. 1324-1328.

[19] R. Ando, "Internet phone and video service Skype went down in a global service outage," Reuters News, December 22nd, 2010.

[20] Y. Hong, C. Huang, and J. Yan, "Modeling and Simulation of SIP Tandem Server with Finite Buffer," *ACM Transactions on Modeling and Computer Simulation*, **21**(2), 2011.

[21] "Internet Traffic Report", http://www.internettrafficreport.com/, 2010.

[22] S. Floyd, V. Jacobson, "Random Early Detection Gateways for Congestion Avoidance," *IEEE/ACM Transactions on Networking*, **1**(4), 1993, pp. 397-413.

[23] Y. Hong, C. Huang, and J. Yan, "Mitigating SIP Overload Using a Control-Theoretic Approach," in *Proceedings of IEEE Globecom*, Miami, FL, U.S.A, December 2010.

[24] Y. Hong, C. Huang, and J. Yan, "Design Of A PI Rate Controller For Mitigating SIP Overload," in *Proceedings of IEEE ICC*, Kyoto, Japan, June 2011.