Design Of A PI Rate Controller 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 collapses of SIP servers in the carrier networks (e.g., Skype outage) indicate that the built-in SIP overload control mechanism cannot mitigate overload effectively. In this paper, by employing a control-theoretic approach that models the interaction between an overloaded downstream server and its upstream server as a feedback control system, we investigate the root cause of SIP server crash by studying the impact of the retransmission on the queuing delay of the overloaded server. Then we design a PI rate controller to mitigate the overload by regulating the retransmission rate based on the round trip delay. We derive the guidelines for choosing PI controller gains to ensure the system stability. Our OPNET simulation results demonstrate that our proposed control theoretic approach can cancel the short-term SIP overload effectively, thus preventing widespread SIP network failure.**

I. INTRODUCTION

Internet protocol (IP) telephony is experiencing rapidly growing deployment due to its lower-cost telecommunications solutions for both consumer and business services. SIP (Session Initiation Protocol) [1] has become the main signaling protocol to setup, manage and terminate sessions for IP 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]. With the 3rd Generation wireless technology being adopted by more and more carriers, most cellular phones and other mobile devices are starting to use or are in the process of supporting SIP for session establishment [3].

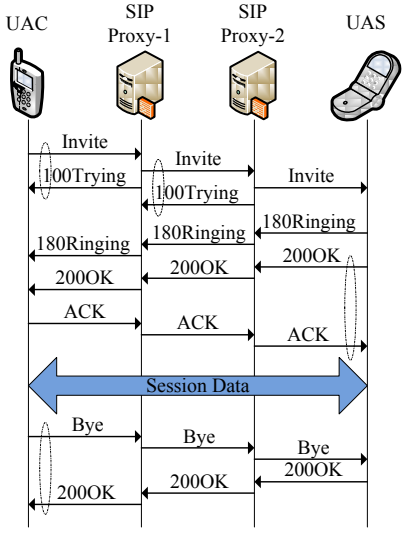


Fig. 1. A typical procedure of session establishment.

Fig. 1 illustrates a basic operation of a SIP system. To set up a call, a user agent client (UAC) sends an "Invite" request to a user agent server (UAS) via the two proxy servers. The proxy server returns a provisional "100 (Trying)" response to confirm the receipt of the "Invite" request. The UAS returns a "180 (Ringing)" response after confirming that the parameters are appropriate. It also evicts a "200 (OK)" message to answer the call. The UAC sends an "ACK" response to the UAS after receiving the "200 (OK)" message. Finally the media communication between the UAC and the UAS is established through the call session. The "Bye" request is generated to close the session, thus terminating the communication.

SIP introduces a retransmission mechanism to provide the reliable message delivery [1, 4]. In practice, a SIP sender uses timeout to detect message losses. One or more retransmissions would be triggered if the corresponding reply message is not received in predetermined time intervals.

When the message arrival rate exceeds the service capacity at a SIP server, overload occurs and the queue builds up, which may result in a long queuing delay and trigger unnecessary retransmissions from its upstream servers. The redundant retransmissions increase the CPU loads of both the overloaded server and its upstream servers. Such overload propagation may bring potential SIP network collapse [5-18].

SIP RFC 3261 [1] suggests that the SIP retransmission mechanism should be disabled for hop-by-hop transaction when running SIP over TCP to avoid redundant retransmissions at both SIP and TCP layer [1]. However, nearly all vendors choose to run SIP over UDP instead of TCP for the following reasons [5-19]: (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) Designed for preventing congestion caused by bandwidth exhaustion, the complex TCP congestion control mechanism provides little help for SIP overload which is caused by CPU constraint.

RFC 5390 [20] identified the various reasons that may cause server overload in a SIP network. These include poor capacity planning, component failures, 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.

The contributions of this paper are: (1) Extending classical control theory to model the impact of the retransmission rate on the queuing delay of an overload SIP server. As shown in Eq. (6), we have used the frequency domain technique to accomplish the modeling; (2) Developing a PI controller to mitigate the SIP overload by controlling retransmission rate

based on the round trip delay (i.e., the approximated queuing delay as discussed later on); (3) Providing the guidelines for obtaining PI controller parameters to guarantee the stability of the SIP overload control system. We perform OPNET simulation to validate the efficiency of the proposed control theoretical approach on SIP overload control.

II. Related Work

Recent collapses of SIP servers in carrier networks (e.g., Skype outage [21]) have motivated numerous overload control solutions (e.g., [5-15]). For example, both centralized and distributed overload control mechanisms for SIP were developed in [9]. Three window-based feedback algorithms were proposed to adjust the message sending rate of the upstream SIP servers based on the queue size or queuing delay [10]. 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 informs its upstream sending servers to reduce their original message sending rates. Such pushback control solution would increase the queuing delays of newly arrival original messages at the upstream servers, which in turn cause overload at the forwarding upstream servers. Overload may thus propagate server-by-server to sources and block large amount of calls which means the revenue loss for carriers.

Since unnecessary retransmissions caused by overload would exacerbate the overload [15], different from all existing solutions discussed above, our goal for mitigating the overload is to reduce the retransmission rate only. Controlling retransmission rate based on redundant retransmission ratio was proposed in [22]. However, redundant retransmission messages can only be detected after their corresponding response messages are received, such delay might lead to sluggish reaction and potential throughput loss. The queuing delay has been well accepted as a more reliable indicator of overload by some push-back solutions (e.g., [5, 10]), therefore, in this paper, we propose to control retransmission rate based on the queuing delay of an overloaded server which can be approximated by the round trip delay of its upstream server.

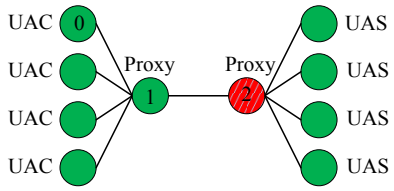


Fig. 2. SIP network topology with an overloaded downstream receiving Server 2 (which is marked with diagonal lines) and its upstream sending Server 1.

III. PI Controller For Mitigating SIP Overload

The topology of a real SIP network can be quite complex. Fig. 2 depicts a typical SIP network topology [9]. To focus our study on the interactions between overloaded receiving Server 2 and its upstream sending Server 1, we assume the upstream servers of Server 1 and the downstream servers of Server 2 have sufficient capacity to process all arrival messages without any delay. 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. We assume that the buffer sizes for all servers are large enough to hold all arrival messages. Given the proportionate nature and the general similarity of the retransmission mechanisms between the "Invite" and "non-Invite" messages in a typical session [1], this paper will focus on the hop-by-hop Invite-100(Trying) transaction.

A. Queuing Dynamics of Overloaded 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 [9, 12]. We can obtain the queuing dynamics for the message queue of Server 2 as

$$\dot{q}_2(t) = \lambda_2(t) + r_2(t) + \upsilon_2(t) - \mu_2(t), \quad (1)$$

where $q_2(t)$ denotes the queue size and $q_2(t)\geq 0$; $\lambda_2(t)$ denotes original message rate; $r_2(t)$ denotes retransmission message rate; $\upsilon_2(t)$ denotes response message rate; $\mu_2(t)$ denotes the message service rate.

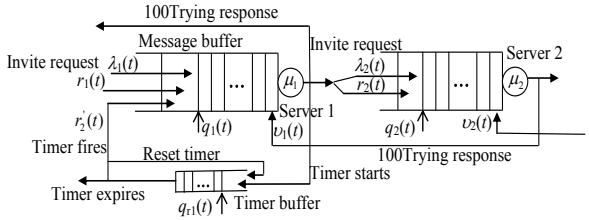


Fig. 3. Queuing dynamics of an overloaded server and its upstream server.

Like Eq. (1), we can obtain the queuing dynamics for the message queue of Server 1 as

$$\dot{q}_1(t) = \lambda_1(t) + r_1(t) + r_2'(t) + \upsilon_1(t) - \mu_1(t), \quad (2)$$

where $q_1(t)$ denotes the queue size and $q_1(t)\geq 0$; $\lambda_1(t)$ denotes arbitrarily distributed original message rate, which may follows Poisson, Pareto, Gamma or Lognormal distribution; $r_1(t)$ denotes retransmission rate corresponding to $\lambda_1(t)$; $r'_2(t)$ denotes retransmission message rate generated by Server 1 for $\lambda_2(t)$; $\upsilon_1(t)$ denotes response message rate corresponding to $\lambda_2(t)$; $\mu_1(t)$ denotes arbitrarily distributed service rate.

When Server 2 performs its routine maintenance and reduces its service capacity for signaling messages, the original message rate $\lambda_2(t)$ is larger than the service rate $\mu_2(t)$, the queue size $q_2(t)$ tends to increase according to Eq. (1) (i.e., $\dot{q}_2(t) > 0$). After a short period of overload, the queuing delay of Server 2 is long enough to trigger the retransmissions $r'_2(t)$ which enter the queue of Server 1. If the total new message arrival rate of $\lambda_1(t)$, $\upsilon_1(t)$ and $r'_2(t)$ is larger than the service rate $\mu_1(t)$, the queue size $q_1(t)$ would increase (i.e., $\dot{q}_1(t) > 0$, as indicated by Eq. (2)) and may trigger the retransmissions $r_1(t)$ to propagate the overload from Server 2 to Server 1. After queuing and processing delay at Server 1, the retransmitted messages $r'_2(t)$ enter Server 2 as $r_2(t)$ to increase the queue size $q_2(t)$ more quickly (as described by Eq. (1)), thus making the overload at Server 2 much worse.

B. Overload Control Plant

In order to present our overload control mechanism more clearly, we assume that the upstream Server 1 can process all arrival messages without any delay before the overload is

propagated from its downstream Server 2, i.e., $\lambda_2(t)=\lambda_1(t)$ and $r_2(t)=r'_2(t)$. Therefore, we can update the queuing dynamics for the message queue of Server 2 as

$$\dot{q}_2(t) = \lambda_1(t) + r'_2(t) + \upsilon_2(t) - \mu_2(t) . \quad (3)$$

Then the corresponding message queuing delay of Server 2 becomes

$$\dot{\tau}_2(t) = [r'_2(t) + \lambda_1(t) + \upsilon_2(t) - \mu_2(t)] / \mu_2(t) . \quad (4)$$

Each request message corresponds to a response message, and the time to process a response message is typically much smaller than a request message [1, 10]. Thus we can use the request message service rate (i.e., the response message rate $\upsilon_1(t)$) to approximate the total service rate $\mu_2(t)$. Then we can approximate the queuing delay of Server 2 as

$$\dot{\tau}_2(t) \approx [r'_2(t) + \lambda_1(t) + \upsilon_2(t) - \upsilon_1(t)] / \upsilon_1(t) , \quad (5)$$

We assume that the system is locally stable and therefore the uncontrolled variables λ_1, υ_2 and υ_1 are constant around an operating point. Truncating the component $(\lambda_1+\upsilon_2-\upsilon_1)$ only has impact on the zeros of closed-loop overload control system, and thus will not reduce the system stability margin. Therefore, the transfer function between the instantaneous queuing delay $\tau_2(t)$ and the retransmission rate $r'_2(t)$ can be approximated as

$$P(s)=\tau_2(s)/r'_2(s)=\mathcal{L}\{\tau_2(t)\}/\mathcal{L}\{r'_2(t)\}\approx 1/(\upsilon_1 s). \quad (6)$$

Such control plant approximation has been widely adopted in industrial process control system design, e.g., [23, 24], which has recently found its application in network traffic control, e.g., [25]. The validity of this approximation has also been verified by our performance evaluation in Section IV.

As processing signaling messages are typically CPU capacity constrained rather than bandwidth constrained, for the round trip delay between an upstream server and its overloaded downstream server, the queuing delay is dominant, while transmission and propagation delay are negligible [10]. Therefore, the round trip delay between the upstream Server 1 and the downstream Server 2 can *approximate* the queuing delay τ_2 of the overloaded downstream Server 2 when the overload happens.

C. The PI Rate Controller Design

As the retransmitted messages $r'_2(t)$ may increase queue sizes at both Server 1 and Server 2 and bring the overload to both servers, we propose a PI rate control algorithm that can mitigate the overload by reducing the retransmission rate $r'_2(t)$, thus preventing the network collapse (such as Skype outage [21]) caused by overload propagation.

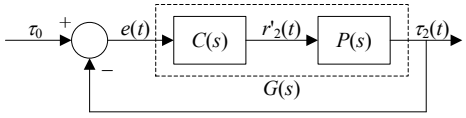


Fig. 4. Feedback SIP overload control system.

Fig. 4 depicts a feedback SIP overload control system, where the overload control plant $P(s)$ represents the interaction between an overloaded downstream receiving server and its upstream sending server, and adaptive PI rate controller $C(s)$ is located at the upstream server for mitigating the overload to clamp the round trip delay of the upstream server (i.e., the approximated queuing delay of the overloaded downstream server) below a target value, when the overload is anticipated at its downstream server.

Based on the instantaneous round trip delay of the upstream server (which approximates the queuing delay of the overloaded downstream server) $\tau_2(t)$, the retransmission rate $r'_2(t)$ can be obtained by the following PI control algorithm expressed via

$$\begin{aligned} r'_2(t) &= K_P e(t) + K_I \int_0^t e(\eta)\mathrm{d}\eta \\ &= K_P(\tau_0 - \tau_2(t)) + K_I \int_0^t (\tau_0 - \tau_2(\eta))\mathrm{d}\eta \end{aligned} . \quad (7)$$

where K_P and K_I denote the proportional gain and integral gain of the PI controller at the upstream server, and τ_0 denotes the target round trip delay. In the real-time implementation, a retransmission probability is equal to the ratio between the retransmission rate $r'_2(t)$ and the measured timer expiration rate. Since the original message rate is maintained to keep the revenue and achieve the user satisfaction in case of the overload, PI controller aims at clamping the round trip delay of the upstream server below a desirable target delay τ_0 rather than reaching the target delay τ_0.

It can be easy to obtain the transfer function between the retransmission rate $r'_2(t)$ and the round trip delay deviation $e(t)$ as $C(s)=K_P+K_I/s$. (8)

Then the open-loop transfer function of overload control system becomes $G(s)=C(s)P(s)=(K_P+K_I/s)/(\upsilon_1 s)$. (9)

From the definition on the phase margin ϕ_m of $G(s)$ [26], we can obtain

$$\arctan\left[\frac{K_P\omega_g}{K_I}\right] - \frac{\pi}{2} - \frac{\pi}{2} = -\pi + \phi_m , \quad (10)$$

$$\frac{\sqrt{K_P^2\omega_g^2 + K_I^2}}{\omega_g}\frac{1}{\upsilon_1\omega_g} = \frac{\sqrt{K_P^2\omega_g^2 + K_I^2}}{\upsilon_1\omega_g^2} = 1 , \quad (11)$$

where ω_g is the gain crossover frequency of the overload control system. It is well known that a positive phase margin ($\phi_m>0$) can guarantee the stability of the control system in accordance with the Nyquist Stability Theorem [23-26]. A common control engineering practice suggests an interval of phase margin as $30^0\le\phi_m\le 60^0$ for a good response [24]. To simplify our controller design, we set $\omega_g=1$ based on numerous simulation results. Thus we can obtain K_P and K_I as

$$K_P = \frac{\upsilon_1 \tan(\phi_m)}{\sqrt{1+\tan^2(\phi_m)}} , \quad K_I = \frac{\upsilon_1}{\sqrt{1+\tan^2(\phi_m)}} . \quad (12)$$

So far we have assumed υ_1 be constant. In reality, this is not necessarily true. If the PI controller parameters K_P and K_I remain unchanged, the varying service rate of the overloaded server may drive the phase margin out of its desirable interval. Lemma 1 shows the impact of the response message rate on the phase margin of the overload control system. We omit the proof due to page limit.

Lemma 1: ***If*** current response message rate υ'_1 is larger than previous response message rate υ_1 (that is, $\upsilon'_1>\upsilon_1$) and the PI controller is designed based on υ_1, ***then*** the overload control system with υ'_1 will have less phase margin than that with υ_1 (that is, $\phi'_m<\phi_m$).

To achieve a satisfactory performance, we self-tune PI controller when dramatic change of message response rate υ_1 exceeds a specified interval. Summary of our overload control algorithm is shown in Fig. 5.

Overload Control Algorithm

When each retransmission timer fires or expires
Retransmit the message with a retransmission probability corresponding to a retransmission rate r'_2 calculated by a PI rate controller

Adaptive PI rate control algorithm:

(1) Specify target queuing delay τ_0 and phase margin ϕ_m; Set the initial values for υ_1; Obtain PI controller gains using Eq. (12).

(2) Measure τ_2 and υ_1 upon response message arrivals.

(3) If $\upsilon_1>1.5\upsilon_{10}$ or $\upsilon_1<0.5\upsilon_{10}$, self-tune PI controller gains using Eq. (12), then update $\upsilon_{10}=\upsilon_1$; Otherwise, PI controller remains unchanged.

(4) Calculate the retransmission rate r'_2 using Eq. (7); Go to Step (2).

Varying parameter:

τ_2 : Instantaneous round trip delay
υ_1 : Response message rate
K_P : Proportional gain of PI controller
K_I : Integral gain of PI controller
r'_2 : Message retransmission rate

Fixed parameter:

τ_0 : Target round trip delay
ϕ_m : Phase margin

Fig. 5. A PI rate control algorithm for mitigating SIP overload.

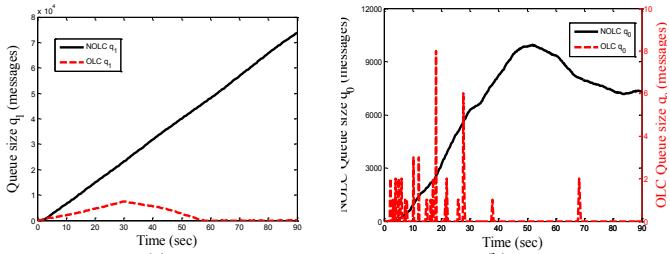
IV. Performance Evaluation And Simulation

To verify our PI controller, we conducted OPNET simulations to observe the dynamic behaviour of the overloaded server and its upstream server. During our experiment, four user agent clients generated original request messages with equal rate, and then sent them to four user agent servers via two proxy servers, as shown in Fig. 2. The message generation rates are Poisson distributed [1]. The message service rate of each server is also Poisson distributed. Since processing a response message takes much less time than processing a request message, we set the ratio α of the mean processing time of a response message to that of a request message as α=0.5. 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 a UAC or a UAS 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 target delay τ_0 is set as 0.5s. The phase margin is set as 45^0. The Internet Traffic Report indicates that current global packet loss statistic averaged 8% packet loss [27], considering possible message corruption in the SIP layer, average message loss probability is set as 10%.

To demonstrate the effectiveness of our overload control solution, two typical overload scenarios were simulated: (1) Overload at Server 1 due to a demand burst; (2) Overload at Server 2 due to a server slowdown. The simulation time is 90s, and the 1st-time retransmission timer is T_1=500ms [1]. In each scenario, we performed our simulations with overload control algorithm and without overload control algorithm separately. In all the simulation plots in this paper, 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

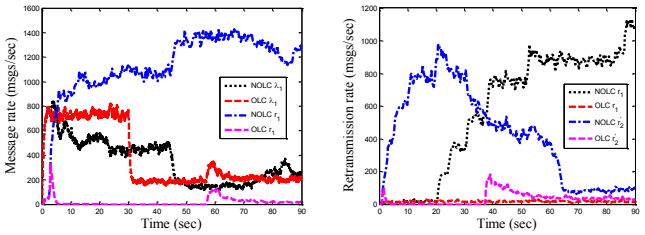
In this scenario, the mean message generation rate for each user agent client was 200 messages/sec (i.e., λ_1=800 messages/sec, emulating a short surge of user demands) from time t=0s to t=30s, and 50 messages/sec (i.e., λ_1=200 messages/sec, emulating regular user demands) from time t=30s to t=90s. The mean service capacities of two proxy servers were C_1=C_2=1000 messages/sec.



(a) (b)

Fig. 6. (a) Queue size q_1 (messages) of Server 1 versus time. (b) Queue size q_0 (messages) of UAC 0 versus time.

Figs. 6 and 7(a) show the dynamic behaviour of overloaded Server 1 and its upstream UAC 0.



(a) (b)

Fig. 7. (a) Scenario A: Moving average original message rate λ_1 (messages/sec) of Server 1 and moving average retransmission rate r_1 (messages/sec) for Server 1 versus time. (b) Scenario B: Moving average retransmission rate r_1 (messages/sec) for Server 1 and moving average retransmission rate r'_2 (messages/sec) for Server 2 versus time.

Without overload control algorithm applied, it is easy to see from Fig. 6(a) that Server 1 became CPU overloaded immediately and the overload deteriorated as time evolves, leading to the eventual crash of Server 1. Since the aggregate service capacity of four user agent clients was larger than that of proxy Server 1, the queue size of each user agent client decreased slowly (see Fig. 6(b)) after new original message generation rates decreased.

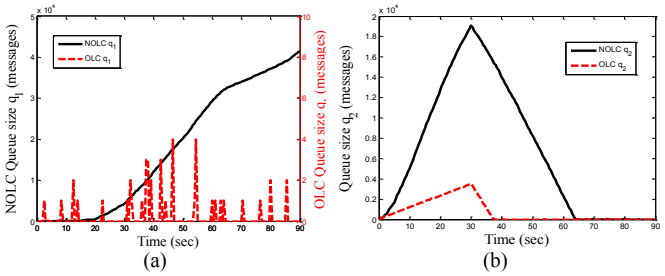
Our overload control algorithm made the queue size of Server 1 increase slowly during the period of the demand burst, and cancelled the overload at Server 1 within 27s (11s faster than the overload control algorithm in [22]) after the new user demand rate reduced at time t=30s.

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

[1] The workload in the real SIP networks can be arbitrarily distributed, which may follows Poisson, Pareto, Gamma or Lognormal distribution. Poisson distributed message arrival rate and service rate are widely adopted by most existing research work (e.g., [10]).

t=90s, 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 user agent client was 50 messages/sec.



(a) (b)

Fig. 8. (a) Queue size q_1 (messages) versus time. (b) Queue size q_2 (messages) versus time.

Without overload control algorithm applied, Figs. 7(b) and 8 demonstrate that Server 2 became overloaded first, which was followed by a later overload at Server 1. The queue size at Server 1 increased more quickly due to the extra work load for handling retransmissions for both Server 1 and Server 2. After Server 2 resumed its normal service at time t=30s, Server 1 and Server 2 had the same service capacity. Because 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. Eventually the overload at Server 2 was cancelled, while the overload at Server 1 persisted (see Fig. 8).

With our overload control algorithm applied, 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. After Server 2 resumed its normal service, it only spent 7s (2s faster than the overload control algorithm in [22]) to cancel the overload and the buffer became empty at time t≈37s.

V. Conclusions

In order to study the impact of the retransmission rate on the queuing delay of an overloaded server, we have employed a control-theoretic approach to model the interaction between the overloaded downstream receiving server and its upstream sending server as a feedback control system. Then we have developed a novel PI rate control algorithm to mitigate the overload by reducing retransmission rate only, while maintaining the original message rate to avoid excessive revenue loss.

By analyzing queuing dynamics and performing OPNET simulations, we have demonstrated that without overload control algorithm applied, the overload at downstream server may propagate or migrate to its upstream servers eventually. Our overload control algorithm can cancel the short-term overload effectively and prevent the overload propagation. Without requiring modification in the SIP header and the cooperation among different carriers in different countries, any carrier can freely implement our proposed solution in its SIP servers to avoid potential widespread server crash.

Acknowledgment

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

References

[1] J. Rosenberg et al., "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] 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.
[5] E. Noel and C.R. Johnson, "Initial simulation results that analyze SIP based VoIP networks under overload," in *Proceedings of 20th International Teletraffic Congress*, Ottawa, Canada, 2007, pp. 54-64.
[6] E. Noel and C.R. Johnson, "Novel Overload Controls for SIP Networks," in *Proceedings of 21st International Teletraffic Congress*, Paris, France, 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] I. Dacosta, V. Balasubramaniyan, M. Ahamad, and P. Traynor, "Improving Authentication Performance of Distributed SIP Proxies," Proceedings of IPTComm, Atlanta, GA, July 2009.
[15] J. Sun, R.X. Tian, J.F. Hu, and B. Yang, "Rate-based SIP Flow Management for SLA Satisfaction," in *Proceedings of* 11th International Symposium on Integrated Network Management (*IEEE/IFIP IM*), New York, USA, June 2009, pp. 125-128.
[16] V. Hilt and H. Schulzrinne, "Session Initiation Protocol (SIP) Overload Control," *IETF Internet-Draft*, January 2011.
[17] 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.
[18] E.M. Nahum, J. Tracey, and C.P. Wright, "Evaluating SIP server performance," in *Proceedings of International Conference on Measurement and Modeling of Computer Systems* (*ACM SIGMETRICS*), San Diego, CA, US, 2007, pp. 349–350.
[19] 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.
[20] J. Rosenberg, "Requirements for Management of Overload in the Session Initiation Protocol," *IETF RFC 5390*, December 2008.
[21] R. Ando, "Internet phone and video service Skype went down in a global service outage," Reuters News, December 22nd, 2010.
[22] 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.
[23] W.K. Ho, T.H. Lee, H.P. Han, and Y. Hong, "Self-Tuning IMC-PID Control with Interval Gain and Phase Margin Assignment," *IEEE Trans. on Control Systems Technology*, **9**(3), May 2001, pp. 535-541.
[24] W.K. Ho, Y. Hong, A. Hansson, H. Hjalmarsson, and J.W. Deng, "Relay Auto-Tuning of PID Controllers Using Iterative Feedback Tuning," *Automatica*, **39**(1), January 2003, pp. 149-157.
[25] D. Katabi, M. Handley, and C. Rohrs, "Congestion Control for High Bandwidth Delay Product Networks", in *Proceedings of ACM SIGCOMM*, August 2002.
[26] K. Ogata, *Modern Control Engineering*, fourth edition, Prentice Hall, New Jersey, 2002.
[27] "Internet Traffic Report", http://www.internettrafficreport.com/, 2010.