Priority-Based Queue Management Scheme to Reduce Overloads on SIP Servers

Tadasuke Nozoe, Masahiko Noguchi, Minoru Sakuma, Kazuaki Misawa, and Mikio Isawa

***Abstract*—If a session initiation protocol (SIP) server receives enough SIP messages to exceed its processing capacity, it goes into overload. When a SIP server receives new SIP messages under overload, the SIP message queue in the server lengthens, which may cause significant throughput degradation in the network. An overload control mechanism using the 503 response code is specified in the SIP standard to improve throughput in the event of such an overload. However, it is known that this is insufficient to control overloading in a large SIP network. In this paper we propose a queue management scheme at a SIP server to improve throughput and prevent overload propagation when overload occurs. In the proposed scheme, SIP messages are dropped based on a dropping probability that is calculated by prioritizing the items in the SIP queue. We also show the effectiveness of the proposed scheme by simulation experiments.**

***Index Terms*—IMS, overload control, queue management, SIP.**

I. Introduction

A SIP can use either TCP or UDP [1]. Most vendors choose UDP as the transport protocol to avoid the overhead of state management such as the TCP three-way handshake [2]. When using UDP as the transport protocol, SIP supports application-level transmission mechanisms in order to achieve reliable delivery in the application layer [1]. If a sender of a SIP message does not receive the corresponding reply message until the retransmission timer expires, it will retransmit the SIP message.

If a SIP server receives a quantity of SIP messages that exceeds its processing capacity, it goes into overload. This can occur for many reasons, including emergency-induced call volume, flash crowds, DoS attacks, and component failures [3]. When a SIP server receives new SIP messages while in an overload state, the message queue in the SIP server lengthens, which may cause queuing delay or queue overflow. Such a delay or overflow may also cause an increase in the number of retransmissions from SIP-UAs and upstream SIP servers. Due to retransmissions, the processing load of both the overloaded SIP server and its upstream servers may be increased even more. Once a server enters the overload state, the overload propagates to its neighbors and eventually spreads throughout the network [4]. In this way, an overload may cause significant throughput degradation in a SIP network. Throughput is defined as the total number of completed calls per second, where a completed call is one whose execution sequence from session establishment to termination has been completed normally. When a call that did not receive any provisional responses (100 Trying or 180 Ringing) to an INVITE receives a final response (200 OK), it is also regarded as a completed call.

An overload control mechanism using the 503 (Service Unavailable) response code is specified in [1]. When an overloaded server receives a SIP request, it will respond with a 503 to the SIP-UA or the upstream sending server to refuse to process the request. However, the 503 response only stops the current request, so other SIP-UAs or upstream servers continue to send the request to the overloaded server. Consequently, the overloaded server continues to expend resources (such as CPU time) to respond with 503 responses, which amplifies the overload of the server. The overload control mechanism defined in [1], as explained above, is insufficient to control overloads in a large SIP network.

In this paper we propose a queue management scheme at a SIP server to improve throughput and provide steady service even when overload occurs. We also demonstrate the effectiveness of the proposed scheme with simulation experiments.

The rest of paper is organized as follows. In Section II, we briefly discuss related work on overload control in a SIP network. In Section III the proposed scheme is presented. Section IV describes the simulation model and Section V presents and discusses simulation results. In Section VI, we conclude the paper and discuss future work.

II. Related Work

More overload control mechanisms have been proposed for a SIP network than just using a 503 response [2].

M. Ohta proposed a queue management scheme that rejects SIP messages with low priority to increase throughput under overload conditions [5]. In this scheme, SIP servers maintain two queues, one a low-priority queue and the other a high-priority queue. Every SIP message is assigned to either the low-priority queue or the high-priority queue according to the priority class of the SIP message, so that an INVITE message is assigned to the low-priority class and other requests and response messages are assigned to the high-priority class. Only when the high-priority queue is empty are INVITE messages in the low-priority queue processed. By giving high priority to non-INVITE messages under overload, the termination of an established session is processed prior to establishing a new session. This results in higher throughput in the network when overload occurs. W. Zhu *et al.* pointed out that the scheme proposed in [6] is inefficient due to its use of two queues. They provided an example of this inefficiency in [6] as follows. When the

Manuscript received November 4, 2014; revised March 27, 2015.

The authors are with the NTT Network Service Systems Laboratories, Nippon Telegraph and Telephone Corporation, 3-9-11 Midori-Cho, Musashino-Shi, Tokyo, 180-8585 Japan (e-mail: nozoe.tadasuke@lab.ntt.co.jp).

DOI: 10.7763/IJCTE.2016.V8.1076

high-priority queue is full and the low-priority queue is not, the received non-INVITE messages must be dropped even if there is plenty of space in the low-priority queue. To maximize queue utilization under overload, they proposed that all SIP messages share the same queue, and every SIP method type is assigned a unique priority. In this scheme, the priority level indicates the order of SIP messages to be replaced under overload. When a SIP server receives a new SIP message and its queue is full, the SIP server tries to find an existing SIP message with a lower priority level than the received message by searching sequentially from the head of the queue to the tail. If a message with lower priority is found in the queue, that message is replaced by the received message. This ensures that SIP messages with a higher priority level are preferentially processed. A higher priority level is given to a SIP method type appearing at a later stage of the basic SIP sequence. This means that an INVITE request has the lowest priority level and a 200 OK (BYE) response has the highest.

The queue management scheme proposed in [6], however, needs to search the queue at every reception of a new SIP message when its queue is full. Assuming that SIP servers receive a large number of packets in an overload situation, the use of their resources on searching the queue may cause further overload.

III. Proposed Scheme

We propose a queue management scheme to solve the problem described in Section II.

The proposed scheme adopts a single-queue model to deal with the question of queue utilization efficiency discussed in [6]. It is also based on RED (Random Early Detection), which is a queue management scheme for a router [7]. In RED, packets are dropped based on a dropping probability that depends on the average queue length. In contrast to the original RED proposed in [7], which did not distinguish packet types, our proposed scheme for a SIP server enables the calculation of different dropping probabilities according to the SIP method type because [5] and [6] showed that queue management based on giving a priority to each SIP method type could improve throughput under overload. The proposed scheme can calculate different dropping probabilities according to the SIP method type even if the queue length is the same when receiving a new SIP message.

When a SIP server receives a SIP message, an average queue length is calculated by (1).

$$avg = (1 - w_q)avg + w_q q \tag{1}$$

where w_q is a weight to balance the impact of burst traffic and q is the current queue length.

The average queue length is compared with two thresholds, the minimum threshold $\min_{th}$, and the maximum threshold $\max_{th}$, and the dropping probability is estimated by the following (2).

$$p_a = \begin{cases} 0 & avg < \min_{th} \\ C_{method} \dfrac{P_b}{1 - count \cdot p_b} & \min_{th} \le avg \le \max_{th} \\ 1 & \max_{th} < avg \end{cases} \tag{2}$$

where *count* is the number of SIP messages enqueued since the last packet dropped. C_{method} is a coefficient defined for each SIP method type. The higher the priority of a SIP method type is, the lower the value of C_{method} because the dropping probability for messages with higher priority should be lower than for those with a lower priority. p_b is calculated by (3).

$$p_b = \frac{avg - \min_{th}}{\max_{th} - \min_{th}} \max_p \tag{3}$$

where $\max_p$ is the maximum value of p_b.

In the proposed scheme, SIP servers drop SIP messages according to the dropping probability p_a estimated by (2).

IV. Simulation Model

A. Network Model

The SIP network topology we have used in our simulation experiments is depicted in Fig. 1. In the simulation, all SIP servers in the network run as a stateless proxy server that just forwards SIP messages and does not manage the SIP transactions. The SIP transactions between two SIP-UAs are managed by the SIP-UAs themselves. SIP servers 1 and 5 are edge servers, and all SIP-UAs connect to either SIP server 1 or 5. The other SIP servers in the network form a core server, relaying SIP messages from the upstream server to the downstream server.

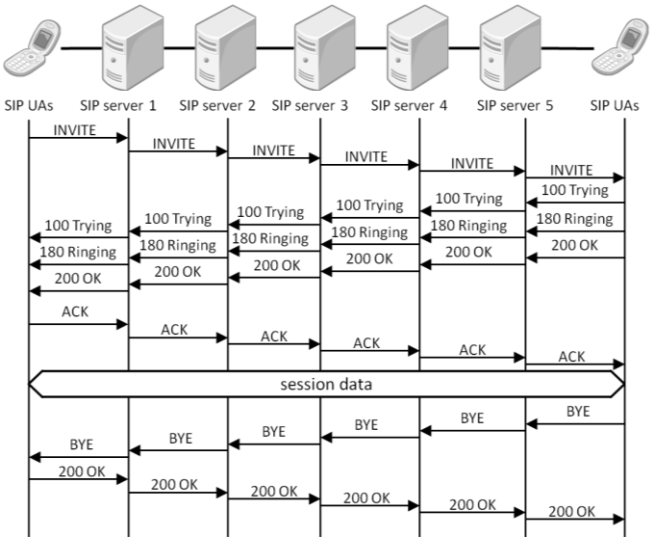

Fig. 1. Network topology and SIP sequence.

In the simulation, all sessions are initiated by SIP-UAs connecting to SIP server 1 and terminated by SIP-UAs connecting to SIP server 5. This means that only SIP-UAs connecting to SIP server 1 can send an INVITE message and only SIP-UAs connecting to SIP server 5 can send a BYE message, as shown in Fig.1. There is, naturally, no session between SIP-UAs connecting to the same edge server in the simulation.

SIP requests and responses sent from SIP-UAs or the upstream SIP server are enqueued in the SIP message queue. The messages in the queue are dequeued in a FIFO manner and processed by the CPU, then forwarded to the SIP-UA or the downstream SIP server.

The processing time of a SIP message depends on the SIP

message type. In general, the processing time of an INVITE message is longer than that of non-INVITE messages because a SIP server has to process it by checking the user profile, determining the next hop, and so on, when receiving an INVITE message. In the simulation, we assume that the processing time of an INVITE message is double that of a non-INVITE message.

We use UDP as the transport protocol. We assume that there is sufficient bandwidth between entities such as SIP-UAs and SIP servers in the network, so we do not consider any packet loss in the simulation other than that from dropping by the queue management scheme. We also ignore the transmission delay of the link between the entities because of the high bandwidth.

Each SIP server has a queue length of 200 messages, and processes an INVITE message in 2 ms and a non-INVITE message in 1 ms.

B. SIP-UA Model

Each SIP-UA connected to SIP server 1 sends an INVITE request to establish a new session, with timing following a Poisson distribution and giving an average arrival rate λ. When the SIP-UA connected to SIP server 5 receives this request, it sends provisional responses and also sends 200 OK after waiting for the ringing time according to an exponential distribution with T_a=5.0 seconds. The holding time for an established session follows an exponential distribution with T_s=30.0 seconds.

The SIP-UA supports Timer A, Timer B, Timer E, Timer F, a retransmit interval timer for an INVITE response, and a wait timer for ACK receipt, and can retransmit a SIP message by using the retransmission mechanism defined in [1]. The values of $T1$ and $T2$ are set to 0.5 seconds and 4.0 seconds respectively in the simulation (the default values proposed in [1]). In the simulation, when Timer B, Timer F or the wait timer for ACK receipt fires, SIP-UAs release the resource assigned to the dialog established between them. The SIP-UA stops retransmitting an INVITE request when receiving a provisional response (100 Trying or 180 Ringing). When SIP-UA haven't received a 200 OK response for T_{abdn} seconds after receiving a provisional response, the SIP-UA cancels establishing the session. The SIP-UA also releases the resource for the dialog if it does not receive a BYE request for T_{max} seconds after sending an ACK request because this situation is regarded as a session time-out. We set the values of T_{abdn} and T_{max} to 32.0 seconds and 300.0 seconds, respectively.

C. Parameter Values

Table I and Table II respectively summarize the values of the parameters for the proposed scheme and of C_{method} for each SIP method discussed in section III. We set C_{method} as follows. C_{INVITE} is set to a higher value than any other so as to increase the dropping probability of an INVITE message because it is shown in [5] and [6] that a high throughput can be achieved by giving the INVITE message lower priority. A SIP-UA retransmits an INVITE request until it receives a response (100 Trying, 180 Ringing, or 200 OK). In other words, it is possible to stop retransmitting if the SIP-UA just receives 100 Trying or 180 Ringing after sending an INVITE request, so we set C_{100} and C_{180} respectively to a higher value and a lower one. C_{ACK} and $C_{200(BYE)}$ are set to a higher value to avoid being dropped because an ACK request and a 200 OK response for a BYE request are not retransmitted in the manner described in [1].

TABLE I: SUMMARY OF PARAMETER VALUES

Parameter	Value
w_q	0.8
$\min_{th}$	0.7
$\max_{th}$	1.0
$\max_p$	0.2

TABLE II: VALUES OF C_{METHOD}

Parameter	Value
C_{INVITE}	5.0
C_{100}	3.0
C_{180}	0.1
$C_{200(INVITE)}$	1.0
C_{ACK}	0.1
C_{BYE}	1.0
$C_{200(BYE)}$	0.1

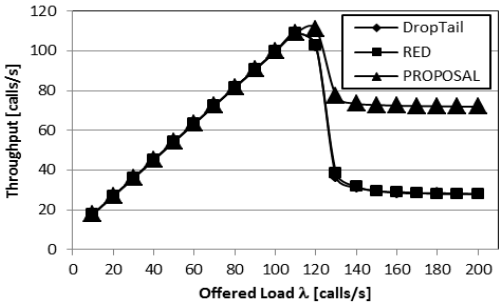

Fig. 2. Throughput as a function of the average arrival rate λ.

V. SIMULATION RESULTS

A. Throughput Characteristics

Fig. 2 shows throughput as a function of the average arrival rate λ. In order to present the effectiveness of the proposed scheme, we compare it with two schemes, the Drop Tail scheme, which is a basic queue management scheme, and the RED scheme, which is similar to our proposal except that the values of each C_{method} are set to 1.0.

Throughput drops significantly at about 120 calls per second (cps) as shown in Fig. 2. This means that overload occurs in the SIP network as the network receives a quantity of SIP messages from SIP-UAs that exceeds the capacity of the network. Fig. 2 indicates that the proposed scheme improves throughput compared to the other schemes under overload conditions, and the throughput of RED is plotted in much the same way as that of Drop Tail. We conclude from the results that dropping SIP messages based on the priorities proposed in Section III improves throughput.

B. Overload Propagation

Fig. 3 and Fig. 4 show the queue utilization as a function of time in Drop Tail and the proposed scheme. In these figures, the horizontal axis and the vertical axis represent the SIP server's number as shown in Fig. 1, and time. The shading in these figures corresponds to the queue utilization of the SIP server.

As shown in Fig. 3 and Fig. 4, the queue utilization of SIP server 1 is higher than that of the other servers because SIP server 1 receives a large variety of both original INVITE

requests and retransmitted requests from the SIP-UAs when overload occurs. In Fig. 3, we can see that the Drop Tail scheme causes overload propagation from a server to its neighbor with time. Fig. 4, on the other hand, indicates that the proposed scheme does not cause such propagation, so it can prevent overload propagation in a SIP network.

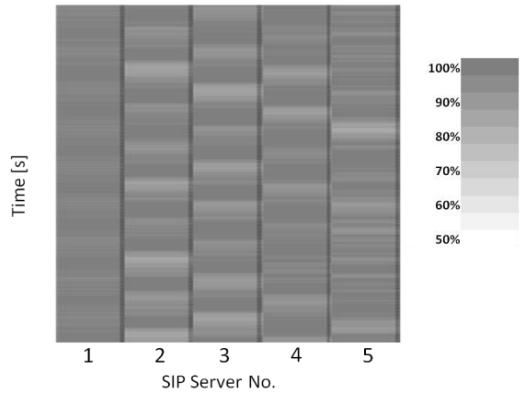

Fig. 3. Queue utilization in Drop Tail.

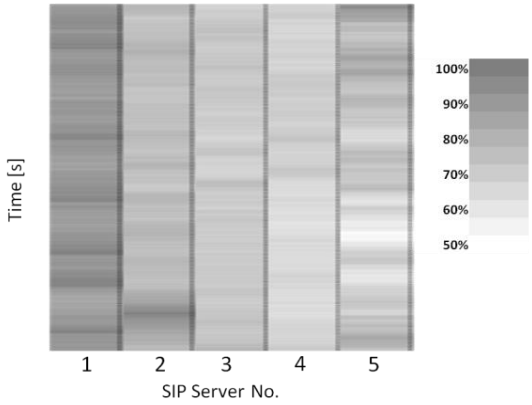

Fig. 4. Queue utilization in proposed scheme.

We assume that overload propagation occurs in Drop Tail due to the SIP retransmission mechanism. When many SIP-UAs send an INVITE request to SIP server 1 at the same time, and the server drops these requests nearly simultaneously because of a filled queue, SIP-UAs retransmit the request repeatedly (with the same timing) until they receive a response. In the proposed scheme, a SIP server drops messages according to the dropping probability, which delays the retransmission of messages from SIP-UAs. The proposed scheme, therefore, can prevent overload propagation in a SIP network.

VI. Conclusion

In this paper we have proposed a queue management scheme for a SIP server to improve throughput under overload conditions, which is based on prioritizing SIP methods by type. The proposed scheme has the advantage that it is not necessary to search the SIP message queue at every reception of a new SIP message when the queue is full, in contrast to the scheme proposed in [6].

We have presented simulation results which suggest that the proposed scheme can improve throughput under overload and prevent overload propagation.

We employed a stateless SIP proxy server as the SIP server in the simulation. In our future work, we would like to apply the proposed scheme to a stateful SIP server that retains its state for a dialog or a B2BUA SIP server that manages multiple dialogs. In the simulation, we used five servers connected in series to examine overload propagation characteristics, so we still need to evaluate the proposal with other network topologies. Finally, we plan to explore how throughput and overload propagation are affected by variation of the parameters in the proposed scheme.

References

[1] J. Rosenberg, H. Schulzrinne, G. Camarillo, A. Johnston, J. Peterson, R. Sparks, M. Handley, and E. Schooler, "SIP: Session initiation protocol," in *Proc. IETF, RFC 3261*, 2002.
[2] Y. Hong, C. Huang, and J. Yan, "A comparative study of SIP overload control algorithms," *Network and Traffic Engineering in Emerging Distributed Computing Applications*, pp. 1-20, 2012.
[3] J. Rosenberg, "Requirements for management of overload in the session initiation protocol," in *Proc. IETF, RFC 5390*, 2008.
[4] V. Hilt and I. Widjaja, "Controlling overload in networks of SIP servers," in *Proc. IEEE ICNP*, 2008, pp. 83-93.
[5] M. Ohta, "Overload protection in a SIP signaling network," in *Proc. ICISP '06*, 2006.
[6] W. Zhu, A. K. A. Hamid, Y. Kawahara, T. Asami, and Y. Murata, "Automatic originator regulation of IMS multiple traffic by stateless signaling prioritization," in *Proc. 2012 IEEE GLOBECOM*, 2012, pp. 2846-2851.
[7] S. Floyd and V. Jacobson, "Random early detection gateways for congestion avoidance," *IEEE/ACM Trans. on Networking*, vol. 1, no. 4, pp. 397-413, 1993.

Tadasuke Nozoe was born in Fukuoka, Japan, in 1982. He received a master of information science degree from the Kyushu University in Fukuoka, Japan, in 2007. He joined Nippon Telegraph and Telephone West Corporation (NTT West) in Osaka, Japan, in 2007. In 2009, he transferred to NTT Network Service Systems Laboratories in Tokyo, Japan, where he has been engaged in the research and development of next generation network.

Masahiko Noguchi was born in Niigata, Japan, in 1968. He received a master of engineering degree from the Niigata University in Niigata, Japan in 1992. He joined NTT Network Service Systems Laboratories in 1992. He worked on the development of switching system of PSTN. He has participated in the research and development of Carrier Grade Middleware. He is a member of IEICE of Japan.

Minoru Sakuma was born in Japan, in 1972. He received a M.A. degree in media and governance from Keio University, Tokyo in 1996. In 1996, he joined NTT Network Service Systems Laboratories. He worked on the development of PSTN Intelligent Networks and the research of privacy control technologies. He has been engaged in the research and development of next generation network.

Kazuyuki Misawa was born in Chiba, Japan, in 1974. He received a master of science degree from the Tokyo Institute of Technology in Tokyo, Japan, in 1999. He joined Nippon Telegraph and Telephone East Corporation (NTT East) in 1999. He has been with the NTT Network Service Systems Laboratories as a senior research engineer since 2012, where he has worked on next generation network.

Mikio Isawa was born in Japan, in 1964. He received a master of engineering from the Tokyo University of Agriculture and Technology in Tokyo, Japan, in 1989. He has been with the NTT Network Service Systems Laboratories since 1989, where he is currently a senior research engineer. He has conducted research and development widely in telecommunication systems such as ATM, STP, SIP and so on.