Impact of Retransmission Mechanism on SIP Overload: Stability Condition and Overload Control

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*—SIP (Session Initiation Protocol) has been widely adopted as a signaling protocol to establish, modify and terminate media sessions between end-users in the Internet. SIP introduces a retransmission mechanism to ensure the reliability of its real-time message delivery. However, retransmission can make server overload worse, leading to server crashes in SIP-based carrier networks (e.g. Skype). In order to study the impact of retransmission mechanism on SIP overload, in this paper, we create a discrete time fluid model to describe the queuing dynamics of an overloaded SIP server. Then we derive a sufficient stability condition that a SIP server can handle the overload effectively under the retransmission mechanism. Fluid model allows us to run fluid-based Matlab simulation directly to evaluate the overload performance. Event-driven OPNET simulation was also conducted to validate our fluid model. Our simulation results demonstrate that: (1) the sufficient stability bound is quite tight. The bound indicates that effective CPU utilization as low as 20% can still lead to an unstable system after a short period of demand burst or a temporary server slowdown. Resource over-provisioning is not a viable solution to the server crash problem; (2) by satisfying the stability condition, the initial queue size introduced by a transient overload can avoid a system crash. Such stability condition can help the operator to determine whether and when to activate overload control mechanism in case of heavy load. A simple overload control solution is also proposed.

Index Terms*—-SIP, SIP Overload, SIP Overload Control, SIP Retransmission, Stability Condition, CPU Utilization

I. INTRODUCTION

SIP (Session Initiation Protocol) [1] has been widely deployed for significantly growing session-oriented applications in the Internet, such as Voice-over-IP, instant messaging and video conference. As a signaling protocol, SIP is responsible for creating, modifying and terminating sessions in a mutual real-time communication [2]. 3GPP (3rd Generation Partnership Project) has adopted SIP as the basis of the IMS (IP Multimedia Subsystem) architecture [3-5].

Fig. 1 illustrates a simplified configuration of a SIP network which consists of two basic elements: UA (User Agent) and P-Server (Proxy Server) [1]. A UA may perform two roles: in the UAC (User Agent Client) role, the originating UA sends requests; in the UAS (User Agent Server) role, the terminating UA receives requests and sends responses. The task of a P-server is to receive SIP requests and forward them to the terminating UA (or to another P-server that is closer to the terminating UA). Each P-server is assigned to serve multiple individual UAs. UA and P-server cooperate to establish, modify and terminate media sessions.

Manuscript received December 15, 2010; revised March 31, 2011; accepted March 31, 2011.

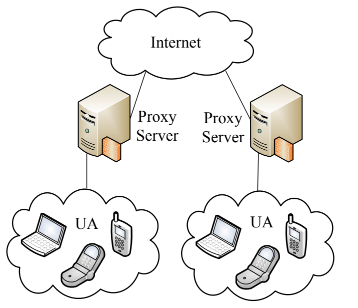


Figure 1. Simplified configuration of a SIP network.

SIP is designed to be an application layer protocol independent of the underlying transport mechanism which may be TCP (Transmission Control Protocol) or UDP (User Datagram Protocol). SIP introduces a retransmission mechanism (which will be reviewed briefly in Appendix A) to maintain its reliability by retransmitting lost SIP messages either end-to-end or hop-by-hop [5, 6]. A SIP source uses a delay to detect a message loss. It would produce one or more retransmissions if the corresponding reply message is not received in a predetermined time interval. If a retransmission triggered by a delay caused by the overload, it would introduce the overhead rather than reliability into the network. Such redundant retransmissions increase the memory and CPU load for a SIP server, which may cause a system overload and deteriorate the signaling performance [6-20]. In an overload situation, the throughput drops down to a small fraction of the original processing capacity, thus poses a serious problem for a SIP network [12]. This kind of behaviour has happened in real carrier networks where large scale collapses of SIP servers have been observed in present of sporadic SIP traffic burst (e.g., emergency-induced call volume) [13].

A SIP server can be overloaded due to various reasons such as poor capacity planning, dependency failures, component failures, avalanche restart, flash crowds, denial of service attacks, etc., as indicated by RFC 5390 [21]. In general, a short period of demand burst or a server slowdown may bring a server overload and lead to server crash. The built-in SIP overload control mechanism has proven to be ineffective in practice, because it attempts to mitigate the overload by rejecting some calls, but the cost of rejecting a SIP session is comparable with the cost of serving a session.

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, TCP layer lacks awareness of application context at SIP layer. TCP flow control mechanism cannot prevent SIP overload collapse, as indicated by recent experimental evaluation on SIP-over-TCP overload behaviour in [14]. Therefore, almost all real SIP networks run SIP over UDP mainly because the following reasons [9-23]: (1) TCP is optimized for accurate delivery by sacrificing its timeliness which is a critical requirement for real-time application such as SIP; (2) SIP works at application layer while TCP works at transport layer. Even TCP can provide reliability at transport layer, SIP messages can still be dropped or corrupted while being processed at application layer; (3) TCP keeps retransmitting outstanding packets until an ACK is received. Each retransmitted packet is pushed to the application layer to be a SIP message which costs extra CPU time and introduces more delay, therefore making CPU overload worse.

The contributions of this paper are: (1) Deriving a sufficient stability condition that a SIP server can handle the overload effectively under the retransmission mechanism; (2) Developing a discrete-time fluid model to reduce significantly simulation effort; (3) Comparing the results of both fluid-based Matlab simulation and event-driven OPNET simulation to demonstrate that the fluid-based simulation is relatively accurate and scalable for evaluating the performance of a SIP network; (4) Performing fluid-based simulation and event-driven simulation to verify that the stability bound is quite tight, or an initial queue size (created by a transient overload) which is 5% higher than the bound will bring a server crash. An effective CPU utilization as low as 20% cannot prevent a SIP server from overload during a short period of maintenance service and such overload continues to spread even after the normal service resumes; (5) Proposing a simple overload control solution to prevent a SIP server from overload collapse.

The paper is organized as follows. Section II reviews the related work on SIP overload and discusses simulation approaches. Section III analyzes the queuing dynamics of an overloaded SIP server under retransmission mechanism. Section IV derives a stability condition for SIP retransmission mechanism in the case of server overload. Section V evaluates the performance of an overloaded server. An overload control algorithm is proposed and validated in Section VI. Some conclusions are made in Section VII.

II. Related Work

A large scale collapse of SIP servers would result in widespread outage similar in impact to that recently reported by Skype [24]. Such a major risk has motivated a surge of research attention to SIP overload control. Experiment evaluations demonstrate that the retransmission mechanism can deteriorate the overload performance [12, 19]. Thus it is necessary to investigate the impact of the retransmission mechanism on the SIP overload. A demand burst or routine server maintenance such as database synchronization may accumulate the signaling messages to create a long queue. Excessive queuing delay, introduced by a long initial queue size, may continue to trigger the redundant retransmissions to crash the server after an overloaded server resumes its normal service with a low effective CPU utilization. It would be interesting to find a sufficient stability condition for the initial queue size, which indicates whether the SIP server can handle overload effectively. Such stability condition can help the SIP operator to decide whether and when to activate the overload control algorithm. It also can help researchers propose more effective solutions to avoid SIP overload collapse caused by the SIP retransmissions.

Some attempts have been made to avoid overload collapse in SIP networks (e.g., [9-19]). For example, three window-based feedback algorithms were proposed to adjust the message sending rate of the upstream SIP servers based on the queue length [12]. Both centralized and distributed overload control mechanisms for SIP were investigated in [13]. Retry-after control, processor occupancy control, queue delay control and window based control were proposed to improve goodput and prevent overload collapse in [10]. However, these overload control proposals suggested that the overloaded receiving server advertises to its upstream sending servers to reduce their 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 upstream servers. Overload may thus propagate server-by-server to sources and block large amount of calls which means revenue loss for carriers. Similarly, a small buffer size has been proved to be a simple overload control mechanism at a cost of arbitrarily high call rejection rate and revenue loss in [25].

When retransmissions are caused by overload rather than message loss, they would bring extra overhead instead of reliability to the network and exacerbate the overload [19]. Different from all existing solutions discussed above, we propose to reduce the retransmission rate instead of reducing original message sending rate of the upstream servers. This will mitigate the overload while maintaining original message sending rate, which leads to less blocking calls and more revenue for carriers.

Event-driven simulation has been widely used for evaluating network performance. Its computation cost grows linearly with network sizes and message volumes [26]. When event-driven simulation is used to evaluate a SIP network, each outstanding SIP message requires a timer being maintained. When an overload happens, outstanding messages are built up, and the simulator needs to increase the number of timers dramatically in order to track message retransmissions. Tracking and manipulating these timers consume large amount memory and CPU time which make the simulation process extremely slow, thus in some cases, cause the simulator to crash and terminate simulation unexpectedly. In order to simplify the CPU-consuming timer-tracking process, fluid-based simulation tracks time slot instead of individual messages. Messages arriving within the same time slot are aggregated and processed together. This will greatly simplify the complexity caused by large number of messages and allow smooth scalability by choosing different granularities as required.

III. Dynamics Of SIP Retransmission Mechanism

A real SIP network consists of a series of geographically distributed P-servers and a large amount of UAs. Each P-server is responsible for setting up a session call between two UAs. It forwards the requests and also generates a provisional response to confirm the receipt of every request from the upstream sender (an originating UA or a P-server) [1]. It provides a retransmission mechanism to guarantee a reliable delivery of a SIP message [5]. However, the arrival of too many SIP messages may cause an unnecessary queuing delay, stimulate redundant retransmissions and accelerate the overload, thus eventually bring down the entire network [12]. Therefore, it is necessary to describe the queuing dynamics of an overloaded SIP server (e.g., [8, 25]), before we derive a stability condition for SIP retransmission mechanism.

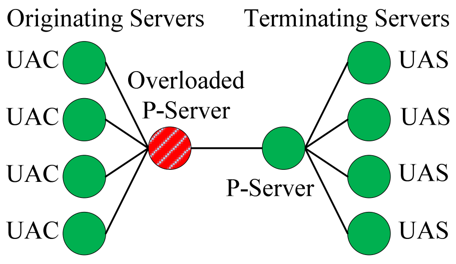


Fig. 2. SIP network topology with an overloaded P-server (which is marked in diagonal lines) and its multiple upstream originating servers.

When overloads happen in the network, at any time, one of the servers will be the most overloaded one among all the overloaded servers. Without loss of generality, we consider a typical SIP network which consists of an overloaded P-server and its multiple upstream originating servers [12], as shown in Fig. 2. For a clear presentation, we use difference equations to describe the queuing dynamics of an overloaded P-server. In our discrete time model, we make the following assumptions according to SIP RFC [1]:

(a) We investigate the retransmissions which are mainly caused by long queuing delay of the overloaded server. Therefore, for the round trip response time between the overloaded server and its neighbouring server, the queuing and processing delays are dominant, while transmission and propagation delay are negligible [13]. This assumption is valid because signaling messages are typically CPU capacity constrained rather than bandwidth constrained;

(b) Time is divided into discrete time slots. This makes it easy to describe how many retransmitted messages are triggered by a delay caused by the overload. The errors introduced by the discrete time slot can be made arbitrarily small by making the interval of a timeslot smaller and smaller. We use t and n to denote time and timeslot respectively;

(c) The SIP RFC [1] does not specify the queuing and scheduling 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. This FIFO queuing model reflects the common practice by most vendors today [11]. Within the same time slot, original request messages enter the tail of the queue prior to retransmitted request messages. Such enqueuing priority has negligible impact if the interval of the time slot is very small. There is no enqueuing difference for the messages arriving at different time slots;

(d) The time to process a response message or a timer timeout is typically much smaller than a request message [1]. We assumed that, within a time slot, the server has enough CPU capacity to process the incoming response messages, thus response messages will not be enqueued as long as they are treated with higher priority such as interrupt. They will not be dropped either when the queue for request messages are overflowed. The service capacity of the overloaded server includes the rate for processing response messages;

(e) In order to focus our analysis on the overloaded server, we assume multiple upstream originating servers and the downstream server of the overloaded P-server have sufficient capacity to process all requests, retransmissions, and response messages immediately without any delay;

(f) 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. Therefore there is no message loss at all servers.

(g) The hop-by-hop Invite-100Trying transaction is the major workload contributor due to its role for call setup and its hop-by-hop retransmission mechanism [1]. 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-100Trying transaction and ignore other end-to-end transactions in this paper.

Fig. 3 depicts the queuing dynamics of an overloaded SIP server in a SIP network (as shown in Fig. 2). The overloaded server receives the original Invite requests with an aggregate rate $\lambda(n)$ at time slot n, where $\lambda(n)$ can be arbitrary. We can obtain the queue size $q(n+1)$ at next time slot $n+1$ based on the information at the current time slot n, i.e.,

$$q(n+1)=[q(n)+\lambda(n)+r(n)-\mu(n)]^{+}. \qquad (1)$$

where $q(n)$ denotes the queue size; $r(n)$ denotes the aggregated retransmitted messages; $\mu(n)$ denotes an arbitrary service process for the request messages, which is equal to the server service capacity minus the service rate for the response messages. $\lambda(n)$ plus $r(n)$ give the total arrival messages at current time slot n. Adding $q(n)$ and deducting $\mu(n)$ would generate a new queue size $q(n+1)$ in the next time slot $n+1$, as described by Eq. (1). We use $[]^+$ to indicate that the queue size at each time slot should be nonnegative.

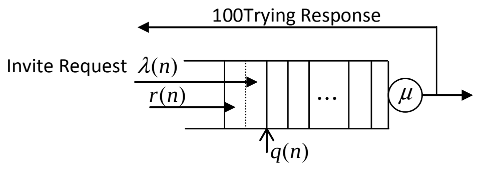


Fig. 3. Queuing dynamics of an overloaded SIP server ($\lambda(n)$ denotes aggregated original message arrivals, $r(n)$ denotes aggregated retransmitted message arrivals from multiple upstream servers, $q(n)$ denotes queue size, $\mu(n)$ denotes service rate).

If the server does not receive the corresponding response message for an original request message at a specific time-out, it would trigger retransmission. There are maximum 6 retransmissions for every original request message [1]. Thus we can obtain the total retransmitted messages $r(n)$ at current time slot n as

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

where $r_j(n)$ denotes the j^{th}–time retransmission for the original request messages arriving at time $n-T_j$, $T_j=(2^j-1)T_1$ and $1\le j\le 6$.

At time $(n-T_j)$, the original request message arrivals were $\lambda(n-T_j)$ and the queue size was $q(n-T_j)$. Since the overloaded server can process $\sum_{k=1}^{T_j}\mu(n-T_j+k)$ messages during the T_j time slots, the remaining messages at current time slot n become $[\lambda(n-T_j)+q(n-T_j)-\sum_{k=1}^{T_j}\mu(n-T_j+k)]^+$, which should be nonnegative. This may include both the original arrival messages at time $(n-T_j)$ and the queued messages right before the time slot $(n-T_j)$. However, only the remaining original arrival messages $\lambda(n-T_j)$ need to be retransmitted at current time n, we use min{} function to get the j^{th}–time retransmitted messages at current time slot n, i.e.,

$$r_j(n)=\min\{[\lambda(n-T_j)+q(n-T_j)-\sum_{k=1}^{T_j}\mu(n-T_j+k)]^+,\ \lambda(n-T_j)\} \tag{3}$$

Eqs. (1) to (3) shows the dynamic behaviour of an overloaded SIP server. Due to its nonlinear characteristic, it may show complex, sometimes chaotic, patterns that bring a potential server collapse.

IV. Stability Condition For SIP Retransmission Mechanism

The retransmission can provide a reliable delivery of SIP messages. However, it also increases the queuing size and enhances the overload. It would be interesting to derive a stability condition that the server can handle overload effectively. The messages accumulated by a transient overload (e.g., a demand burst or a server slowdown) create an initial queue size when the server returns to its normal service state. Such initial queue size may bring a queuing delay long enough for the retransmissions of old remaining original messages in the queue as well as all the new incoming original messages. We would like to investigate whether the SIP server can serve the original messages in the initial queue size and their retransmissions under a low effective CPU utilization.

Without loss of generality, we consider the "Invite-Trying" request-response pair with a deterministic arrival rate λ and a deterministic service rate μ; there are i retransmissions for the new arrival original messages; the initial queue size is $q(0)$.

Theorem 1: If the initial queue size $q(0)$ created by a demand burst can satisfy a sufficient stability condition described by Eq. (4), then the SIP server is stable.

$$q(0)<\ \min\left\{(2^{j+1}-1)\mu T_1,\ \frac{(2^{j+1}+3\times 2^i-i-4)\mu T_1-((i-1)2^i+1)\lambda T_1}{i+1},1\le i\le j\le 6\right\} \tag{4}$$

Proof:

To prevent messages from accumulating unlimitedly in SIP server, the total average incoming rate should be less than the service rate. Assume that there would be i retransmissions for an arbitrary original Invite request message, a conservative condition to maintain stability is

$(i+1)\lambda/\mu\le 1$,

which is equivalent to

$$\mu\ge(i+1)\lambda, \tag{5}$$

i.e.,

$$i\le(\mu-\lambda)/\lambda. \tag{6}$$

To achieve the above sufficient stability condition, we need to guarantee that the original messages from both the initial queue size and the new arrivals are not retransmitted more than j times, where we denote j as $j=\lfloor(\mu-\lambda)/\lambda\rfloor$. Then we update the equivalent stability condition in Eq. (6) as

$$i\le j=\lfloor(\mu-\lambda)/\lambda\rfloor. \tag{7}$$

To avoid $j+1$ retransmissions for the original messages in the initial queue size, we obtain a stability condition for the initial queue size as

$q(0)/\mu<T_{j+1}=(2^{j+1}-1)T_1$,

which is equivalent to

$$q(0)<\mu T_{j+1}. \tag{8}$$

To avoid $(j+1)$ retransmissions for any newly arrival original messages, the queue size in any time should satisfy

$$q(t)<\mu T_{j+1}. \tag{9}$$

Eq. (4) can certainly satisfy Eq. (8). To show that Eq. (4) can satisfy the requirement of Eq. (9), we consider five cases in the following discussion.

1. We first consider the queue sizes at each specified retransmission times $T_i=(2^i-1)T_1$ using Eqs. (1) to (3) as follows,

$q(T_1)=q(0)-(\mu-\lambda)T_1+[q(0)-\mu T_1]^+$,

$q(T_2)=q(T_1)-2(\mu-2\lambda)T_1+[q(0)-\mu T_2]^+$,

$\vdots$

$q(T_i)=q(T_{i-1})-2^{i-1}(\mu-i\lambda)T_1+[q(0)-\mu T_i]^+$, (10)

$\vdots$

$q(T_6)=q(T_5)-32(\mu-6\lambda)T_1+[q(0)-\mu T_6]^+$.

2. We next consider the queue sizes between any two neighbouring retransmission times T_{i-1} and T_i. Eqs. (1), (2) (3), (7) and (10) lead to

$q(t)=q(T_{i-1})-(\mu-i\lambda)(t-T_{i-1})<q(T_{i-1})$, (11)

The inequality in Eq. (11) indicates that the queue size is decreasing continuously with a slope of $\mu-i\lambda$ during the time period. However, at time $t=T_i$, the *ith* retransmission for the remaining $[q(0)-\mu T_i]$ messages from the initial queue size $q(0)$ is triggered, resulting in a sudden increase in the queue size described by (10). Then when $0<t\le T_j$, the condition described by (9) becomes

$q(T_i)<\mu T_{j+1} \qquad 1\le i\le j\le 6.$ (12)

Given the condition of Eq. (8), we assume the worst case with $q(0)-\mu T_i\ge 0$. Using recursive substitution for Eq. (10), we can obtain

$$q(T_i)=(i+1)q(0)-\sum_{k=1}^{i}2^{k-1}(\mu-k\lambda)T_1-\sum_{k=1}^{i}(2^k-1)\mu T_1$$

which can be reorganized as

$$q(T_i)=(i+1)q(0)-\sum_{k=1}^{i}2^{k-1}\mu T_1 +\frac{d}{dx}\sum_{k=1}^{i}\lambda T_1x^k\Bigg|_{x=2}-\sum_{k=1}^{i}2^k\mu T_1+i\mu T_1,$$

or

$$q(T_i)=(i+1)q(0)-\frac{1-2^i}{1-2}\mu T_1 +\frac{d}{dx}\left[\frac{x-x^{i+1}}{1-x}\lambda T_1\right]\Bigg|_{x=2}-\frac{2-2^{i+1}}{1-2}\mu T_1+i\mu T_1.$$

Then we can obtain

$q(T_i)=(i+1)q(0)+((i-1)2^i+1)\lambda T_1-(3\times 2^i-i-3)\mu T_1$. (13)

Combining Eqs. (12) and (13), we can obtain the second condition in Eq.(4) as

$$q(0)<\frac{(2^{j+1}+3\times 2^i-i-4)\mu T_1-((i-1)2^i+1)\lambda T_1}{i+1} \qquad 1\le i\le j\le 6 \tag{14}$$

3. We then consider the case that $T_j<t<T_{j+1}$. From Eqs. (1), (2), (3), (7), (9) and (12), we have

$q(t)=q(T_j)-(\mu-(j+1)\lambda)(t-T_j)\le q(T_j)<\mu T_{j+1}$. (15)

This means the queue size is non-increasing during the time period $T_j<t<T_{j+1}$.

4. Next, we consider the case that $t=T_{j+1}$. Since Eq. (8) indicates $[q(0)-\mu T_{j+1}]^+=0$, from Eqs. (1), (2), (3), (7) and (12), we can obtain

$q(T_{j+1})=q(T_j)-2^j(\mu-(j+1)\lambda)T_1+[q(0)-\mu T_{j+1}]^+\le q(T_j)<\mu T_{j+1}$. (16)

5. Finally, we consider the case that $t>T_{j+1}$. From Eqs. (1), (2), (3), (7) and (16), we have

$q(t)=q(T_{j+1})-(\mu-(j+1)\lambda)(t-T_{j+1})\le q(T_{j+1})<\mu T_{j+1}$. (17)

Combining Eq. (8), (11), (12), (14), (15), (16) and (17), we can reach a sufficient stability condition for the initial queue size described by Eq. (4).□

V. Performance Evaluation and Simulation

Since violating the sufficient stability condition does not always bring the instability to a SIP system, we would like to investigate how tight the sufficient stability bound for the retransmission mechanism is when a SIP overload happens. To achieve this goal, we will evaluate the performance of an overloaded SIP server by performing fluid-based Matlab simulation using the fluid model described by Eqs. (1) to (3), where the time slot is 50ms. In the mean time, in order to validate the accuracy and scalability of fluid-based simulation, we also performed event-driven OPNET simulation in a real SIP network as depicted by Fig. 2. In the OPNET simulator, messages were handled one by one instead of being aggregated over a time slot as in our Matlab simulation. Four originating servers generated original request messages with equal rate, and then sent them to four terminating servers via two P-servers. All the sending servers also maintained a list of all outstanding messages for tracking retransmissions. The mean service rate of P-Server is set to be the same as the mean service rate of the corresponding P-server in MATLAB simulation. For OPNET simulation, messages were enqueued based on first-come-first-in principle. That is, Assumption (c) was unnecessary for OPNET simulation. The default timer for the first retransmission was T_1=0.5s [1].

The retransmission messages triggered by the overload are redundant messages. Therefore, only the CPU consumed by the original messages can be regarded as effective use of resources. We define effective CPU utilization ρ as the ratio between the total mean arrival rate for the original messages and the mean service rate, i.e., $\rho=\lambda/\mu$.

Simulation Setting of Network Configuration Parameters: The mean arrival rate and the service capacity of a SIP server may be different across different real carrier networks. To reduce the overload probability, resource over-provisioning has been employed to maintain a low effective CPU utilization in the real carrier networks [12]. Therefore, for both Matlab and OPNET simulation, we make the CPU utilization low by selecting appropriate mean original message arrival rate of regular demands and mean service rate. For the fluid-based Matlab simulation based on the fluid model, smaller time slot corresponds to more accurate simulation result, but a longer simulation time. In addition, the smallest time slot for arrival rate or service rate should guarantee at least one message per slot, e.g., the smallest time slot for 200 messages/sec is 5ms. Performance comparison between Matlab simulation and OPNET simulation in the following subsection demonstrates that a time slot of 50ms is relatively accurate for the fluid-based simulation.

To verify the sufficient stability condition for the initial queue size, we have considered two typical overload scenarios: (1) The overload was caused by a demand burst, while arrival rate and service rate were deterministic; (2) The overload was caused by a server slowdown, while arrival rate and service rate were Poisson distributed[1].

[1] Currently there is no measurement result for the workload in the real SIP networks. Poisson distributed message arrival rate and service rate are widely adopted by most existing research work (e.g., [13]).

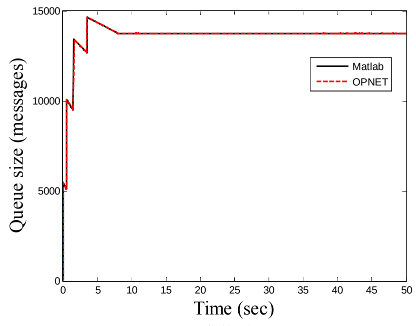
A. Overload Caused by Demand Burst

In this scenario, a demand burst overloaded the server and created an initial queue size at time t=0s, emulating a short surge of user demands; normal original request messages arrived at the overloaded server with a constant rate λ=200 messages/sec, emulating regular user demands. The overloaded server maintained a constant service rate μ=1000 messages/sec. Thus the effective CPU utilization for regular user demands is $\rho=\lambda/\mu$=20%. The simulation period is 50s.

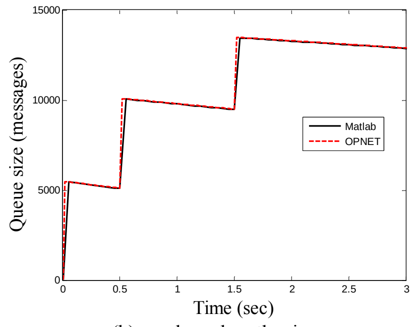
Eq. (7) gives $j=\lfloor(\mu-\lambda)/\lambda\rfloor$=4. Then using Eq. (4), we can obtain the stability condition for the overloaded server as q(0)<min{15500, 8200, 6167, 5700, 6220}=5700 messages. We will consider two sub-scenarios with different initial queue sizes.

1) Sub-scenario (*a*): In this sub-scenario, a demand burst created an initial queue size as q(0)=5500 messages < 5700 messages, obeying the stability condition described by Eq. (4).

Figs. 4 and 5 show the dynamic behaviour of the overloaded SIP server using both Matlab simulation and OPNET simulation. One can observe that the curves obtained by Matlab simulation are very close to the curves obtained by OPNET simulation. The difference for instantaneous retransmission rate shown in Fig. 5 was caused by enqueuing priority within the same time slot (Assumption (c)). The similarity between Matlab simulation result and OPNET simulation result demonstrates that fluid-based simulation is a relatively accurate and cost-effective approach for performance evaluation of a SIP network, while it can simplify a CPU-consuming timer-tracking process by tracking single time slot instead of individual message timer.



(a) full view



(b) enlarged partly view

Fig. 4. Queue size q (messages) versus time for the overloaded server when the initial queue size obeys the stability condition.

Fig. 4(b) shows that the queue size decreased linearly with 800 messages/sec at the beginning.

At time $t=T_1$=0.5s, the overloaded SIP server had processed 500 messages, the 1st-time retransmission for the residual 5000 original messages in the initial queue happened (as shown in Fig. 5). The new 100 original messages arriving between t=0s and $t=T_1$=0.5s joined the queue together with 5000 retransmitted SIP messages, so the queue size became 10,100 messages (as shown in Fig. 4(b)). The new arrival original messages at time t=0s started to trigger the first-time retransmissions (as shown in Fig. 5). Similarly, due to the 2nd-time and 3rd-time retransmissions, the queue size increased dramatically at time $t=T_2$=1.5s and $t=T_3$=3.5s respectively.

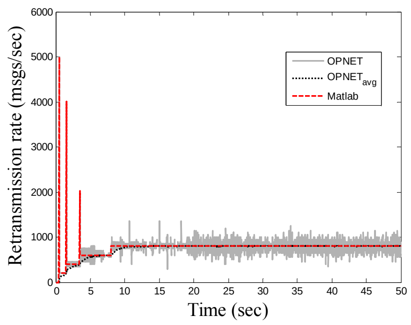


Fig. 5. Retransmission rate r and moving average retransmission rate r_{avg} (messages/sec) versus time for the overloaded server when the initial queue size obeys the stability condition.

At time t=8s, the retransmission rate of new arrival original messages increased from 600 messages/sec to 800 messages/sec (as shown in Fig. 5), thus the total incoming traffic rate of both original messages and retransmitted messages was equal to the service rate μ=1000 messages/sec (or $\rho'=5\lambda/\mu$=1). Between the time t=3.5s and t=8s, 900 new incoming original messages and 2700 incoming retransmitted messages entered the overloaded SIP server, thus the queue size reached and stayed at a steady queue size as 14700+900+2700-4500=13800 messages, well matching our theoretical analysis on the queuing dynamics in Section III.

2) Sub-scenario (*b*): In this sub-scenario, a demand burst created an initial queue size as q(0)=6000 messages > 5700 messages, violating the stability condition described by Eq. (4).

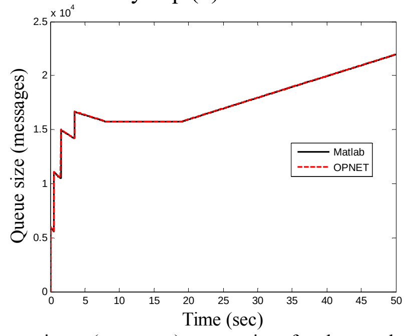


Fig. 6. Queue size q (messages) versus time for the overloaded server when the initial queue size violates the stability condition.

Fig. 6 shows that the queue size decreased linearly except 4 spikes due to the dramatic retransmissions until the time t=19s. At time t=19s, the retransmission rate of new arrival original messages increased from 800 messages/sec to 1000 messages/sec (as shown in Fig. 7). The total incoming traffic rate of both original messages and retransmitted messages was larger than the service rate μ=1000 messages/sec (or $\rho'=6\lambda/\mu=1.2>1$). Therefore, after the time t=19s, the queue size increased linearly and continuously with 200 messages/sec (as shown in Fig. 6), which would bring a SIP server crash eventually.

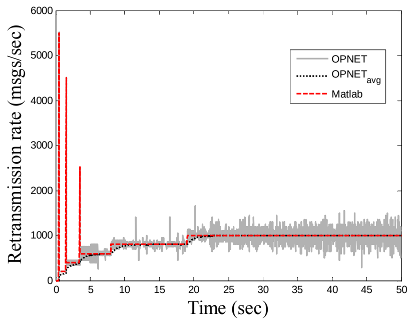


Fig. 7. Retransmission rate r and moving average retransmission rate r_{avg} (messages/sec) versus time for the overloaded server when the initial queue size violates the stability condition.

3) Remarks: In case of the deterministic traffic pattern, slightly different initial queue sizes due to the demand bursts (the difference is less than 10% in the two sub-scenarios) create totally different dynamic behaviour patterns. The slightly smaller initial queue size of 5500 messages allows the server to handle the initial temporary overload effectively, while the slightly larger initial queue size of 6000 messages will result in infinitely increasing queue size, thus bring a SIP server to crash. This indicates that the sufficient stability bound described by Eq. (4) is quite tight when both original message arrival rate of regular demands and service rate are deterministic.

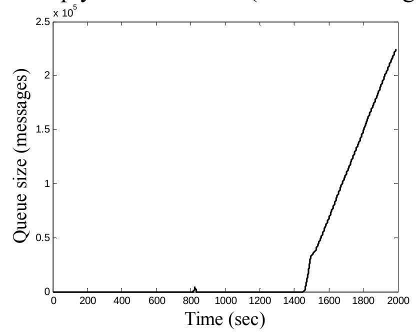
When the original message arrival rate and service rate, are arbitrarily distributed, the operators can apply Theorem 1 to determine the stability condition using the mean values of the original message arrival rate and service rate. A moving average filter can be used to measure the mean values λ and μ. Our simulation result in the next subsection indicates that the sufficient stability bound is also tight for Poisson distributed arrival rate and service rate which has been widely considered as one of the typical traffic patterns in the real carrier networks [13].

B. Overload Caused by Server Slowdown

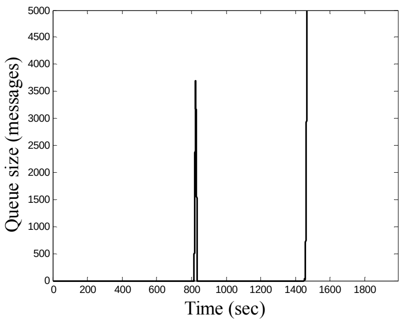
In this application scenario, the overloaded SIP server worked in one of the two states (i.e., normal service state and maintenance state) alternately. During the maintenance period, the overload may happen due to the server slow down. The mean service time at the normal service state was m_1=600sec; the mean service time at the maintenance state was m_0=30sec; all were exponential distributed. The mean service rate at the normal service state was μ_1=1000 messages/sec; the mean service rate at maintenance state was μ_2=180 messages/sec; the mean arrival rate of the SIP messages was λ=200 messages/sec; all were Poisson distributed. The simulation period is 2000s. The overall effective mean utilization was equal to $\rho = \lambda(m_1 + m_0)/(m_1\mu_1 + m_0\mu_0) \approx 0.21$. We will not show the OPNET simulation result because it was very close to the Matlab simulation result as the previous subsection.

Figs. 8 to 11 show the dynamic behaviour of the overloaded SIP server under two different service states.

Between the time t=812s and 823s, SIP server had short period of maintenance, service rate decreased (as shown in Fig. 11). The messages started to accumulate and the queue size increased to reach a peak around 3690 messages at time t=823s (as shown in Fig. 8(b)). When a mean service rate was 180 messages/sec, the queue size larger than 90 messages brought a queuing delay longer than 0.5s, and started to stimulate the retransmission (as shown in Fig. 10). After the server resumed normal service at time t=823s, the initial queue size was less than 5700 messages (the stability condition described by Eq. (4)). The server could process these accumulated messages in time, so the queue size decreased until the buffer was empty at time t≈833s (as shown in Fig. 8(b)).



(a) full view



(b) enlarged partly view

Fig. 8. Queue size q (messages) versus time for the overloaded server which performed normal service and maintenance service alternately.

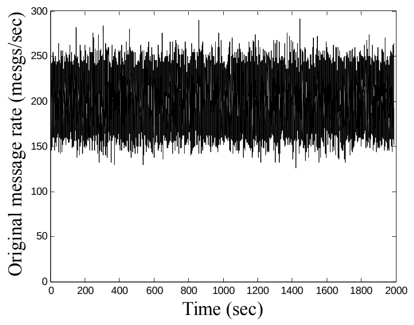


Fig. 9. Original message arrival rate λ (messages/sec) versus time for the overloaded server which performed normal service and maintenance service alternately.

However, maintenance with a relatively long period (or a equivalent large demand burst)) happened at time t=1452s, the queue size increased continuously and triggered more than 5 retransmissions that made the total arrival message arrival rate exceed the normal service rate (as shown in Fig. 11). After the server entered the normal service state at time t=1495s, the initial queue size was larger than 5700 messages. Since the stability condition for the initial queue size was violated, the SIP server cannot handle the overload effectively. The queue size tended to infinity (as shown in Fig. 9(a)), thus eventually crashed the server.

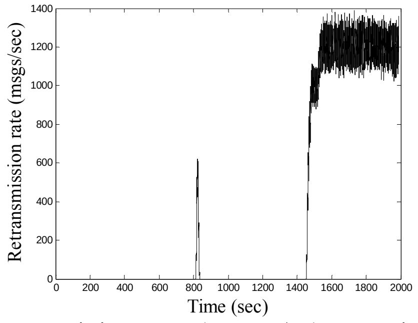


Fig. 10. Retransmission rate r (messages/sec) versus time for the overloaded server which performed normal service and maintenance service alternately.

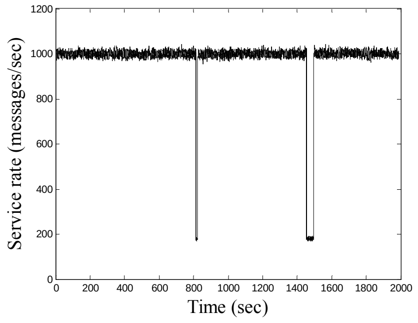


Fig. 11. Service rate μ (messages/sec) versus time for the overloaded server which performed normal service and maintenance service alternately.

In summary, although the effective mean utilization is as low as 20%, if the accumulated messages in the SIP server during the short maintenance period violate the stability condition for the initial queue size, the server cannot mitigate the overload effectively after it resumed its normal service. Goodput collapse persists and the server would crash eventually, well matching our theoretical analysis.

VI. Overload Control Algorithm

The retransmitted messages triggered by the overload delay are redundant and may bring the overload collapse. Therefore, quite different from current existing overload control algorithms which adopt the push-back mechanism, our goal for mitigating the overload is to control the retransmission rate r, thus helping a server to cancel the overload eventually. To achieve our target, we use a retransmission probability p to determine whether to retransmit an original message when its retransmission timer fires or expires. When the total message arrival rate of a downstream server exceeds its message processing capacity, its queue size would increase continuously, which indicates that an overload occurs. A long queue size would make a long queuing delay for the new arrival original messages sent by its upstream servers, thus triggering corresponding redundant retransmissions. Therefore, we propose to control the retransmission probability p based on the average queue size q_{avg} of the overloaded downstream server. Similar to existing push-back overload control solutions, we introduce a minor change to SIP protocol by defining one extra field in every response message to carry the retransmission probability p calculated by the overloaded downstream server. Our simple overload control algorithm is given in Fig. 12.

Setting q_{min} and q_{max}: Since the upstream servers only generate the retransmissions for the original messages whose retransmission timers fire or expire, and a queuing delay less than the 1st retransmission timer T_1 at the overloaded downstream server will not trigger any retransmissions, we suggest minimum and maximum thresholds as

$$q_{1\min}=0.2\mu^*T_1, \qquad (18)$$

$$q_{1\max}=\mu^*T_1, \qquad (19)$$

where μ is the mean service rate of the overloaded downstream server.

```
Overload Control Algorithm

Upstream Server Behaviour
    When each retransmission timer fires or expires
        Retransmit the message with probability $p$

Overloaded Downstream Server Behaviour
    Calculate retransmission probability $p$:
        if $q_{avg} < q_{min}$
            $p \leftarrow 1$
        else
            if $q_{min} \leq q_{avg} \leq q_{max}$
                $p \leftarrow (q_{max}-q_{avg})/(q_{max}-q_{min})$
            else
                $p \leftarrow 0$

Server parameter:
    $q_{avg}$: Average queue size of the overloaded
    downstream server
    $q_{min}$: Minimum threshold for $q_{avg}$
    $q_{max}$: Maximum threshold for $q_{avg}$
```

Fig. 12. Overload control algorithm for controlling retransmission rate.

A. Modeling of Overload Control Algorithm

Since our overload control algorithm mitigates the overload by controlling retransmission probability, we need to create a respective fluid model for the fluid-based simulation.

Since stochastic process of both message arrival rate λ and service rate μ may cause transient fluctuation in the instantaneous queue size q, we use an exponentially weighted moving average filter to obtain the average queue size q_{avg}, i.e.,

$$q_{avg}(n)=(1-w_q)q_{avg}(n-1)+w_q q(n), \qquad (20)$$

where w_q is the filter weight.

According to the overload control algorithm described by Fig. 12, the retransmission probability p generated by the overloaded server can be computed as

$$p(n)=\min\{[(q_{max}-q_{avg})/(q_{max}-q_{min})]^{+},1\} \quad (21)$$

By integrating the retransmission probability p into the theoretical retransmission rate r, we can get the actual retransmission rate as rp. Therefore, we can reuse the fluid model (or Eqs. (1) to (3)) developed for the regular server in Section III by replacing the theoretical retransmission rate with the actual retransmission rate, i.e., we only need to update Eq. (3) as

$$r_j(n)=\min\{[\lambda(n-T_j)+q(n-T_j)-\sum_{k=1}^{T_j}\mu(n-T_j+k)]^{+}, \lambda(n-T_j)\}\,p(n) \quad (22)$$

B. Evaluation of Overload Control Algorithm

In order to validate our overload control algorithm, we simulated a typical overload scenario caused by the server slowdown. We perform fluid-based MATLAB simulation using the fluid model we have derived. We performed our simulations with overload control algorithm and without overload control algorithm separately.

The simulation period is 90s. The mean server rate of the overloaded server was μ=100 messages/sec from time t=0s to t=30s, and μ=1000 messages/sec from time t=30s to t=90s; the mean original message arrival rate was λ=200 messages/sec; all were Poisson distributed. The maximum and minimum thresholds for average queue size were set as q_{max}=500 messages and q_{min}=100 messages respectively. The averaging filter weight w_q was 0.1.

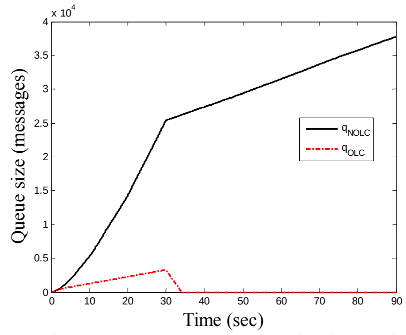


Fig. 13. Queue size q (messages) versus time for the overloaded server when the overload control algorithm was/was not activated.

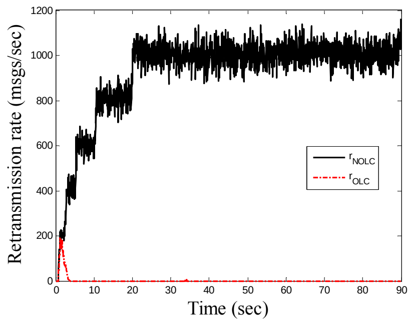


Fig. 14. Retransmission rate r (messages/sec) versus time for the overloaded server when the overload control algorithm was/was not activated.

Figs. 13 and 14 show the dynamic behaviour of the overloaded server. We use "OLC"/"NOLC" to indicate that overload control algorithm "was"/"was not" activated at the upstream server of the overloaded server.

Without overload control algorithm applied, the redundant retransmissions were triggered to increase the queue size of the overloaded server sharply. After the overloaded server resumed its normal service at time t=30s, the mean new original message arrival rate was less than the mean service rate. However, the long initial queue size continued to stimulate redundant retransmissions (see Fig. 14). These retransmissions made the aggregated mean arrival rate of both original messages and retransmitted messages exceed the mean service rate, thus continuing to build up the queue (see Fig. 13). The persisted overload would eventually lead to an overload collapse.

With our overload control algorithm applied, the retransmission rate r was restricted (see Fig. 14). The overload was mitigated and the queue size of the overloaded server increased relatively slowly. After the overloaded server resumed its normal service, it only spent 4s to cancel the overload and the buffer became empty at time t≈34s (see Fig. 13).

VII. Conclusions

We have investigated the SIP retransmission mechanism in case of the overload. We have derived a sufficient stability condition that SIP server can handle the overload effectively under the retransmission mechanism. To prevent the system from crashing, the initial queue size caused by a transient overload should satisfy the stability condition. Such stability condition can help the SIP operator to trigger the overload control algorithm ahead of time to avoid the SIP server collapse.

We have performed simulation using both fluid-based simulation approach and event-driven simulation approach to evaluate the performance of an overloaded SIP server. Our study indicated that the behaviour of the SIP server is highly sensitive to the temporary overload due to the demand burst or the server slow down. The sufficient stability bound for the initial queue size caused by the overload is quite tight. Effective resource utilization as low as 20% cannot prevent an overloaded server from crash, if an initial queue created by a short-term overload (due to a demand burst or a temporary server slowdown) exceeds the sufficient stability bound slightly.

Event-driven simulation, adopted by most existing literature on SIP study, requires a series of retransmission timers to track outstanding messages, thus makes the experiment computationally expensive. As the network size increases to a large scale, the number of timers may build up to consume excessive memory and CPU time, thus crashes the simulator eventually. On the contrary, fluid-based simulation tracks time on a slot-by-slot basis. Events happening within the same time slot will be aggregated and processed together. Individual timers do not need to be tracked anymore. Thus fluid-based approach is much simpler than event-driven approach.

The similarity between fluid-based Matlab simulation result and event-driven OPNET simulation result demonstrate that fluid-based simulation can be a relatively accurate and cost-effective approach for evaluating the performance of a SIP network.

We also proposed a simple overload control algorithm to mitigate the overload by reducing the retransmission rate only, while keeping the original message rate unchanged to avoid blocking the calls unnecessarily. Our solution can help the carriers to maintain their revenue in case of the overload. The simulation has validated the effectiveness of our overload control algorithm.

Appendix A. SIP Retransmission Mechanism Overview

SIP works in the application-layer for media session establishment and tear-down. To briefly describe a basic SIP operation, we only consider originating UA, P-server and terminating UA, as shown in Fig. A1. 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 an "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 is established and the media communication is created between the originating UA and the terminating UA through the SIP session. The "Bye" request is generated to terminate the session thus cancel the communication.

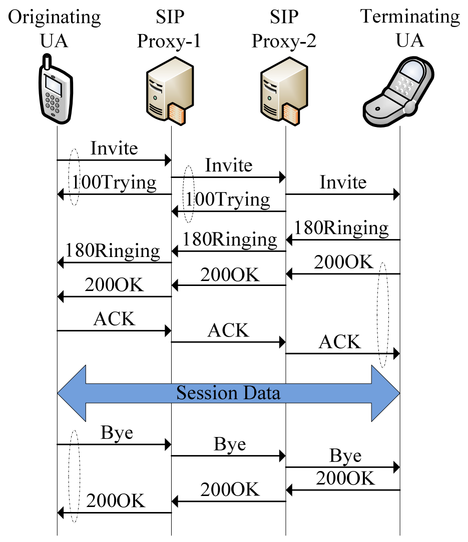


Fig. A1. A typical procedure of session establishment.

SIP has two types of message retransmission: (a) a sender starts the first retransmission of the original message at T_1 seconds, the time interval doubling after every retransmission (exponential backoff), if the corresponding reply message is not received. The last retransmission is sent out at the maximum time interval 64xT_1 seconds. Default value of T_1 is 0.5s, thus there is a maximum of 6 retransmissions. The hop-by-hop "Invite"-"Trying" transaction shown in Fig. A1 follows this rule [1]; (b) a sender starts the first retransmission of the original message at T_1 seconds, the time interval doubling after every retransmission but capping off at T_2 seconds, if the corresponding reply message is not received. The last retransmission is sent out at the maximum time interval 64xT_1 seconds. Default value of T_2 is 4s, thus there is a maximum of 10 retransmissions. The end-to-end "OK"-"ACK" and "Bye"-"OK" transactions shown in Fig. A1 follows this rule [1].

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] J. Rosenberg and H. Schulzrinne, "SIP: Locating SIP Servers," *IETF RFC 3263*, June 2002.
[3] 3GPP TS 24.228 v5.f.0 (2006-10), "Signaling flows for the IP Multimedia call control based on SIP and SDP; Stage 3 (Release 5)," October 2006.
[4] 3GPP TS 24.229 v8.5.1 (2008-09), "IP Multimedia call control protocol based on SIP and SDP; Stage 3 (Release 8)," September 2008.
[5] J. Rosenberg and H. chulzrinne, "Reliability of provisional responses in the Session Initiation Protocol (SIP)," *IETF RFC 3262*, June 2002.
[6] 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, May 2003.
[7] E.M. Nahum, J. Tracey, and C.P. Wright, "Evaluating SIP server performance," in *Proceedings ACM SIGMETRICS*, San Diego, CA, US, 2007, pp. 349–350.
[8] 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.
[9] 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.
[10] E. Noel and C.R. Johnson, "Novel Overload Controls for SIP Networks," in *Proceedings of 21st International Teletraffic Congress*, 2009.
[11] 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.
[12] V. Hilt and I. Widjaja, "Controlling Overload in Networks of SIP Servers," in *Proceedings of IEEE ICNP*, Orlando, Florida, October 2008, pp. 83-93.
[13] C. Shen, H. Schulzrinne, and E. Nahum, "SIP Server Overload Control: Design and Evaluation," in *Proceedings of IPTComm*, Heidelberg, Germany, July 2008.
[14] C. Shen and H. Schulzrinne, "On TCP-based SIP Server Overload Control," in *Proceedings of IPTComm*, Munich, Germany, August 2010.
[15] 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.
[16] A. Abdelal and W. Matragi, "Signal-Based Overload Control for SIP Servers," in *Proceedings of IEEE CCNC*, Las Vegas, NV, January 2010.
[17] "SIP Express Router" http://www.iptel.org/ser/.
[18] 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.

[19] 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* (*IFIP/IEEE IM*), New York, USA, June 2009, pp. 125-128.

[20] V. Gurbani, V. Hilt, and H. Schulzrinne, "Session Initiation Protocol (SIP) Overload Control," *IETF Internet-Draft*, draft-ietf-soc-overload-control-02, February 2011.

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

[22] W. R. Stevens, *TCP/IP Illustrated*, Volume 1, Addison-Wesley, Boston, 1994.

[23] 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.

[24] R. Ando, "Internet phone and video service Skype went down in a global service outage," Reuters News, December 22nd, 2010. http://www.reuters.com/article/idUSTRE6BL47520101222

[25] 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), February 2011.

[26] Y. Liu, F. L. Presti, V. Misra, D. F. Towsley, and Y. Gu, "Scalable fluid models and simulations for large-scale IP networks," ACM *Transactions on Modeling and Computer Simulation*, **14**(3), July 2004, pp. 305–324.

Yang Hong received his Ph.D. degree in electrical engineering from University of Ottawa, Ottawa, Canada. His research interests include SIP overload control, Internet congestion control, modeling and performance evaluation of computer networks, and industrial process control.

Changcheng Huang received his B.Eng. in 1985, and M.Eng. in 1988, both in Electronic Engineering from Tsinghua University, Beijing, China. He received a Ph.D. degree in Electrical Engineering from Carleton University, Ottawa, Canada in 1997. From 1996 to 1998, he worked for Nortel Networks, Ottawa, Canada where he was a systems engineering specialist. He was a systems engineer, and network architect in the Optical Networking Group of Tellabs, Illinois, USA during the period of 1998 to 2000. Since July 2000, he has been with the Department of Systems and Computer Engineering at Carleton University, Ottawa, Canada where he is currently an associate professor.

Dr. Huang won the Canada Foundation for Innovation (CFI) new opportunity award for building an optical network laboratory in 2001. He was an associate editor of IEEE COMMUNICATIONS LETTERS from 2004 to 2006. He is currently a senior member of IEEE.

James Yan is currently an adjunct research professor with the Department of Systems and Computer Engineering, Carleton University, Ottawa, Canada. Dr. Yan received his B.A.Sc., M.A.Sc., and Ph.D. degrees in electrical engineering from the University of British Columbia, Vancouver, Canada. From 1976 to 1996 with Bell-Northern Research (BNR) and from 1996 to 2004 with Nortel, he was a telecommunications systems engineering manager responsible for projects in performance analysis of networks and products, advanced technology research and assessment, planning new network services and architectures, development of network design methods and tools, and new product definition. From 1988 to 1990, he participated in an exchange program with the Canadian Federal Government, where he was project prime for the planning of the evolution of the nationwide federal government telecommunications network. Dr. Yan is a member of IEEE and Professional Engineers Ontario.