Analysis of SIP Retransmission Probability Using a Markov-Modulated Poisson Process Model

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*—As a main signaling protocol for multimedia sessions in the Internet, SIP (Session Initiation Protocol) introduces a retransmission mechanism to maintain the reliability for its real-time transmission. However, retransmission will make the server overload worse. Recent collapse of SIP servers due to emergency-induced call volume indicates that the built-in SIP overload control mechanism cannot prevent the server from overload collapse under heavy load. In this paper, we apply a MMPP (Markov-Modulated Poisson Process) model to analyze the queuing mechanism of SIP server under two typical service states. The MMPP model allows us to investigate the probability of SIP retransmissions. By performing numerous experiments statistically to verify SIP retransmission probability calculated by MMPP model, we find that high retransmission probability caused by short demand surge or reduced server processing capacity during maintenance period may overload and crash a server. We run simulations using time-series directly to observe and analyze the system performance of an overloaded SIP server. This is much faster than event-driven simulation. Numerical results demonstrate that low resource utilization corresponds to low retransmission probability. However, a utilization as low as 20% cannot always guarantee a SIP system stability upon a temporal server slowdown or a short period of demand burst.**

Index Terms—SIP, MMPP, Overload, Resource Utilization, Retransmission Probability

1. Introduction

SIP (Session Initiation Protocol) [1] has been widely deployed for significantly growing session-oriented applications in the Internet, such as Internet telephony, instant messaging and video conference. As a signaling protocol, SIP is responsible for creating, modifying and terminating session 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].

Recent collapse of SIP servers due to emergency-induced call volume or "American Idol" flash crowd [11] highlighted the need for better solutions to manage the performance of SIP servers under overload. RFC 5390 [9] identified the various reasons that may cause server overload in a SIP network. These include but not limited to poor capacity planning, dependency failures, component failures, avalanche restart, flash crowds, denial of service attacks, etc. In general, anything that may trigger a demand burst or a server slowdown can bring server overload and lead to server crash.

SIP introduces a retransmission mechanism to maintain reliability [5]. But if an original SIP message arrives at its destination with an unexpected long delay, the unnecessary retransmissions are triggered, thus bringing more overhead rather than more reliability to the SIP network. Such redundant retransmissions increase the memory and CPU loads for a SIP server, which may cause a system overload and deteriorate the signaling performance [6, 7]. 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 [8].

SIP works independently of the underlying transport layer where TCP (Transmission Control Protocol) and UDP (User Datagram Protocol) are located. SIP RFC 3261 [1] suggests that 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. However, it should be noted that this does not solve the overloading problem mainly because of the following three reasons [8-13]: (1) TCP control mechanism is designed for the congestion caused by limited network bandwidths in the transport layer [13], while the overload in SIP networks is caused by limited CPU processing capacities of SIP servers in the application layer [8-12]. While TCP's "receiver window" can handle the CPU shortage at the TCP layer of the receiver, due to the different functions at different layers, TCP is not aware of CPU overload happening at the SIP layer; (2) For a SIP message which traverses multiple SIP servers, TCP becomes hop-by-hop in the transport layer. When overload happens at one server, TCP congestion control mechanism can only move the overload to its upstream servers rather than to cancel the overload [8]. Therefore hop-by-hop TCP cannot provide overload control for end-to-end SIP messages; (3) TCP implementation is much heavier than UDP in the sender and the receiver. TCP congestion control mechanism that makes TCP heavy is designed for congestion caused by bandwidth exhaustion. It is not effective for server overload control, thus becoming overhead for SIP applications. Furthermore, TCP congestion control mechanism introduces unpredictable delay which is not acceptable for real-time signaling protocols.

Some people may think that this problem can be easily solved by enhancing SIP with an end-to-end congestion control mechanism similar to TCP. Unfortunately this solution does not work because it ignores a major difference between TCP and SIP. TCP is designed for the end-to-end congestion control where each source sends large number of packets to a destination. TCP achieves the congestion control purpose by reducing the sending rate of each source [13]. However, each SIP UA (User Agent) only sends very few signaling messages to its destination for session management. Therefore TCP cannot reduce its sending rate effectively. Similar observation can be found in [8].

Experimental evaluation of SIP servers showed the overload collapse behaviour in [8]. Some solutions have been proposed to prevent SIP overload. For example, a queue-based control scheme was proposed to prevent the overload by rejecting some requests under the heavy load in [7]. Three window-based feedback algorithms were proposed to adjust the message sending rate of the upstream SIP servers based on the queue length [11]. Both centralized and distributed overload control mechanisms for SIP were investigated in [8]. An overload control scheme was discussed in [12]. It has been revealed that retransmission mechanism is a main factor to make the overload condition worse. This motivates us 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 messages to create a long queue. When the server resumes its normal service, the initial long queue size may continue to stimulate the retransmissions and crash the server even the effective resource utilization is low. It would be interesting to obtain the retransmission probability, using which the service providers can decide whether the SIP server would handle overload effectively when they perform configuration management of a SIP network. Modeling and analysis can develop in-depth knowledge to the SIP queuing mechanism and the probability of the retransmission, thus help more researchers find an effective solution to avoid SIP overload caused by the SIP retransmissions. On the other hand, Markov model has been used to investigate the queue size probability of a data handling switch which receives data traffic from the sources with "on" and "off" states, while the data service rate remains constant in [14]. The data switch is quite different from the SIP server which processes signalling traffic with varying service rates at different service states (as described later).

The contributions of this paper are: (1) Applying a MMPP (Markov-Modulated Poisson Process) model to analyze the queuing mechanism of SIP under two typical service states; (2) Investigating the probability of SIP retransmissions using MMPP model; (3) Run simulations using time-series directly to observe and analyze the system performance of SIP server. We will demonstrate that a resource utilization as low as 20% cannot prevent a SIP server overload during a short period of maintenance service and such overload continues to spread even at time when the normal service resumes (i.e., the original message arrival rate of SIP server is only 20% of its processing capacity).

2. SIP Retransmission Mechanism Overview

To briefly describe the basic SIP operation, we only consider originating UA, SIP P-server (Proxy-server) and terminating UA, as shown in Fig. 1. Fig. 1 illustrates a simple network topology for SIP signaling. An originating UA initiates a session by sending a SIP request to P-Server 1 which forwards the request to P-Server 2 allocated to a terminating UA. A respective response is generated to reply to each request for establishing the session between the originating UA and the terminating UA via P-Servers. Each P-Server is responsible for the routing of SIP requests and responses.

Fig. 1 depicts a typical procedure of a session establishment. To set up a call, an originating UA sends an "Invite" request to a terminating UA via two P-servers. The P-server returns a provisional "100(Trying)" response to confirm the receipt of the "Invite" request. The terminating UA returns an "180(Ring)" 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 multimedia communication is created between the originating UA and the terminating UA through the SIP session. The "Bye" request is generated to finish the session thus terminating the communication.

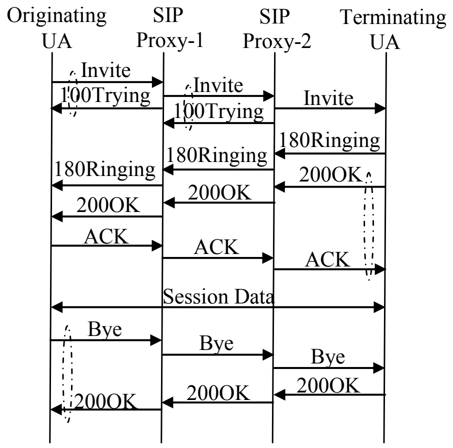


Figure 1: A typical procedure of session establishment

SIP has two types of message retransmission: (a) A message that travels from an originating UA to a terminating UA is confirmed on a hop-by-hop basis. For each hop, the sender starts the first retransmission of the original message at T_1 seconds, and the time interval doubles after every retransmission (exponential back-off), if the corresponding reply message is not received. The last retransmission is sent out at the maximum time interval 64xT_1 seconds. Thus there is a maximum of 6 retransmissions. The default value of T_1 is 0.5s. The hop-by-hop "Invite"-"100(Trying)" transaction shown in Fig. 1 follows this rule [1]. (b) A message that travels from an originating UA to a terminating UA is confirmed on an end-to-end basis. 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. 1 follows this rule [1].

2.1. Queuing Dynamics Of SIP Retransmission Mechanism

Before investigating the retransmission probability using MMPP model, we would like to describe the queuing dynamics of an overloaded SIP server with retransmission mechanism first. Let us consider a scenario with a single overloaded server among a group of SIP servers. We make the following assumptions in accordance with SIP RFC 3261 [1]:

(a) For the round trip delay between two neighbouring SIP servers, the queuing and processing delays are dominant, while transmission and propagation delay is negligible [11]. This assumption is valid because signaling messages typically are constrained by server rather than link bandwidth;

(b) Time is divided into discrete time slots. This allows us to run simulations using time-series directly to observe the dynamic SIP behaviour based on discrete time model. It is easy to see that the errors caused by the discrete model can be made arbitrarily small by making the interval of a timeslot smaller and smaller;

(c) The SIP RFC 3261 [1] does not specify the queuing and scheduling discipline to be deployed by a SIP server. We assume that within a time slot, the original request messages enter the tail of the queue prior to the retransmitted request messages. The impact of this specific priority scheme will be negligible when the interval of the time slot is very small. The server processes the existing messages in the queue according to the FIFO (First-In First-Out) service discipline. The buffer size of the SIP server is infinite;

(d) The upstream and downstream servers for the single overloaded server have infinite capacity to process all original and retransmitted messages immediately without any delay;

(e) Given the proportionate nature and the general similarity of the retransmission mechanisms between the “Invite” and “non-Invite” messages in a typical session [1], we will focus on the hop-by-hop Invite-100(Trying) transaction and ignore other end-to-end transactions. In the mean time, the hop-by-hop Invite-100(Trying) transaction is the major workload contributor due to its role for call setup and its hop-by-hop retransmission mechanism [1]. Such queuing dynamic description can be naturally extended to include end-to-end transactions due to the general similarity between end-to-end retransmission and hop-by-bop retransmission as discussed earlier.

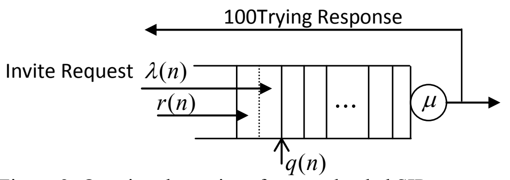


Figure 2: Queuing dynamics of an overloaded SIP server ($\lambda(n)$ denotes original message arrivals, $r(n)$ denotes retransmitted message arrivals, $q(n)$ denotes queue size, $\mu(n)$ denotes service rate)

As shown in Fig. 2, the overloaded server receives the original Invite requests with an aggregate rate $\lambda(n)$ at time slot n. 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.,

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

where $q(n)$ denotes the queue size; $r(n)$ denotes the retransmitted messages; $\mu(n)$ denotes the processed messages. $\lambda(n)+r(n)$ 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 show that the queue size in each time slot should be nonnegative.

According to the SIP retransmission mechanism, we can obtain the total retransmitted messages $r(n)$ at current time slot n as

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

where $r_j(n)$ denotes the j^{th} retransmission for the original request messages arriving at time n-T_j (where $T_j=(2^j-1)T_1$) and there are maximum 6 retransmissions for every original request message (i.e., $1 \le j \le 6$).

At time n-T_j, the original message arrivals were $\lambda(n-T_j)$ and the queue size was $q(n-T_j)$. To decide how many of these messages will be retransmitted, we need to know how many of them are still in queue at time n. The overloaded SIP server can process $\sum_{k=1}^{T_j} \mu(n-T_j+k)$ messages during the past T_j time slots. After T_j time slots, the remaining messages of those queued prior to the time slot n-T_j becomes

$$[\mathrm{q}(n-T_j) - \textstyle\sum_{k=1}^{T_j} \mu(n-T_j+k)]^{+} .$$

The newly arrival original messages $\lambda(n-T_j)$ entered the queue prior to the retransmitted messages $r(n-T_j)$, according to Assumption (c). Without counting $r(n-T_j)$, the remaining messages in the queue becomes

$$[\lambda(n-T_j) + \mathrm{q}(n-T_j) - \textstyle\sum_{k=1}^{T_j} \mu(n-T_j+k)]^{+} .$$

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 time n, so we use minimum function to obtain $r_j(n)$ as

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

Eqs. (1), (2) and (3) present a time series model which gives a complete description of the dynamic behaviour of an overloaded SIP server. The SIP time series model allows us to run fluid-based simulations later on. This is much faster than event-driven simulation which is almost infeasible with a regular simulator due to the large number of timers in SIP protocol.

3. Analysis Of SIP Queuing Mechanism Using MMPP Model

Overload introduces a long queuing delay to SIP server, which would trigger a series of retransmissions to make the overload worse. Understanding the retransmission probability can help the service providers to take actions to prevent excessive retransmissions and overload collapse. We can obtain the retransmission probability by analyzing SIP queuing mechanism using MMPP (Markov-Modulated Poisson Process) model [15]. Since most of SIP servers need to update and synchronize their user databases regularly, a typical application scenario is considered for our queuing analysis, in which the SIP server works in one of the two states alternately. The first state is the normal service with a mean service rate μ_1 and the second state is the maintenance period with a mean service rate μ_0, both assumed to be Poisson distributed. The mean time for the normal service is $1/\beta$ second, while the mean time for the maintenance is $1/\alpha$ second, both assumed to be exponential distributed. Then the system leaves normal service state with a mean rate β, while it leaves the maintenance state with a mean rate α. The SIP traffic arrives in the server with a mean rate λ, and the arrival rate is assumed to be Poisson distributed. Transition between the normal service state and the maintenance state can be governed by an underlying continuous Markov chain. We define p_{ij} as the joint probability that the queue size q is equal to i (the queue state with $i \geq 0$) under the service state j (j=0,1), that is, p_{ij}=P[queue size=i, service state=j], where j=0 and j=1 represent the maintenance state and the normal service state respectively. The doublet (i, j) defines a two-dimensional state space. Assumption (c) indicates that the state space ranges from 0 to *infinity* in the i direction; from 0 to 1 in the j direction. Fig. 3 depicts the resultant two-dimensional state space with transitions between two service states.

The marginal probability matrix P_i, that the queue size is i, is given by

$$P_i = [p_{i0} \quad p_{i1}]. \tag{4}$$

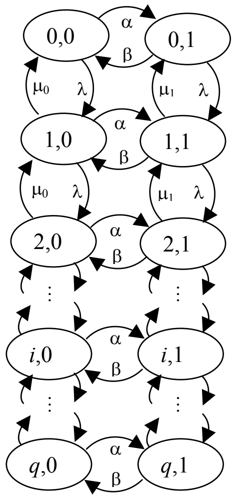


Figure 3: MMPP model for buffering q messages at two multiplexed service states

According to the queuing theory [15], the balance matrix-vector equation for the zero queue size (i=0), can be expressed as,

$$P_0 = P_0 B_0 + P_1 B_1 = P_0 \begin{bmatrix} 1-(\alpha+\lambda) & \alpha \\ \beta & 1-(\beta+\lambda) \end{bmatrix} + P_1 \begin{bmatrix} \mu_0 & 0 \\ 0 & \mu_1 \end{bmatrix} \tag{5}$$

For the queue size $i \geq 1$, the balance matrix-vector equation can be expressed as,

$$\begin{aligned} P_i &= P_{i-1}A_0 + P_i A_1 + P_{i+1}A_2 \\ &= P_{i-1}\begin{bmatrix} \lambda & 0 \\ 0 & \lambda \end{bmatrix} + P_i \begin{bmatrix} 1-(\alpha+\lambda+\mu_0) & \alpha \\ \beta & 1-(\beta+\lambda+\mu_1) \end{bmatrix} + P_{i+1}\begin{bmatrix} \mu_0 & 0 \\ 0 & \mu_1 \end{bmatrix} \end{aligned} \tag{6}$$

Combining Eq. (5) and (6), the complete set of balance matrix-vector equations for the entire two-dimensional chain of the MMPP model can be represented as,

$$P = PC. \tag{7}$$

$$P = [P_0 \quad P_1 \quad \cdots \quad P_i \quad \cdots]. \tag{8}$$

where P consists of the two element vector P_i, and C is the transition probability matrix and can be represented as,

$$C = \begin{bmatrix} B_0 & A_0 & 0 & 0 & \cdots \\ B_1 & A_1 & A_0 & 0 & \cdots \\ 0 & A_2 & A_1 & A_0 & \cdots \\ 0 & 0 & A_2 & A_1 & \cdots \\ 0 & 0 & 0 & A_2 & \cdots \\ \vdots & \vdots & \vdots & \vdots & \cdots \end{bmatrix}. \tag{9}$$

The sum of all the probability is equal to 1, i.e., the sum of all two-element row vector P_i can expressed as

$$\sum_{i=0}^{\infty} P_i = 1. \tag{10}$$

The desired solution for P_i is given by [15]

$$P_{i+1} = P_i R \quad i \geq 0, \tag{11}$$

where R is a 2x2 matrix with $0<r_k<1$ ($1\leq k\leq 4$) [15]. Using recursive substitution [15], we can obtain

$$P_i = P_0 R^i = P_0 \begin{bmatrix} r_1 & r_2 \\ r_3 & r_4 \end{bmatrix}^i \quad i \geq 0. \tag{12}$$

where R is a 2x2 matrix with $0<r_k<1$ ($1\leq k\leq 4$) [15]. The minimal, nonnegative solution for R can be obtained by the following matrix equation [15]

$$R = A_0 + RA_1 + R^2 A_2. \tag{13}$$

<u>Recursive Solution for R</u>

In order to obtain a recursive solution for the matrix R, we rewrite Eq. (13) as

$$R = [A_0 + R^2 A_2][I - A_1]^{-1}. \tag{14}$$

By setting an initial value to R=0, we can reach a minimal and nonnegative solution for R by substituting R into Eq. (14) iteratively [15]. A stable SIP system requires that all the elements of R should be less than 1.

<u>Solution for the probability of zero queue size</u>

Assuming that the matrix R has been found, in order to obtain the probability of all the queue sizes described by Eq. (12), we need to find the probability of zero queue size.

Eqs. (5) and (11) can lead to an eigenvector equation as [15]

$$\hat{P}_0 = [\hat{p}_{00} \quad \hat{p}_{01}] = \hat{P}_0 B_0 + \hat{P}_0 R B_1. \tag{15}$$

By solving Eq. (15), we can obtain the two elements of eigenvector as

$$\hat{p}_{00} = \frac{[(1-r_1)(1-r_4) - r_2 r_3](\beta + r_3\mu_0)}{(1-r_1+r_3)(\alpha+\lambda-r_1\mu_0) + (1-r_4+r_2)(\beta+r_3\mu_0)}, \tag{16}$$

$$\hat{p}_{01} = \frac{[(1-r_1)(1-r_4) - r_2 r_3](\alpha+\lambda-r_1\mu_0)}{(1-r_1+r_3)(\alpha+\lambda-r_1\mu_0) + (1-r_4+r_2)(\beta+r_3\mu_0)}. \tag{17}$$

By carrying out the normalization, we can obtain the probability of the zero queue size as

$$P_0 = [p_{00} \quad p_{01}] = \gamma \hat{P}_0 = [\gamma\hat{p}_{00} \quad \gamma\hat{p}_{01}], \tag{18}$$

where γ is the normalized parameter. Replacing $\hat{P}_i$ by the value $\hat{P}_0 R^i$ using Eq. (12), the sum of all the elements of $\hat{P}_i$ is given as $\hat{P}_0(I-R)^{-1}[1;1] = 1/\gamma$. Since the sum of all the elements of P_i is equal to 1, i.e., $P_0(I-R)^{-1}[1;1]=1$, we can obtain the normalized parameter γ as

$$\gamma = 1/(\hat{P}_0(I-R)^{-1}[1;1]) . \tag{19}$$

Assumption (a) and RFC3261 [1] indicate that a queuing delay $q/\mu \geq T_1$ (i.e., $q \geq \mu T_1$) can trigger the retransmissions for newly arrival original request messages. Once the retransmissions are triggered, the mean arrival rate λ of SIP messages (including original messages and retransmitted messages) would increase to overload the server. Such overload would increase the retransmission probability in return, and may stimulate up to six retransmissions and collapse the server eventually. Therefore it is useful to investigate the probability that the retransmissions would happen.

When the queue size q of a SIP server starts to exceed μT_1, the *1st* retransmission for the newly arrival original messages is triggered after a determined retransmission timer T_1. Prior to the *1st* retransmission, the mean arrival rate λ of the request messages remained little changed. Therefore, in the normal service state, the probability of $q_{i1} \geq \mu_1 T_1$ in a single time slot becomes

$$\begin{aligned} \mathrm{P}(\mathrm{q}_{\mathrm{i}1} \geq 0.5\mu_1) &= \mathrm{P}(\mathrm{i} \geq 0, j=1) - \mathrm{P}(\mathrm{i} < 0.5\mu_1, j=1) \\ &= \sum_{i=0}^{\infty} p_{i1} - \sum_{i=0}^{0.5\mu_1 - 1} p_{i1} \\ &= p_0(I-R)^{-1}[0 \quad 1]^T - p_0(I - R^{0.5\mu_1})(I-R)^{-1}[0 \quad 1]^T \\ &= p_0 R^{0.5\mu_1}(I-R)^{-1}[0 \quad 1]^T \end{aligned} \cdot \tag{20}$$

Similarly in the maintenance state, the probability of $q_{i0} \geq 0.5\mu_0$ without retransmissions in a single time slot becomes

$$\begin{aligned} \mathrm{P}(q_{i0} \geq 0.5\mu_0) &= \mathrm{P}(i \geq 0, j=0) - \mathrm{P}(\mathrm{i} < 0.5\mu_0, j=0) \\ &= \sum_{i=0}^{\infty} p_{i0} - \sum_{i=0}^{0.5\mu_0 - 1} p_{i0} \\ &= p_0(I-R)^{-1}[1 \quad 0]^T - p_0(I - R^{0.5\mu_0})(I-R)^{-1}[1 \quad 0]^T \\ &= p_0 R^{0.5\mu_0}(I-R)^{-1}[1 \quad 0]^T \end{aligned} \cdot \tag{21}$$

Then the total probability which can trigger the retransmissions becomes the sum of $P(q_{i1} \geq 0.5\mu_1)$ and $P(q_{i0} \geq 0.5\mu_0)$, i.e.,

$$\mathrm{P_{rt}} = \mathrm{P}(q_{i1} \geq 0.5\mu_1) + \mathrm{P}(q_{i0} \geq 0.5\mu_0) . \tag{22}$$

Once the *1st* retransmission is triggered, the retransmitted messages come with the original messages to increase the mean arrival rate λ significantly and enhance the server overload. If the server cannot handle with the overload effectively, maximum six retransmissions would be triggered and may crash the server eventually.

4. Performance Evaluation And Simulation

In order to investigate the retransmission probability effectively, we perform the simulations in the following four scenarios.

4.1. Scenario A: Validity of time-series simulation

We use Eqs. (1) to (3) to conduct time-series simulation. This will significantly reduce the simulation time comparing to the traditional event-driven simulation where a large number of timers for retransmissions need to be tracked. In order to investigate the retransmission probability, we need to perform thousands of simulation replications. For the event-driven simulation, each message transmission corresponds to an event. When the overload happens, the messages are built up, and the server needs to increase the number of timers to maintain the message retransmissions. The timers used to track millions of events may exceed the CPU capacity extremely, thus cause the server crash and terminate the simulations unexpectedly. Therefore, event-driven simulation approach employed by [8] is not suitable for numerous simulation replications.

In order to verify the validity of time-series simulation, we consider a scenario with reference to [8]: A demand burst of 6000 messages arrive at a SIP server and create an initial queue size at time t=0s. In the mean time, the server has a constant original message arrival rate λ=200 messages/sec and a constant service rate μ=1000 messages/sec, thus the effective resource utilization $\rho=\lambda/\mu$=0.2. The default timer for the first retransmission is T_1=0.5s [1]. Each timeslot is 10ms.

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

At time $t=T_1$=0.5s, the SIP server had processed 500 messages, the first retransmission for the residual 5500 original messages in the initial queue happened (as shown in Fig. 5(a)). The new 100 original messages arriving between t=0s and $t=T_1$=0.5s joined the queue together with 5500 retransmitted SIP messages, so the queue size became 11,100 messages. The new arrival original messages at time t=0s started to trigger the first retransmissions (as shown in Fig. 5(b)).

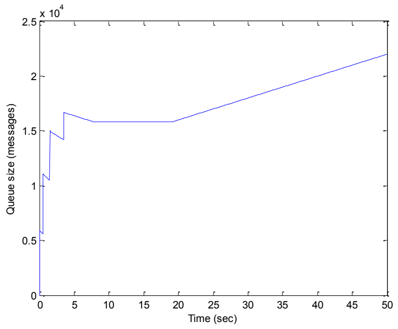
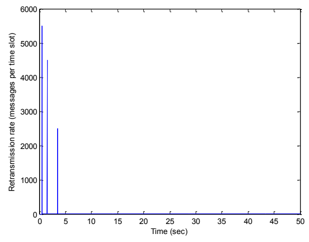
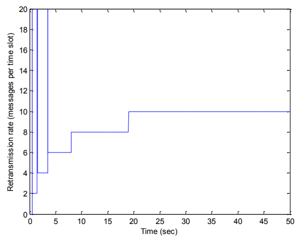


Figure 4: Queue size (messages) versus time



(a) full view



(b) enlarged view

Figure 5: Retransmission rate (messages per time slot) versus time

At time $t=T_2$=1.5s, the SIP server had processed another 1000 messages. The second retransmission for the residual 4500 original messages in the initial queue happened (as shown in Fig. 5(a)). The new arrival original messages at time t=0s started to trigger the second retransmissions, while the new arrival original messages at time t=0.5s had started to trigger the first retransmissions. The retransmission rate of new arrival original messages increased from 200 messages/sec to 400 messages/sec (as shown in Fig. 5(b)). The queue size jumped to 15,000 messages.

At time $t=T_3$=3.5s, the SIP server had processed another 2000 messages. The third retransmission for the residual 2500 original messages happened (as shown in Fig. 5(a)). The retransmission rate of new arrival original messages increased from 400 messages/sec to 600 messages/sec (as shown in Fig. 5(b)). The queue size jumped to 16,700 messages and then decreased linearly with 200 messages/sec until the queue reached a steady value of 15,800 messages at time t=8s (as shown in Fig. 4).

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(b)), thus the total incoming traffic rate of both original messages and retransmitted messages was equal to the service rate μ=1000 messages/sec (or $\rho_1^{'}=5\lambda/\mu=1$). Between the time t=3.5s and t=8s, 900 new incoming original messages and 2700 incoming retransmitted messages entered the SIP server, thus the queue size reached and stayed at a steady queue size as 16700+900+2700-4500=15800 messages, well match the theoretical analysis.

However, at time t=19s, the retransmission rate of new arrival original messages increased from 800 messages/sec to 1000 messages/sec. The total incoming traffic rate of both original messages and retransmitted messages was larger than the service rate μ=1000 messages/sec (or $\rho_2^{'}=6\lambda/\mu=1.2>1$). Therefore, after the time t=19s, the queue size increased linearly and continuously with 200 messages/sec, which would bring a SIP server crash eventually (as shown in Fig. 4). Our numerical result of time-series simulation corresponds to the event-driven simulation results in [8]. However our simulation is much faster than the event-driven simulation in [8] because our approach doesn't need to track the timers for individual messages. This is especially useful when large number of replications need to be to estimate the probability of rare events as will be seen in the next subsection.

4.2. Scenario B: Validity of Retransmission Probability for MMPP Model

In order to investigate the retransmission probability using MMPP model (as described by Eqs. (4)-(22), we consider a scenario: the mean service rate at the normal service state is μ_1=1000 messages/sec; the mean service rate at maintenance state is μ_0=200 messages/sec; the mean arrival rate of the SIP original messages is λ=199 messages/sec; the mean time at the normal service state $1/\beta$; the mean time at the maintenance state $1/\alpha$; all are assumed to be exponential distributed. Each time slot is 10 ms; T_1 is 500ms. We have perform SIP parameter tuning on numerous simulation scenarios. Due to the page limit, we only consider three sub-scenarios with different mean time of maintenance and normal service in this paper: (I) $1/\beta$=5sec, $1/\alpha$=50s and the overall effective mean utilization is equal to $\rho = \lambda(\alpha+\beta)/(\alpha\mu_1+\beta\mu_0) \approx 0.22$; (II) $1/\beta$=5sec, $1/\alpha$=500sec and $\rho\approx 0.2$; (III) $1/\beta$=50sec, $1/\alpha$=5000s and $\rho\approx 0.2$.

Probability Calculation

Using recursive solution based on Eq. (14), we can obtain R=[0.9663 0.0057; 2.4x10^{-5} 0.199]. (23)

From Eqs. (19) to (21), we can obtain p_{00}=0.003 and p_{01}=0.7277. Prior to the 1st retransmission, from Eqs. (21) and (22), we can obtain

$$P(q_{i1} \ge 0.5\mu_1) = P(i \ge 500, j=1) = 2.5\text{x}10^{-11}, \tag{24}$$

$$P(q_{i0} \ge 0.5\mu_0) = P(i \ge 100, j=0) = 0.003, \tag{25}$$

$$P_{rt} = P(q_{i1} \ge 0.5\mu_1) + P(q_{i0} \ge 0.5\mu_0) \approx 0.003. \tag{26}$$

The theoretical probability shows that the retransmission probability of the normal service state with 20% resource utilization is much less than that of the maintenance state with 99.5% resource utilization. This means that low resource utilization corresponds to low retransmission probability.

Similarly using recursive solution, we can obtain the retransmission probability for sub-scenario II as $P(q_{i1}\ge 0.5\mu_1)$ =2.7x10^{-12}, $P(q_{i0}\ge 0.5\mu_0)$=3.2x10^{-4} and $Prt\approx$3.2x10^{-4}, the retransmission probability for sub-scenario III as $P(q_{i1}\ge 0.5\mu_1)$ = 3.2x10^{-8}, $P(q_{i0}\ge 0.5\mu_0)$= 2.7x10^{-3} and $Prt\approx$2.7x10^{-3}.

Based on the theoretical probability obtained by MMPP model, we select the simulation parameters for the three sub-scenarios: (I) 10,000 replications and simulation time for each replication is 600s; (II) 100,000 replications and simulation time for each replication is 3000s; (III) 100,000 replications and simulation time for each replication is 10,000s.

The probability calculated by our MMPP model does not take the retransmission into account. Therefore, in order to make the verification of our MMPP model accurate, we turn off the retransmission mechanism during the simulation in this scenario. We define an overload event as the event which can trigger the retransmissions at the end of each simulation replication. We use "1" and "0" to indicate the overload event and the under-load event.

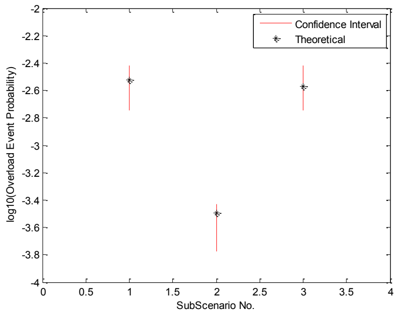


Figure 6: Overload event probability versus sub-scenario No.

For Sub-Scenario I, 28 overload events were recorded among 10,000 replications. The sample mean was

$\overline{X} = 2.8*10^{-3}$, and the standard error was $\sigma/\sqrt{N} = 5.3*10^{-4}$, thus the 95% confidence interval became $\overline{X} \pm 1.96*\sigma/\sqrt{N}$, i.e., $3.8*10^{-3}$ and $1.8*10^{-3}$. Then one can see that the theoretical retransmission probability calculated by Eq. (26) was located within the confidence interval, i.e., $3.8*10^{-3}<P_{rt}=3*10^{-3}<1.8*10^{-3}$. In the mean time, we also find that all 28 overload event happened during the maintenance period, well match our theoretical analysis described by Eqs. (24) to (26).

Fig. 6 shows the statistical graph for three simulation sub-scenarios. All the probabilities *p* have been scaled using log10(*p*). The probability of the overload event and its 95% confidence interval are depicted. The theoretical probability calculated by our MMPP model is also depicted. The theoretical probabilities of all three sub-scenarios are located within the confidence interval of our replications. This demonstrated that the retransmission probability calculated by MMPP model is correct according to the statistics [16].

4.3. *Scenario C: Server Crash due to Retransmission*

Retransmission introduces the overload during the maintenance period, but the server can cancel the overload most of time after it resumes its normal service. If the total arrival rate of the original and retransmitted messages exceeds the normal service rate, the queue size will approach infinity to crash the server eventually.

To avoid the messages to accumulate unlimitedly in a SIP server, the total average incoming rate should be less than the normal service rate μ_1. Assume that there are *i* retransmissions, a conservative condition to avoid overload collapse is:

$$(i+1)\lambda / \mu_1 = (i+1)\rho \le 1,$$

which is equivalent to

$$\mu_1 \ge (i+1)\lambda, \tag{27}$$

or

$$i \le j = \lfloor (1-\rho)/\rho \rfloor. \tag{28}$$

To achieve this, we need to guarantee that all the original messages are not retransmitted more than *j* times.

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

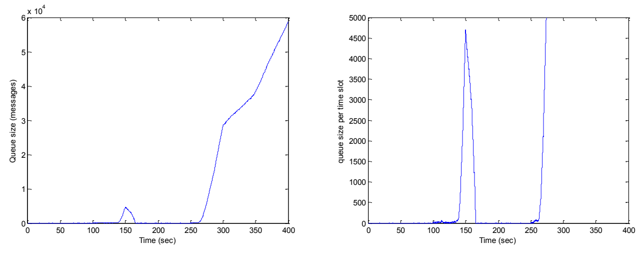
$$q(t)/\mu_1 < T_{j+1} = (2^{j+1}-1)T_1,$$

or

$$q(t) < \mu_1 T_{j+1}. \tag{29}$$

If the queue size accumulated during the maintenance period is less than the conservative stability bound described by Eq. (29), the server can cancel the overload after it resumes its normal service. Therefore the server crash probability should be less than the theoretical retransmission probability calculated by MMPP model. The retransmission probability can therefore be used as an upper bound.

We select two replications of sub-scenario III in Section 4.2, at which the retransmission was triggered and the overload event happened. We observe the transient performance of the server overload in details. If the total mean arrival rate of original retransmitted messages is larger than the mean service rate during the normal service period, the queue size would increase continuously and approach to infinity and the server will crash eventually. If the server can cancel the overload after resuming its normal service, the buffer would remain almost empty as the time evolves. Since the normal service period was too long, to present our simulation result more concisely, we only show the server behaviour during a part of the normal service period.



(a) full view (b) enlarged view

Figure 7: Queue size (messages) versus time

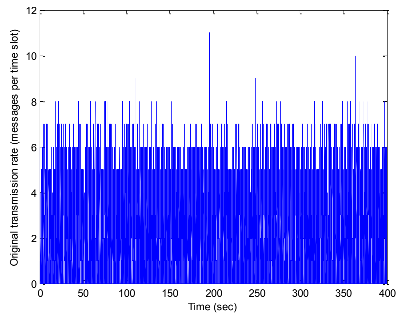


Figure 8: Original transmission rate (messages per time slot) versus time

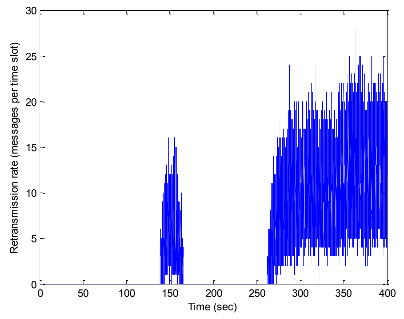


Figure 9: Retransmission rate (messages per time slot) versus time

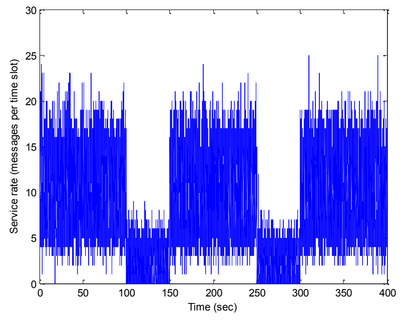


Figure 10: Service rate (messages per time slot) versus time

Figs. 7 to 10 show the dynamic performance of an overloaded SIP server. Between the time *t*=0s and 100s, the server performed its normal service, and the buffer was almost empty. At time *t*=100s, the server started its routine maintenance, the mean service rate decreased to 200 messages/sec (as shown in Fig. 10), thus the queue size larger than 100 messages would bring a queuing delay longer than 0.5s thus stimulate the retransmissions. The messages started to accumulate and the queue size increased to reach a peak around 4699 messages at time *t*=150s (as shown in Fig. 7(b)).In the mean time, maximum 3 retransmissions were triggered (as shown in Fig. 9). After the server resumed normal service at time *t*=150s, the server can process these accumulated messages with a mean rate of 1000 messages/sec, so the queue size decreased until the buffer was empty at time *t*=166s (as shown in Fig. 7(b)). The server cancelled the overload effectively. The server maintained almost empty buffer for about 5000s. As discussed, only 100s of normal service period was shown, while around 4900s normal service period was omitted. At time *t*=250s, the server started its maintenance service again. The queue size increased continuously and triggered maximum 5 retransmissions that made the total arrival message arrival rate exceeded the normal service rate (as shown in Fig. 9). After the server entered the normal service state at time *t*=300s, the initial queue size is larger than 28,600 messages. The SIP server cannot handle the overload effectively. The queue size tended to infinity (as shown in Fig. 7(a)), thus eventually crashes the server.

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 cannot be processed effectively in the normal service period, the server cannot avoid the overload and crash. Goodput collapse persists even after the server resumes its normal service and increases its capacity a lot.

5. Conclusions

We have investigated the SIP retransmission mechanism which may cause server crash upon SIP overload. We have set up an MMPP model to describe the SIP queuing mechanism and then calculate the probability of SIP retransmission. We have performed thousands of simulation replications to verify the retransmission probability obtained by our MMPP model. The event-driven simulation requires large number of timers to track outstanding messages. When overload happens, the messages are built up. This may drive the number of timers to an extreme value which takes so much memory that it crashes the simulator eventually and terminate the simulation unexpectedly. Therefore, event-driven simulation approach is not suitable for the scenarios where a large number of replications are required or where the queue sizes can be very large. To solve this problem, we have run simulation using time-series approach which does not need to track timers at all and therefore is very scalable. Our study indicated that a short term queue build-up may cause the server to crash. We discovered that a large queue size introduced by a demand burst or a temporal server slowdown can overload and crash a SIP server with effective resource utilization as low as 20%. The retransmission probability calculated by our MMPP model is the upper bound of the server crash probability, which can help the service providers to make proper capacity planning and the maintenance scheduling.

In our future work, we will make sensitivity analysis on parameter tuning and discuss the consequences on the server stability. We will consider limited memory modeling in practical systems and take into account potential effects in derived traffic attenuation.

Acknowledgment

We appreciate the financial support from the NSERC grant #CRDPJ 354729-07 and the OCE grant #CA-ST-150764-8. This work is also supported in part by Nortel Networks.

References

[1] J. Rosenberg et al., “SIP: Session Initiation Protocol,” *RFC 3261*, IETF, June 2002.

[2] J. Rosenberg and H. Schulzrinne, “SIP: Locating SIP Servers,” *RFC 3263*, IETF, 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. Schulzrinne, “Reliability of provisional responses in the Session Initiation Protocol (SIP),” *RFC 3262*, IETF, June 2002.

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

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

[8] V. Hilt and I. Widjaja, “Controlling Overload in Networks of SIP Servers,” *Proceedings of 16th IEEE International Conference on Network Protocols* (*IEEE ICNP*), Orlando, Florida, pp. 83-93, October 2008.

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

[10] V. Hilt, I. Widjaja, and H. Schulzrinne, “Session Initiation Protocol (SIP) Overload Control,” IETF, Internet-Draft, draft-hilt-sipping-overload-05, July 2008.

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

[12] 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), pp.173-182, 2004.

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

[14] D. Anick, D. Mitra, and M. M. Sondhi, “Stochastic theory of a data-handling system with multiple sources,” Bell System Technical Journal, 61(8), pp. 1871-1894, 1982.

[15] M. Schwartz, *Broadband Integrated Networks*, Prentice-Hall, 1996.

[16] G.R.Grimmett, and D.R. Stirzaker, *Probability and Random Processes*, 2nd Edition. Clarendon Press, Oxford, 1992.