

IMS Presence Server: Traffic Analysis & Performance Modelling

C. Chi

Bell Laboratories,
Alcatel-Lucent
chic@alcatel-lucent.com

R.Hao

Bell Laboratories,
Alcatel-Lucent
rbhao@alcatel-lucent.com

D.Wang

Bell Laboratories,
Alcatel-Lucent
wangd@alcatel-lucent.com

Z. Cao

Institute of Information Science
Beijing Jiaotong University
caozhenzhen1983@gmail.com

Abstract—Presence is a service that allows a user to be informed about the reachability, availability, and willingness to communicate of another user. Presence service has become a key enabler for many popular applications such as instant messaging and push-to-talk. However a recent study shows that presence service can account for 50% or more of the total signalling traffic the IMS core network handles[1]. This is quite a burden for a real IMS network and need to be tackled. In this paper, a thorough analysis of the traffic load distribution reveals that NOTIFY messages account for the largest portion of the traffic load on a presence server. We propose a mathematical model of a queueing system with batch arrival and controlled vacation to describe the processing of NOTIFY messages inside a presence server. The model is then used to calculate the optimal value of the timer that controls the NOTIFY queue, which is an essential parameter to adjust in network sizing upon deploying a presence server. An approximation of the model is also provided so that the performance of a presence server can be tuned in an online manner based on real traffic statistics.

I. INTRODUCTION

Presence is a service that allows a user to be informed about the reachability, availability, and willingness of communication of another user. The presence service is able to indicate whether other users are online or not and if they are online, whether they are idle or busy. Additionally the presence service allows users to give details of their communication means and capabilities. Presence service has become a key enabler for many popular applications such as push-to-talk (PTT) and instant messaging (IM) which have facilitated communications among communities of interest, such as friends, colleagues, and families [1]. With its popularization and being beneficial to operators, 3G IP Multimedia Subsystem (IMS) already has presence service well supported in its architecture.

A presence service includes four fundamental entities [2][3][4]: a principal, a presentity, a watcher, and a presence server (PS), which may exist independently or as part of application servers (e.g., IM or PTT). A principal refers to a user supported by a presence service and the owner of presentity; presentity information represents the principal's ability and willingness to communicate and rules on how this information can be accessed; a watcher is an entity that subscribes to or that requests information from a presentity; and a presence server is a network entity that is responsible for managing presence information. Fig.1 presents the three basic SIP methods surrounding the presence service [5][6]: SUB-

SCRIBE, PUBLISH, and NOTIFY. In the first event, a watcher subscribes to a particular presentity via a SUBSCRIBE method (SIP message). The PS internally verifies whether this watcher is authorized to subscribe to this particular presentity. If so, it acknowledges with a 200 OK method and sends the current state of the presentity to the watcher via a NOTIFY method [7]. When a presentity's state changes, it uses the PUBLISH method to advise the PS of the change. The PS as a result notifies all subscribed watchers of the presentity's state change. There are also other mechanisms for obtaining presence information such as via the home subscribe server (HSS), however only these three basic methods are considered in the following discussion.

From the standard call flow of presence service as shown in Fig.1, it is concluded that presence service can generate high volume of signalling traffic even when there is no application bearer traffic (e.g.,IM,PTT) [1], since one presentity's state change will in general result in multiple notification messages to watchers. A traffic model for presence service is proposed in [1] and it shows that the presence service related load on Call Session Control Function (CSCF) could be as high as 50% or more of the total signaling load it handles. Simulation study on the performance of a SIP based presence and instant messaging service for UMTS indicates that traffic load has great impact on IMS network performance and it is necessary to control signaling traffic load via various techniques [15]. An admission control mechanism is proposed to control the watcher's subscription time such that traffic load can be reduced [9]. Many proposals are also given in standard organization to reduce the high volume of traffic generated by presence services[9]. For example, the concept of resource lists is introduced in [10] to reduce the number of SUBSCRIBE requests from watchers; partial notification [11] is indicated through SUBSCRIBE request such that the frequency of notification can be reduced; event-throttling mechanism allows a subscriber to an event package to indicate the minimum period of time between two consecutive notifications with the loss of precise of the subscription state information [12]; compression of SIP message is another technique to reduce the data sent in networks [13].

While reducing traffic load is an important way to improve quality of a presence service, behaviors of components in the service system have much more impact. End-to-end delay

measurement for IM relay nodes has been studied in [14] and an optimization model is proposed to minimize delay with respect to the relay node throughput, utilization and buffer size. Though end-to-end delay analysis of presence service is desirable, it will be much more helpful for operators or PS developers to know the system behavior in face of heavy traffic and what is its performance bottlenecks.

With products providing presence services being developed and deployed, the operators need to understand the traffic load distribution and its impact on the product's performance, need to study issues related with network sizing before deploying the service and what is the bottleneck of the service. Unfortunately, there is no comprehensive study in this area. Though the total number of SIP messages handled by a PS is estimated in [1], the relative impact of each type of messages involved in a presence service and the processing overhead which varies for different SIP messages are not accounted for yet.

Network sizing is essential to ensure that adequate bandwidth and resources are available to carry and process mission-critical traffic. During the network sizing process of deploying Alcatel-Lucent's PS for an operator, we did some analysis on the impact of traffic load on a PS by different SIP messages. Based on our study, we conclude that the largest portion of the traffic load to the PS is resulted from NOTIFY messages, which are used to notify the watchers of an presentity the presentity's status change. We thus propose a mathematical model to describe the processing of NOTIFY messages inside the presence server, and then verify that the performance of a PS related with NOTIFY message can be evaluated using this model. This model can also be applied to optimize the timer value that controls the NOTIFY message process. Both numeric and simulation result is given in this paper.

This paper is organized as follows: section II gives the result on traffic load analysis of a PS. In section III, the queueing mechanism of NOTIFY messages in a PS is studied. Applying the analysis result in optimizing NOTIFY queue control parameters is also given in this part. Section IV gives a simplified approximation model for the processing of NOTIFY messages. The paper concludes in Section V.

II. PS TRAFFIC LOAD ANALYSIS

With a presence service reflects the status update of users, traffic load to a PS is highly related with the behavior patterns of users. The login/logoff interval, online behavior, and the number of contacts all have impact on the traffic load to a PS. Each SIP request message to a PS can invoke creating a transaction and the number of transactions processed or maintained by a SIP server is an important parameter that determines the PS's capability [16]. The process overhead needed by a PS to process transactions are different. For example, when a user logs in, an initial PUBLISH message is sent to a PS, which will result in some authentication process and the creation of a new SIP transaction to maintain some information; when the user updates his state and sends the modifying PUBLISH message, the PS only needs to retrieve the information stored during processing of previous

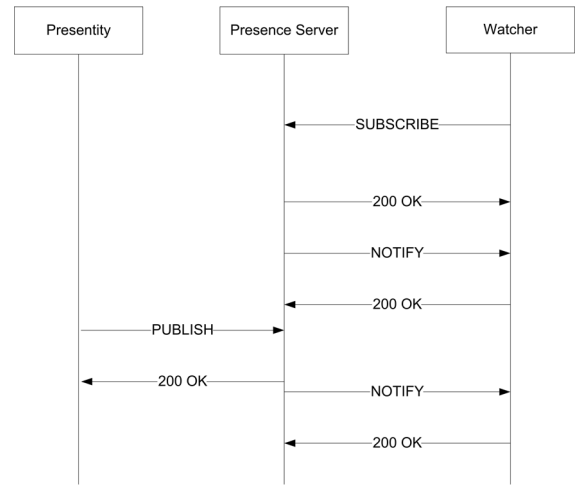


Fig. 1. SIP Presence Call Flow

transaction and much less overhead is needed. In the following analysis, SIP transactions are divided into several types which need different process overhead and the ratio of each type of transactions is studied.

User actions invoke SIP messages as follows:

- User Login and Logout:

A user's login generates an initial PUBLISH message to the PS, after that refresh PUBLISH messages are generated periodically and finally a terminating PUBLISH message is sent to the PS upon the user's logout. With the user's login/out rate being r_{login} and r_{logout} , the rate of initial PUBLISH resulted from the login process is $r_{initial_pub} = r_{login}$, the periodical refresh PUBLISH message rate is denoted as $r_{refresh_pub}$, and the rate of the terminating PUBLISH message resulted from the user's logout is $r_{terminal_pub} = r_{logout}$. $r_{refresh_pub}$ is determined by the user's software setting and it should be no less than once per hour per SIP protocol specification. Each initial PUBLISH and terminating PUBLISH results in a NOTIFY message being sent to all the online watchers who has subscribed to this user's presentity. Assume each user's number of watchers is $n_{watchers}$, the online watchers are given by $n_{online_watcher} = n_{watcher} * p_{online}$. p_{online} is the probability of a user being online.

- Presence Subscription:

Subscription of a user's presentity results in an initial SUBSCRIBE message being sent to the PS, whose rate is denoted as $r_{initial_sub}$; after the initial SUBSCRIBE message is sent, refresh SUBSCRIBE messages will be sent to the PS at a rate of $r_{refresh_sub}$; a terminating SUBSCRIBE message follows at last with rate $r_{terminal_sub}$ to unsubscribe the other user's presentity. $r_{initial_sub}$ depends on the number of contact lists per subscriber, denoted as n_{list} . And

the average number of contacts per list is denoted as $n_{contact}$.

- Presence Status Updates:

The change of a user's status results in a modifying PUBLISH message, assume its rate is r_{modify_pub} ; each modifying PUBLISH message triggers $n_{online_watcher}$ NOTIFY messages.

From above description, traffic to a PS is divided into eight types: initial publish, refresh publish, modify publish, terminal publish, initial subscribe, refresh subscribe, terminal subscribe, and notify. Transaction generation rate of each type is denoted as $r_i, i = 1..8$. Transaction ratio of each type of traffic is given by:

$$s_i = \frac{r_i}{\sum_{i=1}^8 r_i} \quad (1)$$

To get load ratio, the process overhead of each transaction on a given PS needs to be known. Suppose that process overhead is measured in terms of the time period needed to complete a certain transaction, in the following we give a quantitative analysis on the process overhead needed for each type of transaction.

- PUBLISH Transactions:

The CPU time to process one initial PUBLISH transaction is denoted as $t_{initial_pub}$. The CPU time to process one refresh PUBLISH transaction is denoted as $t_{refresh_pub}$, which may be only half of the value of $t_{initial_pub}$. The CPU time to process one modifying PUBLISH transaction is denoted as t_{modify_pub} , which is a little less than $t_{initial_pub}$ and more than $t_{refresh_pub}$. The CPU time to process one terminating PUBLISH transaction is denoted as $t_{terminal_pub}$ and can be less than $t_{refresh_pub}$.

- SUBSCRIBE Transactions:

The CPU time to process one initial SUBSCRIBE transaction is denoted as $t_{initial_sub}$. The CPU time to process one refresh SUBSCRIBE transaction is denoted as $t_{refresh_sub}$. The CPU time to process one terminating SUBSCRIBE transaction is denoted as $t_{terminal_sub}$.

- NOTIFY Transactions:

The CPU time to process one NOTIFY transaction is denoted as t_{notify} .

Having gotten the process overhead needed for each type of transaction, the load incurred by each type of transaction can be denoted as $\rho_i = r_i \times t_i, i = 1, \dots, 8$, where t_i is the processing time for transaction type i . Traffic load ratio resulted by each type of traffic is given by:

$$v_i = \frac{\rho_i}{\sum_{i=1}^8 \rho_i} \quad (2)$$

In the following, we do a quantitative analysis of the traffic load using an assumed rate for each type of transaction in the unit of a hour, but with the real process overhead of each type of transaction in a reference PS. In our assumption, r_{login} and $n_{online_watcher}$ are taken as variables and the other rates are determined values.

$$r_{initial_pub} = r_{login} \quad (3)$$

$$r_{refresh_pub} = 2 \quad (4)$$

$$r_{terminal_pub} = r_{logout} = r_{login} \quad (5)$$

$$r_{modify_pub} = 0.5 \quad (6)$$

$$r_{notify} = n_{online_watcher} \times (r_{modify_pub} + r_{initial_pub} + r_{terminal_pub}) \quad (7)$$

$$r_{initial_sub} = 0.25 \quad (8)$$

$$r_{refresh_sub} = 2 \quad (9)$$

$$r_{terminal_sub} = 0.25 \quad (10)$$

Equation (3) means once a user logs in, an initial PUBLISH message is sent to a PS. Equation (4) refers to that a user's software will send out two PUBLISH messages per hour to a PS to refresh his state while the user is online. Suppose that the logout and login rate are the same for a user in long term statistics, so equation (5) means when a user logs out, a terminating PUBLISH is sent to a PS. Suppose that a user changes his status once every two hours, so (6) holds. Except for refresh PUBLISH, all the other PUBLISH messages result in NOTIFY messages being sent to all the online watchers, so equation (7) holds. A user subscribes to the status of his contact list once every four hours (8), and this subscription is refreshed by the user's software periodically with a rate of two times per hour (9). A user terminates his subscription once every four hours shown in (10).

The time for a presence server to process the transactions above is different and the following gives an example of the processing time, which is taken from a reference implementation of an IMS presence server. The time is in the unit of millisecond.

$$t_{initial_pub} = 6ms \quad (11)$$

$$t_{refresh_pub} = 3ms \quad (12)$$

$$t_{terminal_pub} = 2ms \quad (13)$$

$$t_{modify_pub} = 5ms \quad (14)$$

$$t_{notify} = 5ms \quad (15)$$

$$t_{initial_sub} = 10ms \quad (16)$$

$$t_{refresh_sub} = 8ms \quad (17)$$

$$t_{terminal_sub} = 3ms \quad (18)$$

Based on the assumptions above, the transaction rate ratio as well as the traffic load ratio resulted from the different type of messages is calculated based on equations (1) and (2). Fig.2(1) shows the relation between the login rate and the traffic ratio and Fig.2(2) gives the relation between the login rate and the load ratio. In Fig.2, $n_{online_watcher}$ is set to 10. Fig.2 shows that with the increase of login rate, the traffic ratio of NOTIFY messages also increases while the traffic ratios of some refresh messages decrease.

Fig.3 (1)(2) shows the relation between the number of online watchers and the traffic/load ratio. In Fig.3, $r_{login} = 1/4$, that is a user logs in once every 4 hours. Fig.3 indicates that when the number of online watchers is larger than 3, traffic load resulted by the NOTIFY traffic is much higher than the other types of traffic.

These figures give us an impression on the amount of traffic load resulted by each type of transactions and help us to identify the potential performance bottleneck when we implement or deploy a PS. In [8], a formal analysis on the relationship between user behaviors and messages to a PS is given. Based on the result of [8], user behaviors varies during the time of a day such that their login/logoff rate, online time also varies and the traffic to a PS varies greatly with the time of a day. Fig.4 gives an example messages ratio distribution during the time of a day when $n_{online_watcher} = 5$ and $n_{online_watcher} = 10$. The figure indicates that NOTIFY messages ratio varies with the time of a day, and it is always the largest part the traffic when $n_{online_watcher} = 10$.

All previous studies indicate that NOTIFY messages have great impact on traffic load to a PS as well as to the IMS core network, and its arrival rate to a PS varies greatly with the time of a day. So controlling NOTIFY messages as well optimizing their process mechanism is important to guarantee performance of PSs. It is proposed in [9] to reduce the number of online watchers and their watching time for a presentity to control the amount of NOTIFY messages. In the following sections, we study the NOTIFY message process mechanism in a PS to guarantee service quality.

III. MATHEMATICAL ANALYSIS

In this section, we study the queuing mechanism of NOTIFY messages in a PS and evaluate performance parameters of the mechanism. A PS records its users' status and a user's status can be subscribed by many watchers. Once a user's status is changed, a PUBLISH message is sent to the PS and the PS responses the PUBLISH message with a 200 OK message, then NOTIFY messages are scheduled to be sent to all the online watchers who have subscribed this user's status. The number of watchers of a user is N_w , and the number of online watchers of this user is denoted as $n_w \leq N_w$.

Fig.5 illustrates the process that when a user sends a PUBLISH message, and n_w NOTIFY messages are generated. Suppose that the arrival process of PUBLISH messages conforms to Poisson distribution with parameter λ_p , and the process time of PUBLISH message conforms to exponential distribution, then from the queuing networks result [17], the outgoing

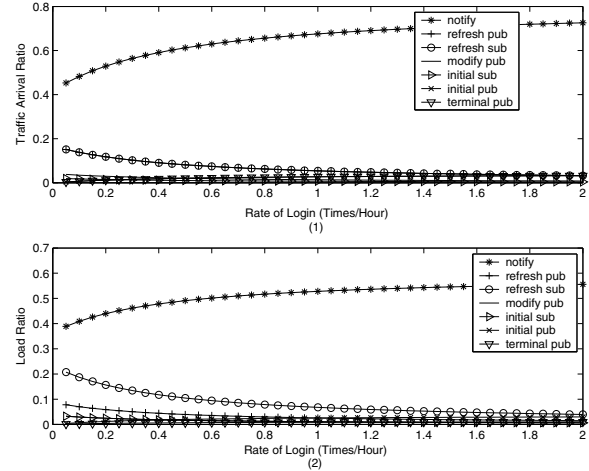


Fig. 2. Ratio of Traffic Load vs. User Login Rate, $n_{online_watcher} = 10$

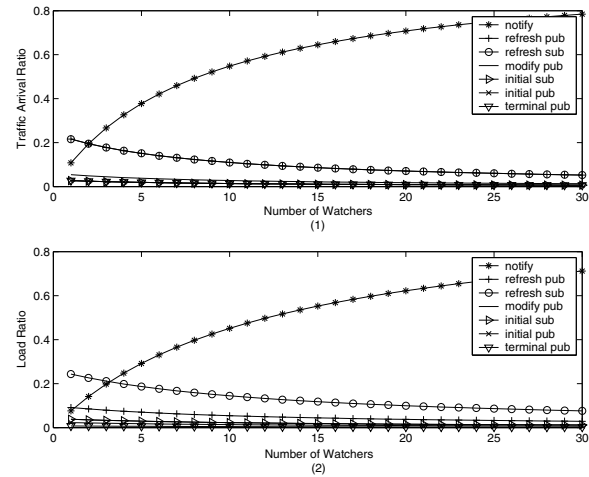


Fig. 3. Ratio of Traffic Load vs. Number of Watchers

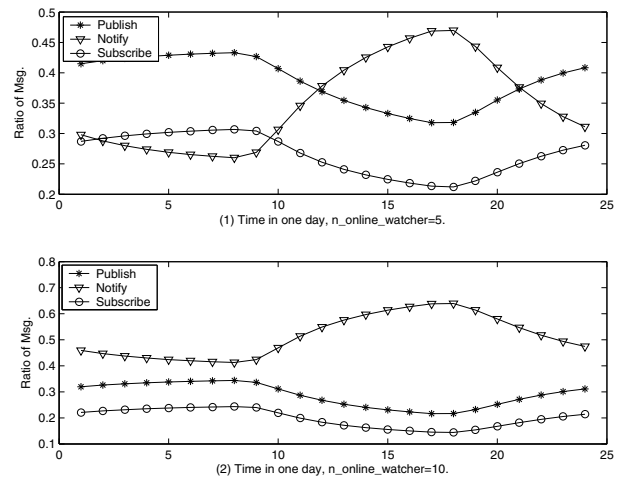


Fig. 4. Messages Ratio Distribution During the Time of a Day.

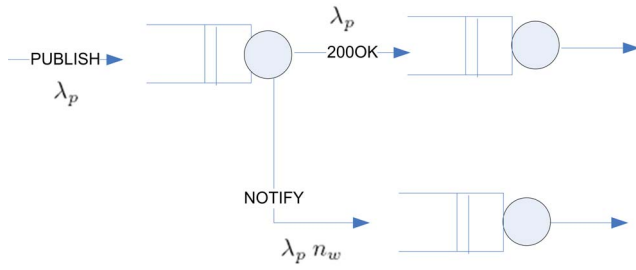


Fig. 5. Publish and Notify Queues

process of 200 OK messages is also Poisson distribution with parameter λ_p . With n_w online watchers, n_w NOTIFY messages are generated. The arrival process of NOTIFY messages is batch Poisson distribution with parameter λ_p and batch number n_w . 200OK messages are sent to the user as soon as possible to prevent message re-transmission. But NOTIFY messages are usually sent periodically with a time interval T because of some throttle or traffic control requirement. Instead of studying the combined queueing network, we only focus on the queueing mechanism of NOTIFY messages for it has occupied most part of the traffic load in a PS.

A. Queueing System with Controlled Vacation and Batch Poisson Arrival

From the description above, queue of NOTIFY messages can be taken as a queueing system with controlled vacation and batch Poisson arrival. That is, NOTIFY messages are sent periodically and when the queue has no NOTIFY messages, the server controlling the NOTIFY queue will be on vacation. When the vacation is ended after time interval T , NOTIFY messages in the queue will be sent. For each PUBLISH message results in multiple NOTIFY messages to different watchers, the arrival process to NOTIFY queue is in the batch of n_w . B is the maximum number of messages that can be buffered by NOTIFY queue and when the buffer is full, arriving messages are discarded. We make the following assumptions for the NOTIFY queue system shown in the lower part in Fig.5.

- 1) NOTIFY messages arrive at the system according to Poisson process with parameter λ_p and there is one server in the system. The queue discipline is FCFS.
- 2) Service time a NOTIFY message is assumed to be exponentially distributed with mean $\frac{1}{\mu}$. The traffic intensity is $\rho = \frac{\lambda_p n_w}{\mu} < 1$.
- 3) When the queue becomes empty, the server begins a vacation whose length is exponentially distributed with mean $T = \frac{1}{\theta}$. Upon completion of a vacation, the server resumes service if the queue is not empty. Otherwise, it takes another vacation having length independent of and identically distributed to the preceding one.

- 4) All aforementioned random variables are independent of each other.

Let $S(t) = 0$ and $S(t) = 1$ denote the events that the server is busy and on vacation at epoch t respectively. Define

$$p_{j,0}(t) = P(Q(t) = j, S(t) = 0), j = 1, 2, 3, \dots, B \quad (19)$$

$$p_{j,1}(t) = P(Q(t) = j, S(t) = 1), j = 0, 1, 2, \dots, B \quad (20)$$

$Q(t)$ denotes the number of messages in the system at time t and B is the maximum buffer size. From the theory of Markov chains, it follows that $\{Q(t), S(t), t \geq 0\}$ has a unique equilibrium distribution [18].

Setting

$$p_{j,0} = \lim_{t \rightarrow \infty} p_{j,0}(t), j = 1, 2, \dots, B \quad (21)$$

$$p_{j,1} = \lim_{t \rightarrow \infty} p_{j,1}(t), j = 0, 2, \dots, B \quad (22)$$

Fig.6 gives the state transition graph for the NOTIFY message queue with batch arrival and controlled vacation. For the simplicity of analysis, we suppose that B is divisible by n_w .

From Fig.6, we get the following state transition equations:

$$\lambda p_{0,1} = \mu p_{1,0} \quad (23)$$

$$(\lambda + \theta) p_{j n_w, 1} = \lambda p_{(j-1) n_w, 1}, \quad j = 1, \dots, \lfloor \frac{B}{n_w} \rfloor - 1 \quad (24)$$

$$\theta p_{B,1} = \lambda p_{B-n_w,1} \quad (25)$$

$$p_{j,1} = 0, \text{ otherwise} \quad (26)$$

$$(\lambda + \mu) p_{j,0} = \lambda p_{j-n_w,0} + \mu p_{j+1,0} + \theta p_{j,1}, \quad j = 2, \dots, B-1, j > n_w \quad (27)$$

$$(\lambda + \mu) p_{j,0} = \mu p_{j+1,0} + \theta p_{j,1}, j \leq n_w \quad (28)$$

$$(\lambda + \mu) p_{B,0} = \lambda p_{B-n_w,0} + \theta p_{B,1}, \quad (29)$$

$$\sum_{j=1}^B p_{j,0} + \sum_{j=0}^B p_{j,1} = 1 \quad (30)$$

This linear equations array (23)~(30) is denoted as model M_1 . M_1 can be solved using Matlab inverse functions to get $p_{j,0}, j = 0, \dots, B$ and $p_{j,1}, j = 1, \dots, B$. NOTIFY messages are used to notify watchers the presentity's status and it is undesirable if NOTIFY messages are lost unexpectedly and users get incorrect state information. So it is important to control queue length of NOTIFY to control the message loss probability. The probability that a message arrives and finds there are more than K messages in the buffer is:

$$P_K = \sum_{i=K+1}^B (p_{i,1} + p_{i,0}) \quad (31)$$

The probability of the buffer being full is: $P_B = p_{B,1} + p_{B,0}$
Mean queue length is :

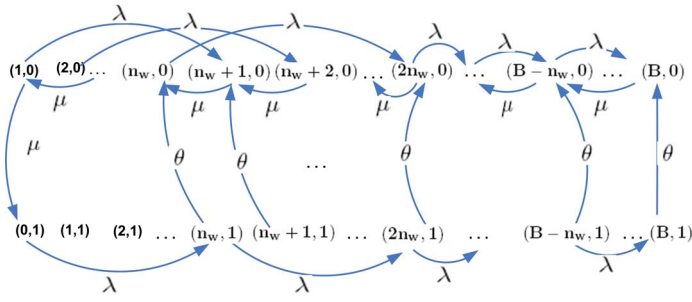


Fig. 6. State Transition Graph for Queue System with Batch Arrival and Controlled Vacation

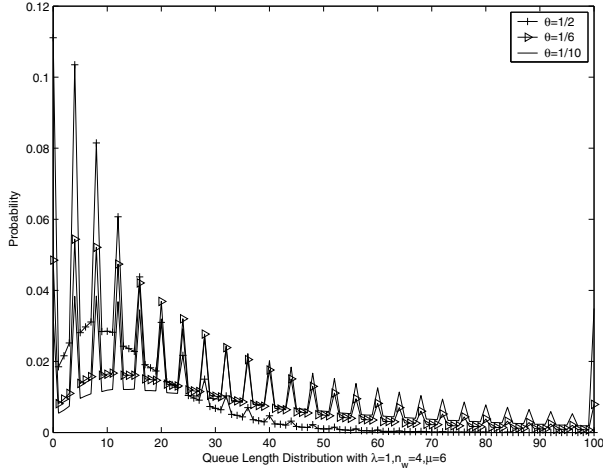


Fig. 7. Calculated Queue Length Distribution

$$\bar{Q} = \sum_{j=0}^B j(p_{j,0} + p_{j,1}), p_{0,0} = 0. \quad (32)$$

Variance of queue length is :

$$var(Q) = \sum_{j=0}^B j^2(p_{j,0} + p_{j,1}) - \bar{Q}^2, p_{0,0} = 0. \quad (33)$$

With $B = 100, \lambda_p = 1, n_w = 4, \mu = 6, K = 80$, we get the calculated $p_{j,0}, j = 0, \dots, B$ and $p_{j,1}, j = 1, \dots, B$ using Matlab for $\theta = \frac{1}{2}, \frac{1}{6}, \frac{1}{10}$. Fig.7 gives the distribution of queue length with different vacation intervals $T = \frac{1}{\theta}$. With the longer vacation time, the variation of queue length becomes larger. Fig.8 gives the calculation result for the mean and variance of queue length as well as the probability of queue length being larger than 80% of the buffer capacity. Fig.8 indicates that the longer vacation time, the longer the queue length and the larger the probability of queue length exceeding 80%B.

B. Experiment Result

To verify the correctness of the model, a queueing system with batch arrival and controlled vacation is simulated with Matlab using the same parameter configuration as the mathematical model calculation in previous subsection. In this

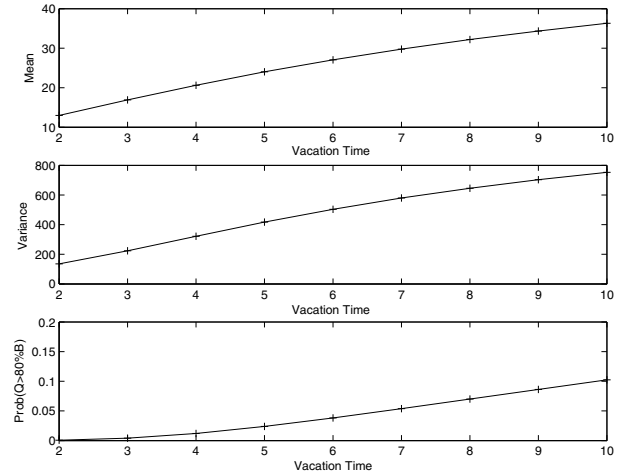


Fig. 8. Calculated Mean, Variance, $Prob(Q > 80\%B)$ with $\lambda = 1, n_w = 4, \mu = 6$.

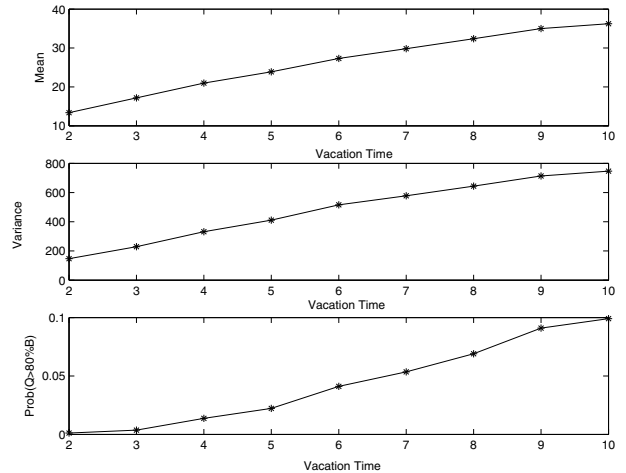


Fig. 9. Simulated Mean, Variance, $Prob(Q > 80\%B)$ with $\lambda = 1, n_w = 4, \mu = 6$.

simulation, each user has four online contacts and the user changes his status with rate $\lambda_p = 1$, each time the user changes his status, $n_w = 4$ NOTIFY messages are generated and queued in the NOTIFY queue. Presence server sends out Notify messages based on the policy given in previous section.

Fig.9 gives $P_K, \bar{Q}, var(Q)$ gotten from the simulation with $K = 80\%B$. Denote the simulated value for $P_K, \bar{Q}, var(Q)$ as $P'_K, \bar{Q}', var(Q')$. Set $\alpha(S) = \frac{|S' - S|}{S}$ as the difference between the simulated result S' and calculated result S , S is P_K, \bar{Q} or $var(Q)$. Comparing Fig.9 and Fig.8, we have $\alpha(P_K) \leq 0.025, \alpha(\bar{Q}) \leq 0.09, \alpha(var(Q)) \leq 0.0083$. This indicates that model M_1 is a pretty accurate description of the NOTIFY message queue.

C. Application of Mathematical Analysis

Traffic to a PS changes with the time during a day. User status may change frequently in some period while seldom change in the other period. Given the PUBLISH traffic to a PS λ_p, QoS parameter K, q and n_w , that is, the probability

of message queue length greater than K should be less than q , then we have the following optimization problem to solve:

$$\min \theta \quad (34)$$

s. t.

$$P_K \leq q \quad (35)$$

The objective (34) is to minimize server busy frequency θ , such that server vacation time $T = \frac{1}{\theta}$ can be maximized. By this objective, PS can have more time to process the other messages and the NOTIFY messages are not sent too frequently. Condition (35) means the probability of NOTIFY messages' queue length greater than K should be less than predefined quality parameter q . With this condition, queue length of NOTIFY messages can be controlled, so does the process delay and loss probability. There should be some other conditions on NOTIFY queue management such that NOTIFY message should not be delayed too much. In this paper, we only consider this single condition. The optimization problem specified by (34)(35) is denoted as \mathcal{F} . Based on the solution to problem \mathcal{F} , the timer controlling the NOTIFY queue in a PS can be tuned to optimize the system performance. An algorithm to get solution to \mathcal{F} is given as follows.

Timer Estimation Algorithm 1:

begin

1. Given PUBLISH traffic arrival rate λ_p , number of online watchers n_w , QoS parameter K, q and timer interval $I = [L, U], i = 1, \max I = L;$
2. for $i=L:U$ /* loop from L to U */
 $\theta = 1/i,$
Solve equations array M_1
Get P_K from equation (31)
if $P_K \leq q$ and $i > \max I, \max I = i;$
end
3. $\theta = \frac{1}{\max I},$ Return

end

Fig.10(1) gives the optimal timer for controlling the NOTIFY message queue with traffic arrival rate varying. In general, a higher arrival rate needs a shorter vacation time to guarantee the probability that the queue length is no more than the specified value. Also we find the timer is not very sensitive to the traffic arrival rate, and there are cases when arrival rate increases, the optimal timer value actually does not change. This means that the control timer needs not change very frequently according to the traffic variance to guarantee the performance requirement of NOTIFY queue. Traffic load can be divided into several levels and each level is associated with a control timer value is enough. Because the timer value is discrete with a granularity of 1 and the insensitiveness of optimal timer value, the loss probability of NOTIFY queue can vary for difference traffic load as indicated in Fig.10(2), but they all satisfies the condition specified by (35).

Whether the calculated timer applicable to a real system and whether it is optimal need to be studied. We use Mat-

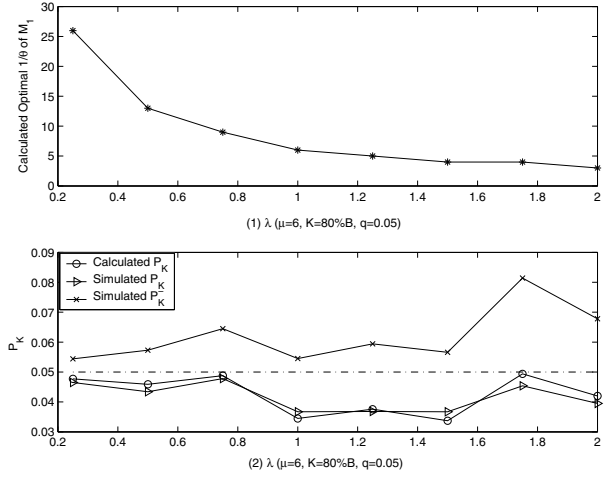


Fig. 10. Optimal Notify Timers

lab to simulate a NOTIFY message queueing system whose behavior conforms to the assumptions we made in section III-A. For each PUBLISH arrival rate $\lambda_p^i, i = 1, \dots, 8$, we get the optimal timer value $\hat{T}_i = \frac{1}{\theta_i}, i = 1, \dots, 8$ and its corresponding $P_K = P_K^i, i = 1, \dots, 8$. With each configuration $f_i = (\lambda_p^i, \hat{T}_i)$, we simulate the NOTIFY queueing system to get $P_K^i, i = 1, \dots, 8$, and denoted the P_K^i from simulation as *simulated* $P_K^i, i = 1, \dots, 8$. The comparison of $P_K^i, i = 1, \dots, 8$ and *simulated* $P_K^i, i = 1, \dots, 8$ is shown in Fig.10(2), which indicates that the simulation matched quite well with the calculation result.

Set $g_i = (\lambda_i, \hat{T}_i + 1), i = 1, \dots, 8$, that is for each traffic arrival rate $\lambda_i, i = 1, \dots, 8$, we set the vacation time one unit longer than the optimal vacation value $\hat{T}_i, i = 1, \dots, 8$, then we simulate the NOTIFY queueing system to get *simulated* P_K^- . From Fig.10, we can see that *simulated* $P_K^- > q$ for all $\lambda_i, i = 1, \dots, 8$. This verifies that the timer value found by solving the optimization problem \mathcal{F} not only satisfies the performance requirement of a real system, it is also the optimal.

IV. A SIMPLIFIED MODEL

Model M_1 gives an accurate description of Notify queue system, but it is a complicated system such that it's hard to get a close form expression. And it is always desirable to get an explicit expression of parameters of a queueing system to facilitate its application online to tune system configuration based on network traffic. As indicated in Fig.10(1), optimal control timer can be unchanged for some traffic load variance. It's possible to use a more simple model to get a control timer value to satisfy the performance requirement. In this section, we use Poisson arrival process to approximate the batch arrival process of a NOTIFY queue system to get a closed form solution of the system state as well as parameters of the NOTIFY queue system. The optimal timer value gotten from this simplified model is also studied to see if it is applicable in real system.

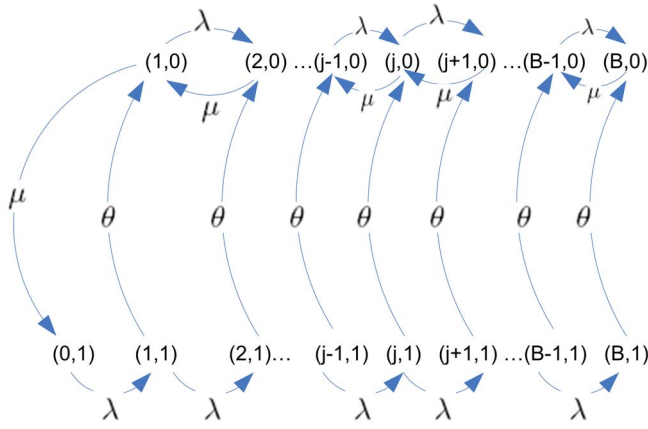


Fig. 11. State Transition Graph of the Simplified System

A. Queue System with Controlled Vacation and Poisson Arrival

If each user has n_w contacts, and the PUBLISH traffic conforms to Poisson distribution with parameter λ_p , then the NOTIFY traffic is approximated using a process conforming to Poisson distribution with parameter $\lambda = n_w \lambda_p$. The state transition graph of the simplified system is given in Fig.11.

From state transition graph shown in Fig.11, we have

$$\lambda p_{0,1} = \mu p_{1,0} \quad (36)$$

$$(\lambda + \mu)p_{1,0} = \mu p_{2,0} + \theta p_{1,1} \quad (37)$$

$$(\lambda + \mu)p_{j,0} = \lambda p_{j-1,0} + \mu p_{j+1,0} + \theta p_{j,1}, \quad j = 2, \dots, B-1. \quad (38)$$

$$\mu p_{B,0} = \lambda p_{B-1,0} + \theta p_{B,1} \quad (39)$$

$$(\theta + \lambda)p_{j,1} = \lambda p_{j-1,1}, j = 1, \dots, B-1. \quad (40)$$

$$\theta p_{B,1} = \lambda p_{B-1,1} \quad (41)$$

$$\sum_{j=1}^B p_{j,0} + \sum_{j=0}^B p_{j,1} = 1 \quad (42)$$

The equations array from (36)~(42) is denoted as model M_2 . Solution of this array is given in the below, the detailed proof is given in the appendix.

$$p_{B,1} = \frac{\mu - \lambda}{\mu(1 + \frac{\theta}{\lambda})^B + \mu \times J - \lambda \times (1 + \frac{\theta}{\lambda})^B - I \lambda} \quad (43)$$

$$I = \frac{\theta}{\lambda} \left(1 + \frac{\theta}{\lambda}\right)^{B-1} \frac{\lambda \left(1 - \left(\frac{\lambda}{\mu}\right)^B\right)}{\mu - \lambda} - \frac{\theta}{\mu} \frac{\theta}{\lambda} \sum_{j=0}^{B-2} \left(\frac{\lambda}{\mu}\right)^j \sum_{k=1}^{B-j-1} \left(1 + \frac{\theta}{\lambda}\right)^{B-k-1} \quad (44)$$

$$J = \frac{\lambda(B-1)}{\mu} \frac{\theta}{\lambda} \left(1 + \frac{\theta}{\lambda}\right)^{B-1} - \frac{\theta}{\mu} \sum_{j=1}^{B-1} (B-j) \frac{\theta}{\lambda} \left(1 + \frac{\theta}{\lambda}\right)^{B-j-1} \quad (45)$$

With $p_{B,1}$ being known, $p_{j,1}, j = 0, \dots, B-1$ can be gotten from the following equations.

$$p_{0,1} = \frac{\theta}{\lambda} \left(1 + \frac{\theta}{\lambda}\right)^{B-1} p_{B,1} \quad (46)$$

$$p_{j,1} = \frac{\theta}{\lambda} \left(1 + \frac{\theta}{\lambda}\right)^{B-1-j} p_{B,1}, \quad j = 1, \dots, B-1. \quad (47)$$

Then $p_{j,0}, j = 1, \dots, B$ can be gotten from equations (48) and (49).

$$p_{1,0} = \frac{\lambda}{\mu} p_{0,1} \quad (48)$$

$$p_{j,0} = \frac{\lambda}{\mu} p_{j-1,0} + \frac{\lambda}{\mu} p_{0,1} - \frac{\theta}{\mu} \sum_{k=1}^{j-1} p_{k,1}, \quad j = 2, \dots, B. \quad (49)$$

Corresponding P_K, \bar{Q} and $var(Q)$ can be gotten from equations (31), (32) and (33).

B. Experiment Result

Model M_2 is a simplified description of the NOTIFY queue system and it is necessary to study if it is applicable to tune the timers of NOTIFY message queue based on this model.

When traffic arrival rate of PUBLISH messages is λ_p , the NOTIFY arrival rate is approximated as Poisson arrival with parameter $\lambda = n_w \lambda_p$. For different $\lambda_p = 0.2, 0.4, \dots, 2$, we can get the optimal timer value $\hat{T}_i' = \frac{1}{\theta_i'}, i = 1, \dots, 8$ by solving the following optimization problem \mathcal{F}' , where P_K' is based on model M_2 . Algorithm 1 can also be used to solve problem \mathcal{F}' with P_K being replaced by P_K' .

$$\min \theta \quad (50)$$

s.t.

$$P_K' \leq q \quad (51)$$

Fig.12(1) gives the comparison of the optimal vacation time \hat{T}_i and $\hat{T}_i', i = 1, \dots, 10$ calculated from M_1 and M_2 , which indicates that there is very little difference between the two optimal values. Vacation time gotten from M_2 is a little longer than that gotten from M_1 . The calculated optimal timer value \hat{T}_i and $\hat{T}_i', i = 1, \dots, 10$ is used to control a NOTIFY queue simulated in MATLAB and their corresponding P_K is shown in Fig.12(2). Fig.12(2) indicates that NOTIFY queue controlled by $\hat{T}_i', i = 1, \dots, 10$ can satisfy the quality requirement of NOTIFY queue under most traffic conditions. But there are also occasions that the vacation time is too long to satisfy the quality requirement. That is resulted by the "inaccurate" modeling of traffic arrival process, and such a loss of information results in some negative impact. To trade off model complexity and quality requirement, it is recommended to use more tight quality requirement (with less q) when applying M_2 to calculate control timers.

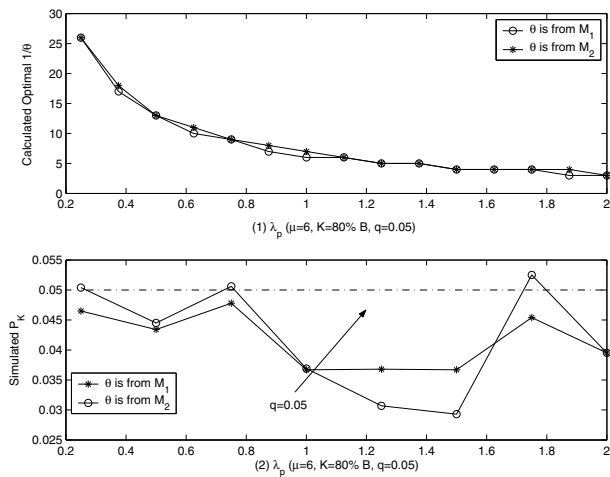


Fig. 12. Optimal Vacation Time Comparison between M_1 and M_2

V. CONCLUSION

The PS is an indispensable component of the IMS architecture, the performance of an IMS PS is key to many IMS applications that depend on presence service. Our study shows NOTIFY messages account for the largest portion of traffic load to a PS and their processing overhead has great impact on the quality of service of the PS. In this paper we use a mathematical model to study the queue length distribution of the NOTIFY message queue as well as some important parameters of the NOTIFY message queue, such as the mean queue length (\bar{Q}), the variance of the queue length ($var(Q)$), and the probability of the queue length being greater than a predefined threshold (P_K). We also apply the mathematical model to get the optimal timer value (\hat{T}) to control the NOTIFY message queue. Actually these performance parameters are directly related to many of the metrics considered by operators during the network sizing before deploying a presence service. Value of \bar{Q} reflects the average memory space required by the NOTIFY message queue; $var(Q)$ gives the variance of the memory space requirement which can be used to estimate the peak hour memory space occupation. Also it is always undesirable for a server to lose messages unexpectedly due to queue overflow, P_K can be used by an operator to find out the loss probability of NOTIFY messages with a given hardware configuration. The value of the timer that controls the NOTIFY message queue, \hat{T} , is a good indication of the CPU usage needed for processing NOTIFY messages, in general, the operator may want to control the CPU usage of the PS under a certain threshold. Moreover, this model can also be applied to a system with multiple CPUs.

There are still lots of work need further study. In this paper, we only consider one service policy in section III-A, however there can be many variations in a real system. For example, a real PS may send out NOTIFY messages in a determined time interval without considering the situation of present NOTIFY queue; or it may send out NOTIFY messages in an interval $T + \Delta t$, with T a fixed value and only Δt a random variable.

How different policies impact on the server's performance needs further study. Also when we apply the mathematical model to get the optimal value of the control timer for the NOTIFY message queue, only one constraint is considered, which may result in the notification time intervals being too long especially when the traffic load is light. To avoid this situation, some other constraints such as the maximum delay between a PUBLISH message and a NOTIFY message could also be considered.

REFERENCES

- [1] Carlos Urrutia-Valds, Amit Mukhopadhyay, and Mohamed El-Sayed, "Presence and Availability with IMS: Applications Architecture, Traffic Analysis, and Capacity Impacts", Bell Labs Technical Journal 10(4),2006, 101 – 107.
- [2] 3rd Generation Partnership Project, "Presence Service; Stage 1 (Release 6)", 3GPP TS 22.141, Sept. 2003, <<http://www.3gpp.org/ftp/Specs/html-info/22141.htm>>.
- [3] 3rd Generation Partnership Project, "Presence Service, Architecture and Functional Description (Release 6)", 3GPP TS 23.141, Sept. 2004, <<http://www.3gpp.org/ftp/Specs/html-info/23141.htm>>.
- [4] M. Day, J. Rosenberg, and H. Sugano, "A Model for Presence and Instant Messaging", IETF RFC 2778, Feb. 2000, <<http://www.ietf.org/rfc/rfc2778.txt?number=2778>>.
- [5] 3rd Generation Partnership Project, "Presence Service Based on Session Initiation Protocol; Functional Models, Information Flows and Protocol Details (Release 6)", 3GPP TR 24.841, June 2004, <<http://www.3gpp.org/ftp/Specs/html-info/24841.htm>>.
- [6] M. Day, S. Aggarwal, G. Mohr, and J. Vincent, "Instant Messaging/Presence Protocol Requirements", IETF RFC 2779, Feb. 2000, <<http://www.ietf.org/rfc/rfc2779.txt?number=2779>>.
- [7] A. B. Roach, "Session Initiation Protocol (SIP) Specific Event Notification", IETF RFC 3265, June 2002, <<http://www.ietf.org/rfc/rfc3265.txt?number=3265>>.
- [8] Zhengzheng Cao, Caixia Chi,Ruibing Hao, "User Behavior Modelling and Traffic Analysis of IMS Presence Servers", to appear in proceeding of IEEE GLOBECOM 2008.
- [9] Muhammad T. Alam,Zheng Da Wu, "Admission Control Approaches in the IMS Presence Service", INTERNATIONAL JOURNAL OF COMPUTER SCIENCE VOLUME 1 NUMBER 4 2006 ISSN 1306-4428,299 – 314.
- [10] A. B. Roach, et.al., "A Session Initiation Protocol (SIP) Event Notification Extension for Resource Lists", IETF RFC 4662.
- [11] M. Lonnfors,et.al., "Session Initiation Protocol (SIP) extension for Partial Notification of Presence Information", draft-ietf-simple-partial-notify-09, Internet-Draft.
- [12] A. Niemi , "Session Initiation Protocol (SIP) Event Notification Extension for Notification Throttling", draft-niemi-sipping-event-throttle-05, Internet-Draft.
- [13] H. Hannu,"Signaling Compression (SigComp) Requirements & Assumptions", IETF RFC 3322.
- [14] Muhammad T. Alam,Zheng Da Wu,"End-to-End Delay Measurement for Instant Messaging Relay Networks", <http://epublications.bond.edu.au/infotech_pubs/28>.
- [15] M.Pous, et.al."Performance Evaluation of a SIP Based Presence and Instant Messaging Service For UMTS", The Institution of Electrical Engineers. 254 – 258.
- [16] Henning Schulzrinne, Sankaran Narayanan and Jonathan Lennox, "SIPstone - Benchmarking SIP Server Performance", <<https://developer.ubiquitysoftware.com/support/whitepapers/sipstone-benchmarking-sip-server-performance>>
- [17] Hayes, Jeremiah F., Chapter 7 in "Modeling and Analysis of Computer Communications Networks", 1984 Plenum Press, New York.
- [18] B.T.Doshi, "Queueing Systems with Vacations -A Survey", Queueing Systems 1 (1986) 29 – 66.

APPENDIX

Solve model M_2 specified as follows:

$$\lambda p_{0,1} = \mu p_{1,0} \quad (52)$$

$$(\lambda + \mu)p_{1,0} = \mu p_{2,0} + \theta p_{1,1} \quad (53)$$

$$(\lambda + \mu)p_{j,0} = \lambda p_{j-1,0} + \mu p_{j+1,0} + \theta p_{j,1},$$

$$j = 2, \dots, B-1. \quad (54)$$

$$\mu p_{B,0} = \lambda p_{B-1,0} + \theta p_{B,1} \quad (55)$$

$$(\theta + \lambda)p_{j,1} = \lambda p_{j-1,1}, j = 1, \dots, B-1. \quad (56)$$

$$\theta p_{B,1} = \lambda p_{B-1,1} \quad (57)$$

$$\sum_{j=1}^B p_{j,0} + \sum_{j=0}^B p_{j,1} = 1 \quad (58)$$

From equations (56)and (57), we have

$$p_{0,1} = \left(1 + \frac{\theta}{\lambda}\right) p_{1,1} \quad (59)$$

$$p_{1,1} = \left(1 + \frac{\theta}{\lambda}\right) p_{2,1} \quad (60)$$

...

$$p_{B-2,1} = \left(1 + \frac{\theta}{\lambda}\right) p_{B-1,1} \quad (61)$$

$$p_{B-1,1} = \frac{\theta}{\lambda} p_{B,1} \quad (62)$$

By multiplying $(1 + \frac{\theta}{\lambda})^j$ on each side of equations (60)~ (62) and sum them up, we have

$$p_{0,1} = \frac{\theta}{\lambda} \left(1 + \frac{\theta}{\lambda}\right)^{B-1} p_{B,1} \quad (63)$$

$$p_{j,1} = \frac{\theta}{\lambda} \left(1 + \frac{\theta}{\lambda}\right)^{B-1-j} p_{B,1}$$

$$j = 1, \dots, B-1. \quad (64)$$

From equations (36)~ (39), we have

$$p_{1,0} = \frac{\lambda}{\mu} p_{0,1} \quad (65)$$

$$p_{j,0} = \frac{\lambda}{\mu} p_{j-1,0} + \frac{\lambda}{\mu} p_{0,1} - \frac{\theta}{\mu} \sum_{k=1}^{j-1} p_{k,1},$$

$$j = 2, \dots, B. \quad (66)$$

Sum up $p_{j,1}, j = 2, \dots, B$ and from equation (64), we have

$$\sum_{k=0}^{j-1} p_{k,1} = \sum_{k=0}^{j-1} \frac{\theta}{\lambda} \left(1 + \frac{\theta}{\lambda}\right)^{B-1-k} p_{B,1}, j = 2, \dots, B. \quad (67)$$

For each equation $p_{j,0}$ of (66), multiple $(\frac{\lambda}{\mu})^{B-j}$ on each side and sum up all equations for $j = 1, \dots, B$, we have

$$p_{B,0} = \sum_{k=1}^B \left(\frac{\lambda}{\mu}\right)^k p_{0,1} - \frac{\theta}{\mu} \sum_{j=0}^{B-2} \left(\frac{\lambda}{\mu}\right)^j \sum_{k=1}^{B-j-1} p_{k,1} \quad (68)$$

$$= I \times p_{B,1} \quad (69)$$

$$I = \frac{\theta}{\lambda} \left(1 + \frac{\theta}{\lambda}\right)^{B-1} \frac{\lambda \left(1 - \left(\frac{\lambda}{\mu}\right)^B\right)}{\mu - \lambda}$$

$$- \frac{\theta}{\mu \lambda} \sum_{j=0}^{B-2} \left(\frac{\lambda}{\mu}\right)^j \sum_{k=1}^{B-j-1} \left(1 + \frac{\theta}{\lambda}\right)^{B-k-1} \quad (70)$$

For $j = 1, \dots, B$, sum up equations (65)(66), we

$$\sum_{j=1}^B p_{j,0} = \frac{B\lambda}{\mu} p_{0,1} + \frac{\lambda}{\mu} \sum_{j=1}^{B-1} p_{j,0} - \frac{\theta}{\mu} \sum_{j=1}^{B-1} (B-j) p_{j,1}$$

$$= \frac{\lambda}{\mu} \sum_{j=1}^{B-1} p_{j,0} + \frac{B\lambda}{\mu} \frac{\theta}{\lambda} \left(1 + \frac{\theta}{\lambda}\right)^{B-1} p_{B,1}$$

$$- \frac{\theta}{\mu} \sum_{j=1}^{B-1} (B-j) \frac{\theta}{\lambda} \left(1 + \frac{\theta}{\lambda}\right)^{B-j-1} p_{B,1} \quad (71)$$

From equation (42), we have

$$\sum_{j=1}^B p_{j,0} = 1 - \sum_{j=0}^B p_{j,1} = 1 - \left(1 + \frac{\theta}{\lambda}\right)^B p_{B,1} \quad (72)$$

Substitute (72) into equation (71), we have

$$p_{B,0} = 1 - \frac{\mu}{\lambda} - p_{B,1} \left(1 + \frac{\theta}{\lambda}\right)^B + p_{B,1} \frac{\mu}{\lambda} \left(1 + \frac{\theta}{\lambda}\right)^B$$

$$+ \frac{\mu}{\lambda} \times J \times p_{B,1} \quad (73)$$

$$J = \frac{\lambda(B-1)}{\mu} \frac{\theta}{\lambda} \left(1 + \frac{\theta}{\lambda}\right)^{B-1}$$

$$- \frac{\theta}{\mu} \sum_{j=1}^{B-1} (B-j) \frac{\theta}{\lambda} \left(1 + \frac{\theta}{\lambda}\right)^{B-j-1} \quad (74)$$

Equation (73) equals to equation (69), we have

$$I \times p_{B,1} = 1 - p_{B,1} \left(1 + \frac{\theta}{\lambda}\right)^B + J \times p_{B,1} \times \frac{\mu}{\lambda}$$

$$+ p_{B,1} \left(1 + \frac{\theta}{\lambda}\right)^B \frac{\mu}{\lambda} - \frac{\mu}{\lambda}$$

$$p_{B,1} = \frac{\mu - \lambda}{\mu \left(1 + \frac{\theta}{\lambda}\right)^B + \mu \times J - \lambda \times \left(1 + \frac{\theta}{\lambda}\right)^B - I\lambda} \quad (76)$$

Having gotten $p_{B,1}$, from equations (63),(64) and (65), (66) $p_{j,0}, j = 1, \dots, B$ and $p_{j,1}, j = 0, \dots, B-1$ can be obtained.