國立交通大學

網路工程研究所

碩 士 論 文

減少 IMS 現狀資訊訊息流量之弱一致性方法

A Weakly Consistent Scheme for IMS Presence Service

研 究 生：劉仁煌

指導教授：林一平 教授

陳懷恩 教授

中 華 民 國 九 十 八 年 六 月

減少 IMS 現狀資訊訊息流量之弱一致性方法
A Weakly Consistent Scheme for IMS Presence Service

研 究 生：劉仁煌　　　　Student：Ren-Huang Liou

指導教授：林一平博士　　　　Advisor：Dr. Yi-Bing Lin

陳懷恩博士　　　　Dr. Whai-En Chen

國 立 交 通 大 學

網 路 工 程 研 究 所

碩 士 論 文

A Thesis

Submitted to Institute of Network Engineering

College of Computer Science

National Chiao Tung University

in partial Fulfillment of the Requirements

for the Degree of

Master

in

Computer Science

June 2009

Hsinchu, Taiwan, Republic of China

中華民國九十八年六月

減少IMS現狀資訊訊息流量之弱一致性方法

學生：劉仁煌　　　　　　　　　　指導教授： 林一平博士

陳懷恩博士

國立交通大學網路工程研究所碩士班

摘　　　要

在第三代行動通訊系統UMTS (Universal Mobile Telecommunications System)中，現狀資訊服務(Presence Service)是由 IP多媒體子系統(IP Multimedia Core Network Subsystem; IMS)負責提供。在IMS中，現狀資訊伺服器(Presence Server)負責將更新後的現狀資訊(Presence Information) 通知給已授權的觀看者(Watcher)。如果現狀資訊更新的頻率比觀看者瀏覽的頻率還要高，現狀資訊伺服器會產生許多不必要的通知訊息(Notification)。本論文實作一完整即時訊息與現狀資訊服務系統。在此實作，本論文使用一個弱一致性的方法，稱為延遲更新(Delayed Update)，來減少發送通知訊息造成的流量。此方法使用一個計時器來控制傳送通知訊息的頻率。本論文針對延遲更新進行效能分析。研究結果顯示，延遲更新可以有效減少發送通知訊息造成的流量，而且不會嚴重降低觀看者看到錯誤資訊的機率。

關鍵字： 延遲更新, IP多媒體子系統, 現狀資訊服務

A Weakly Consistent Scheme for IMS Presence Service

Student: Ren-Huang Liou   Advisors: Prof. Yi-Bing Lin

Prof. Whai-En Chen

Institute of Network Engineering

National Chiao Tung University

ABSTRACT

IP Multimedia Core Network Subsystem (IMS) provides presence service for *Universal Mobile Telecommunications System* (UMTS). In IMS, the presence server is responsible for notifying an authorized watcher of the updated presence information. If the updates occur more frequently than the accesses of the watcher, the presence server will generate many unnecessary notifications. In this thesis, we implemented an instant messaging and presence service system. In our implementation, we use a weakly consistent scheme (called delayed update) to reduce the notification traffic. In this scheme, a delayed timer is defined to control the notification rate. We propose an analytic model and simulation experiments to investigate the performance of delayed update. The study indicates that delayed update can effectively reduce the notification traffic without significantly degrading the valid access probability.

Keywords: Delayed Update, IP Multimedia Core Network Subsystem (IMS), Presence Service

誌　　謝

首先誠摯地感謝指導教授林一平博士與陳懷恩博士，沒有老師的專業建議與細心指導，我無法完成此篇碩士論文。在林一平博士嚴格而且不厭其煩地指導中，我學習到了研究的方法與態度，並從中獲得極大的助益。在陳懷恩博士的指導中，讓我獲得論文研究方向與許多專業知識。我也要感謝楊舜仁博士與顏在賢博士，你們在碩士班口試中給予許多寶貴的建議。

接著感謝實驗室的同伴們，在我論文撰寫的期間，提供了相當多有用的意見。謝謝他們的陪伴，使得我碩士班的生活更加充實。

最後我也要感謝我的家人與我的朋友們在我研究生活中給予的鼓勵及支持。

Contents

List of Tables

List of Figures

Chapter 1

Introduction

Presence service allows an Internet user to access another user's presence information including the user status, the activities (e.g., working, playing, etc.), the email/phone addresses, and so on. This service is typically utilized together with the instant messaging applications such as Windows Live Messenger.

In *Universal Mobile Telecommunication System* (UMTS), presence service is provided through *IP Multimedia Core Network Subsystem* (IMS) [2, 8]. Figure 1.1 illustrates a simplified UMTS network architecture for presence service [1]. In this architecture, a user with a *User Equipment* (UE; Figure 1.1 (1)) accesses presence service in IMS (Figure 1.1 (b)) via the UMTS (Figure 1.1 (a)). The user plays the role as a *presentity* when he/she provides information to the presence server (Figure 1.1 (3)). On the other hand, the user is called a *watcher* if he/she accesses other users' information from the presence server. In IMS, *Call Session Control Function* (CSCF; Figure 1.1 (2)) is responsible for carrying out the control signaling. As an IMS application server, the presence server stores the information of the presentities, and determines the authorized watchers who can access the presence information.

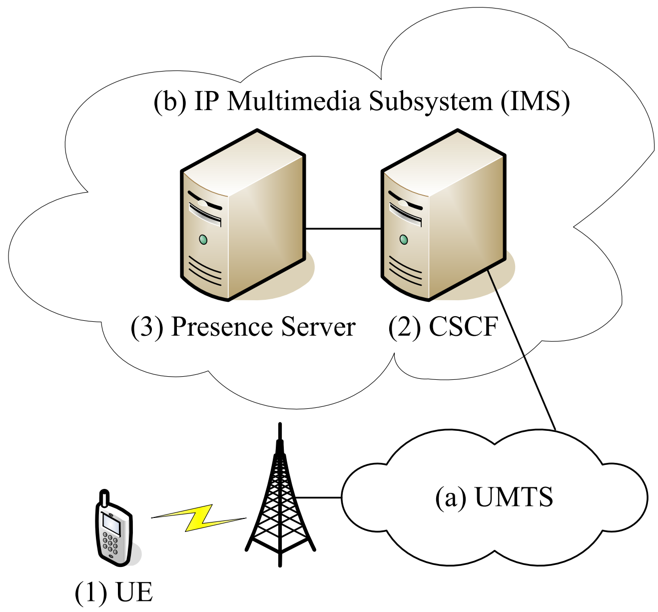

Figure 1.1: The UMTS Network Architecture for Presence Service

1.1 Implementation of Instant Messaging and Presence Service System

3rd Generation Partnership Project (3GPP) defines the presence service procedures for IMS. Based on 3GPP standards, we have implemented an *instant messaging and presence service* (IMPS) system including a presence server and an IMPS client (as illustrated in Figure 1.2). The presence server is implemented using the C language, which runs on the ADLINK Industrial Computer RK110SB-D3S2-852LV-3.0M4G with the Linux operating system. On the other hand, the IMPS client is written using the C++ language, which runs on the Dopod 838 with the Microsoft Windows Mobile platform.

Figure 1.3 shows the user interface of the IMPS client, which consists of two windows. The Main window (Figure 1.3 (a)) is popped up after the IMPS client program is executed. It waits for the user to input the username, the password, and the address of the presence server (Figure 1.3 (1)). After the user clicks the Login buttom (Figure 1.3 (2)), the IMPS client executes the presence service procedures to be described in Chapter 2. After the user logins the presence service, the Main window shows the user's contact list (Figure 1.3 (b)). Each item in the contact list represents a user's friend. The friend's status is represented by a status icon

Figure 1.2: Instant Messaging and Presence Service System

(e.g., the green icon means that the status is available). The user can add/delete a contact through the File menu (Figure 1.3 (3)). The File menu also offers the following commands: logout the presence service, change status, change display name, and exit the program. When the user wants to contact his/her friend, the user clicks the corresponding item (Figure 1.3 (4)) in the contact list. Then, the Message window (Figure 1.3 (c)) is popped up. The user types a message in the Input Text area (Figure 1.3 (5)), and sends it by clicking the Send buttom (Figure 1.3 (6)). The Message Log area (Figure 1.3 (7)) displays the message history.

The above implementation won the 3rd place at *Ministry of Education* (MoE) Communication Competition Contest 2006, and won the 2nd place at *National Chiao Tung University* (NCTU) *Computer Science* (CS) Senior Project Contest 2006.

1.2 Organization

The 3GPP presence service procedures may incur large traffic that significantly consumes the limited radio resources and UE power. To resolve this issue, many mechanisms have been proposed [9]. In this thesis, we consider a weakly consistent scheme called *delayed update*

(a) Main Window (Before Login)   (b) Main Window (After Login)   (c) Message Window

Figure 1.3: The User Interface of Instant Messaging and Presence Service Client

exercised at the presence server to reduce the presence traffic between a watcher and the presence server. In this scheme, the UE needs not be modified.

This thesis addresses the issue of reducing the network traffic in the IMS presence service. This thesis is organized as follows. Chapter 2 describes the message flows for the 3GPP IMS presence service, and then introduces delayed update. Chapter 3 describes the input parameters and output measures for modeling delayed update. Chapters 4-5 propose an analytic model to study the performance of delayed update. Chapter 6 investigates delayed update by numerical examples, and the conclusions are given in Chapter 7.

Chapter 2

Message Flows for IMS Presence Service

This chapter describes the message flows for the 3GPP presence service procedures including the subscription, the publication, and the notification [1, 3]. These procedures are implemented by utilizing *Session Initiation Protocol* (SIP) [10, 11, 12, 16]. Then we introduce delayed update for reducing the notification traffic.

2.1 Subscription Procedure

When a watcher attempts to access the presence information of other users (i.e., the presentities), the subscription procedure is executed. In our implementation (see Section 1.1), the subscription procedure is triggered when the user logins the presence service. Figure 2.1 illustrates the subscription message flow between the watcher and the presence server. Detailed steps are described as follows:

Step A.1: The watcher (i.e., a UE) issues a SIP SUBSCRIBE message to the presence server through the UMTS and the CSCF.

Step A.2: After verifying the identify of the watcher [4], the presence server sends a 200 OK message to the watcher. The message indicates that the watcher is authorized to subscribe to the presence information.

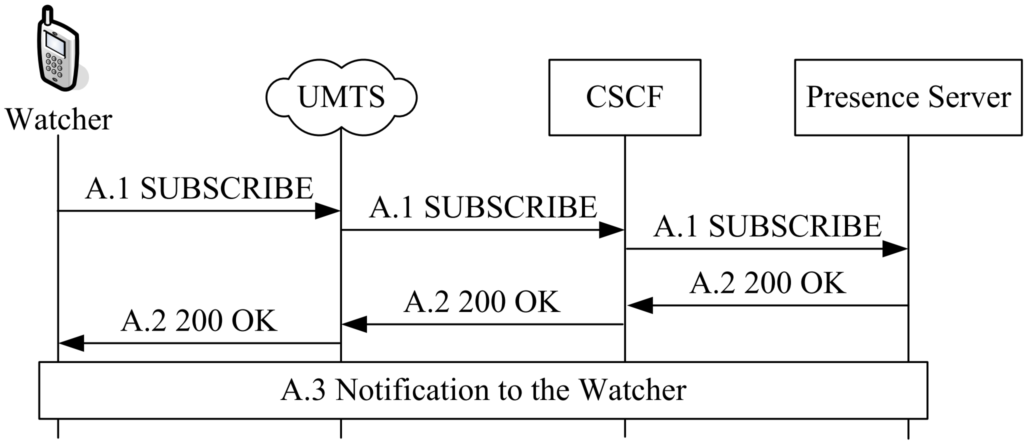

Figure 2.1: The Subscription Procedure

Step A.3: The presence server sends the presence information to the watcher through the notification procedure to be described in Figure 2.3.

2.2 Publication Procedure

The publication procedure is executed when a presentity changes his/her presence information (e.g., the activity is changed from "playing" to "working"). Figure 2.2 illustrates the publication message flow with the following steps:

Step B.1: A SIP PUBLISH message is issued by the presentity to the presence server through the UMTS and the CSCF. This message contains the updated presence information.

Step B.2: After the presence server received this SIP PUBLISH message, it stores the presence information in association with this presentity, and returns the 200 OK message. This message indicates that the publication is successful.

2.3 Notification Procedure

The presence server pushes the subscribed presence information to an authorized watcher through the notification procedure when the information is updated. Figure 2.3 illustrates

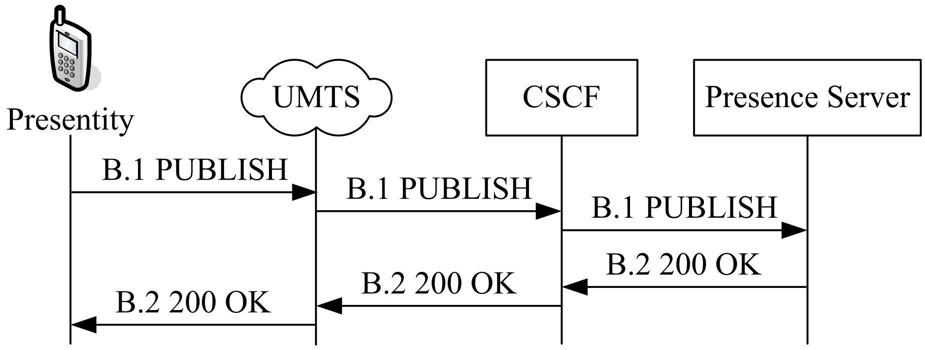

Figure 2.2: The Publication Procedure

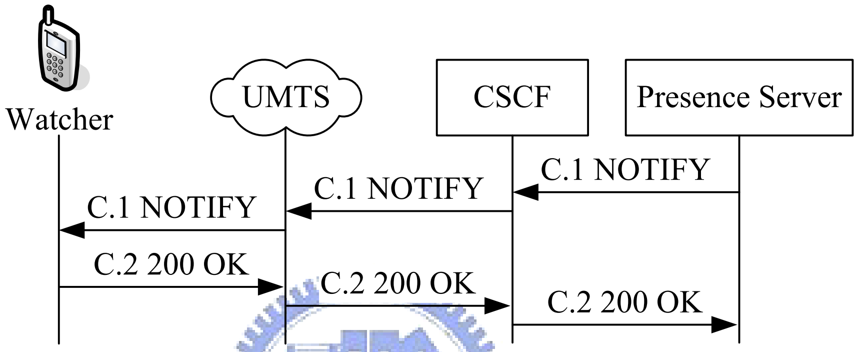

Figure 2.3: The Notification Procedure

the notification message flow with the following steps:

Step C.1: The presence server sends a SIP NOTIFY message to the watcher through the CSCF and the UMTS. This message contains the modified presence information.

Step C.2: The watcher returns the 200 OK message to indicate that the presence information is successfully received.

Note that the notification procedure is a "push" action. A watcher may access the presence information through "pull" action such as the poll-each-read mechanism. The details can be found in [7].

2.4 Delayed Update

The presence server immediately notifies a watcher of the presence information updates. If the updates occur more frequently than the accesses of the watcher, then the presence server will generate many notifications. A weakly consistent scheme called delayed update can be used to reduce the notification traffic between the presence server and the watcher (i.e., Steps C.1 and C.2 in Figure 2.3).

In delayed update, when the presence server receives the updated presence information (i.e., Step B.1 in Figure 2.2), the watcher is not notified of the updated information immediately. Instead, the presence server starts a delayed timer with a period T. This period is referred to as the *delayed threshold*. If the presence information is updated again within period T, the old information in the presence server is replaced by the new one. When the timer expires, the presence server notifies the watcher of the presence information (i.e., Step C.1 in Figure 2.3 is executed). Therefore, the notifications for the updates in T are saved. On the other hand, if an access occurs at any time in T, the watcher may access the obsolete information. RFC 3856 describes a mechanism similar to delayed update, where $T = 5$ seconds [13]. However, this default delayed threshold value may not fit all user activities, and it is important to select an appropriate T value such that notification traffic is reduced while the watcher only occasionally accesses the obsolete presence information.

Chapter 3

Input Parameters and Output Measures

This chapter describes the input parameters and output measures for modeling delayed update. Figure 3.1 illustrates a timing diagram where the watcher accesses the presence information at t_0, t_3, and t_8. The presence information is updated at t_1, t_4, t_5, t_6, and t_9. At t_2 and t_7, the presence server notifies the watcher of modified presence information. In this figure, the delayed threshold $T = t_2 - t_1$ (also $t_7 - t_4$) is a random variable with the mean $1/\gamma$. Let N be the number of update messages occurring during T. In this example, $N = 0$ in $[t_1, t_2]$, and $N = 2$ in $[t_4, t_7]$. The inter-access interval $\tau_a = t_3 - t_0$ (also $t_8 - t_3$) is a random variable with the mean $1/\mu$ and the variance V_a. Let the inter-update interval $\tau_u = t_4 - t_1$ (also $t_5 - t_4$, $t_6 - t_5$, and $t_9 - t_6$) be a random variable which has density function $f_u(\tau_u)$, the mean $1/\lambda$, the variance V_u, and Laplace transform $f_u^*(s)$. Let $\tau = t_6 - t_4$ be the interval between the first update message and the last update message occurring in T. We define an *observation interval* as a period between when the delayed threshold begins and when the first update message occurs after the end of the delayed threshold. In Figure 3.1, $[t_1, t_4]$ and $[t_4, t_9]$ are the observation intervals.

We consider two output measures:

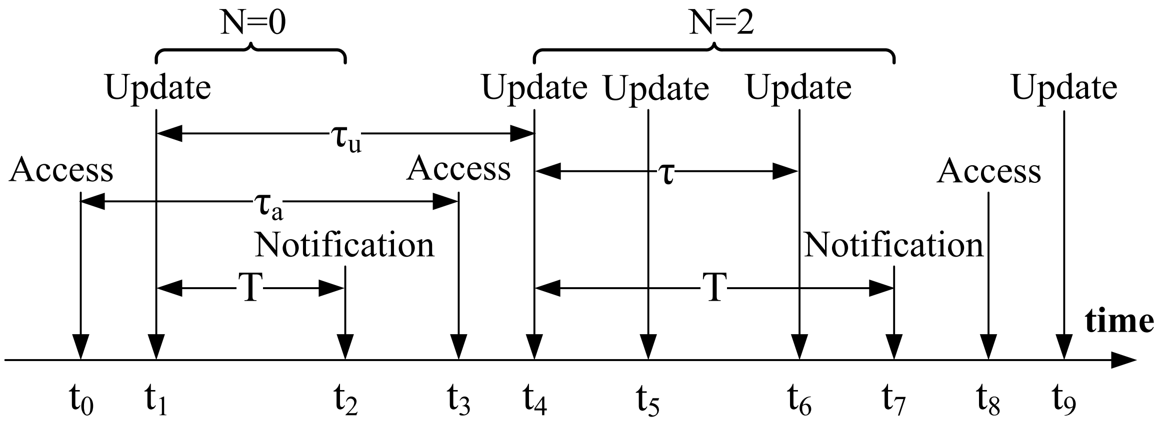

Figure 3.1: Timing Diagram for Delayed Update

- p: the probability that the watcher accesses the valid presence information. In Figure 3.1, an access is valid if it occurs in $[t_2, t_4]$ and $[t_7, t_9]$.
- $E[N]$: the expected number of update messages occurring in T.

It is clear that the update messages occurring in T are the saved notification messages. Therefore, the larger the p and the $E[N]$ values, the better the performance of delayed update.

The notation used in this thesis is summarized below.

- T: the delayed threshold.
- p: the probability that the watcher accesses the valid presence information.
- N: the number of update messages occurring in T.
- τ_u: the inter-update interval of the presence information.
- τ: the interval between the first update message and the last update message occurring in T.
- τ_a: the inter-access interval of the watcher.
- $1/\lambda = E[\tau_u]$: mean inter-update interval.
- $1/\mu = E[\tau_a]$: mean inter-access interval.
- $1/\gamma = E[T]$: mean delayed threshold.

- V_a: the variance for the τ_a distribution.
- V_u: the variance for the τ_u distribution.
- $f_u(\cdot)$: density function for the τ_u distribution.
- $f_u^*(\cdot)$: Laplace transform for the τ_u distribution.

Chapter 4

Derivation for $E[N]$ and p

This chapter describes an analytic model for deriving $E[N]$ and p. If the inter-update interval τ_u has the exponential distribution with the mean $1/\lambda$, then due to the Poisson property of the update arrivals, for a delayed threshold T with an arbitrary distribution with the mean $1/\gamma$, the expected number $E[N]$ of update messages occurring in this period is

$$E[N] = \lambda E[T] = \frac{\lambda}{\gamma} \quad (4.1)$$

If τ_u is not exponentially distributed, (4.1) does not hold, and $E[N]$ is derived as follows. Suppose that T is exponentially distributed, and τ_u has an arbitrary distribution. Assume that the first update message in an observation interval occurs at t_0, and the ith subsequent update message occurs at t_i. Let $\tau_{u,i} = t_i - t_{i-1}$ for $i \geq 1$. Define $t_{u,n} = \sum_{i=1}^{n} \tau_{u,i}$. Let $\Pr[N = n]$ be the probability that there are n update messages occurring in T (excluding the update at t_0). Then

$$\Pr[N = n] = \Pr[t_{u,n} < T < t_{u,n+1}] \quad (4.2)$$

For exponential delayed threshold,

$$\begin{aligned}
\Pr[N=n] &= \int_{\tau_{u,1}=0}^{\infty} f_u(\tau_{u,1}) \cdots \int_{\tau_{u,n}=0}^{\infty} f_u(\tau_{u,n}) \int_{\tau_{u,n+1}=0}^{\infty} f_u(\tau_{u,n+1}) \\
&\quad \times \int_{T=t_{u,n}}^{t_{u,n+1}} \gamma e^{-\gamma T} dT d\tau_{u,n+1} d\tau_{u,n} \cdots d\tau_{u,1} \\
&= \int_{\tau_{u,1}=0}^{\infty} f_u(\tau_{u,1}) e^{-\gamma \tau_{u,1}} \cdots \int_{\tau_{u,n}=0}^{\infty} f_u(\tau_{u,n}) e^{-\gamma \tau_{u,n}} \\
&\quad \times \int_{\tau_{u,n+1}=0}^{\infty} f_u(\tau_{u,n+1}) \left(1 - e^{-\gamma \tau_{u,n+1}}\right) d\tau_{u,n+1} d\tau_{u,n} \cdots d\tau_{u,1} \\
&= \left[f_u^*(\gamma)\right]^n \left[1 - f_u^*(\gamma)\right] \qquad (4.3)
\end{aligned}$$

From (4.3), $E[N]$ is derived as

$$\begin{aligned}
E[N] &= \sum_{n=0}^{\infty} \{n \Pr[N=n]\} \\
&= \sum_{n=1}^{\infty} \left\{n \left[f_u^*(\gamma)\right]^n \left[1 - f_u^*(\gamma)\right]\right\} \\
&= \frac{f_u^*(\gamma)}{1 - f_u^*(\gamma)} \qquad (4.4)
\end{aligned}$$

If τ_u has the Gamma distribution with the mean $1/\lambda$ and the variance V_u, then its Laplace transform is

$$f_u^*(s) = \left(\frac{1}{V_u \lambda s + 1}\right)^{\frac{1}{V_u \lambda^2}} \qquad (4.5)$$

It has been shown that the distribution of any positive random variable can be approximated by a mixture of Gamma distributions [6], and is therefore selected in our study to represent the inter-update interval distribution. Substitute (4.5) into (4.4) to yield

$$E[N] = \frac{1}{(V_u \lambda \gamma + 1)^{\frac{1}{V_u \lambda^2}} - 1} \qquad (4.6)$$

Following the past experience [5, 15, 17, 18], we can measure the inter-update intervals of the presence information in a real mobile network, and generate the Gamma distribution from the measured data. Then (4.6) can be used to compute $E[N]$.

Next, we derive p. Note that probability $1-p$ is proportional to the expected delayed threshold $E[T]$, and is inversely proportional to the expected observation interval $(1+E[N])E[\tau_u]$ [14]. Therefore, p can be expressed as

$$p = 1 - \frac{E[T]}{(1+E[N])E[\tau_u]} \qquad (4.7)$$

When τ_u is exponentially distributed, (4.7) is re-written as

$$p = \frac{\gamma}{\lambda + \gamma} \tag{4.8}$$

In Appendix A, we also derive p by actually using the exponential τ_u distribution for fixed and exponential delayed thresholds. The results are the same as (4.8). When the inter-update interval τ_u is not exponentially distributed, (4.8) does not hold, and its variance V_u has significant impact on p. Assume that T is exponentially distributed and τ_u has an arbitrary distribution, then from (4.4) and (4.7)

$$p = \frac{\gamma - \lambda}{\gamma} + \left(\frac{\lambda}{\gamma}\right) f_u^*(\gamma) \tag{4.9}$$

If τ_u has the Gamma distribution with the mean $1/\lambda$ and the variance V_u, then (4.9) is re-written as

$$p = \frac{\gamma - \lambda}{\gamma} + \left(\frac{\lambda}{\gamma}\right)\left(\frac{1}{V_u \lambda \gamma + 1}\right)^{\frac{1}{V_u \lambda^2}} \tag{4.10}$$

Chapter 5

Simulation Validation

The purpose of the analytic model in Chapter 4 is two folds. First, it provides mean value analysis ((4.1) and (4.8)) to show the trends of the impacts of λ and γ on the output measures. Second, the analytic model is used to validate the discrete simulation experiments. The discrete event simulation model is described in Appendix B. We validate the $E[N]$ values of the simulation experiments for the exponential τ_u and arbitrary T distributions (by using (4.1)), and the Gamma τ_u and exponential T distributions (by using (4.6)). We validate the p values of the simulation experiments for the exponential τ_u and arbitrary T distributions (by using (4.8)), and the Gamma τ_u and exponential T distributions (by using (4.10)). In this chapter, we assume that τ_a is exponentially distributed with the mean $1/\mu$. In Chapter 6, we will investigate the relationship between p and the distribution of the inter-access interval τ_a. Tables 5.1 and 5.2 compare p and $E[N]$ values for the analytic and simulation results under different μ and V_u values. Table 5.1 lists the errors between the analytic and simulation results, where $\lambda = \gamma$, $V_a = 1/\mu^2$, and $V_u = 1/\lambda^2$ with various μ values. The table indicates that the errors between the analytic and simulation results are within 0.5%. Table 5.2 lists the errors between the analytic and simulation results, where $\lambda = \gamma = \mu$ and $V_a = 1/\mu^2$ with various V_u values. For fixed T, we can not derive analytic equation for $E[N]$. We use the $E[N]$ values from simulation to compute (4.7), which is consistent with the p values directly obtained from the simulation experiments. The table shows that the errors between the analytic and simulation results are within 0.3%. Therefore, the analytic analysis is consistent with the simulation results.

Table 5.1: Comparison of Analytic and Simulation Results under Different μ Values ($\lambda = \gamma$, $V_a = 1/\mu^2$, and $V_u = 1/\lambda^2$)

Fixed delayed threshold			
μ	0.1γ	γ	10γ
p (Analytic)	0.5	0.5	0.5
p (Simulation)	0.500146	0.499889	0.500036
Error	0.0292%	0.0222%	0.0071%
$E[N]$ (Analytic)	1	1	1
$E[N]$ (Simulation)	0.999317	1.00076	1.00233
Error	0.0682%	0.0759%	0.2326%
Exponential delayed threshold			
μ	0.1γ	γ	10γ
p (Analytic)	0.5	0.5	0.5
p (Simulation)	0.499937	0.499549	0.500286
Error	0.0126%	0.0902%	0.0571%
$E[N]$ (Analytic)	1	1	1
$E[N]$ (Simulation)	0.999864	1.00187	0.994636
Error	0.0135%	0.1867%	0.5364%

Table 5.2: Comparison of Analytic and Simulation Results under Different V_u Values ($\lambda = \gamma = \mu$ and $V_a = 1/\mu^2$)

Fixed delayed threshold				
V_u	$0.001/\lambda^2$	$0.1/\lambda^2$	$10/\lambda^2$	$1000/\lambda^2$
p (Analytic + Simulation)	0.335125	0.35292	0.798605	0.993404
p (Simulation)	0.334833	0.352609	0.799029	0.993215
Error	0.0871%	0.0882%	0.0531%	0.0189%
Exponential delayed threshold				
V_u	$0.001/\lambda^2$	$0.1/\lambda^2$	$10/\lambda^2$	$1000/\lambda^2$
p (Analytic)	0.368063	0.385543	0.786793	0.993115
p (Simulation)	0.367212	0.385587	0.785982	0.993004
Error	0.2312%	0.0113%	0.1031%	0.0111%
$E[N]$ (Analytic)	0.582437	0.627454	3.69029	144.244
$E[N]$ (Simulation)	0.58043	0.627413	3.69701	144.723
Error	0.3445%	0.0064%	0.1822%	0.3319%

Chapter 6

Numerical Examples

This chapter investigates the performance of delayed update.

Effects of γ/λ and V_u on probability p: Figure 6.1 plots p against $\gamma = \frac{1}{E[T]}$ (normalized by λ) and the variance V_u of the inter-update intervals. From the mean value analysis (4.8), it is clear that p increases as γ/λ increases (i.e., the delayed threshold T decreases). When $V_u \leq \frac{1}{\lambda^2}$, p is not sensitive to the change of γ/λ. This figure also indicates that p increases as V_u increases and large p is observed. This phenomenon is explained as follows. When the inter-update interval τ_u becomes more irregular (i.e., V_u increases), more long and short inter-update intervals are observed. Since access events are more likely to fall in long inter-update intervals and are not in T, larger p is observed. When V_u is very large, p is not sensitive to the γ/λ values, and large p is always observed.

Effects of γ/λ and V_u on $E[N]$: Figure 6.2 plots $E[N]$ against γ/λ and V_u. From the mean value analysis (4.1), it is clear that $E[N]$ increases as γ/λ decreases. This figure also indicates that $E[N]$ increases as V_u increases. For a fixed $E[\tau_u]$ value, when V_u increases, there are more short τ_u periods than long τ_u periods. Therefore, it is likely to find many consecutive update messages occurring in T (i.e., larger $E[N]$). When V_u is very small, $E[N]$ is not sensitive to the V_u values.

Effect of the T distribution on p and $E[N]$: Figure 6.1 indicates that the exponential delayed threshold is better than the fixed delayed threshold when V_u is small. On the

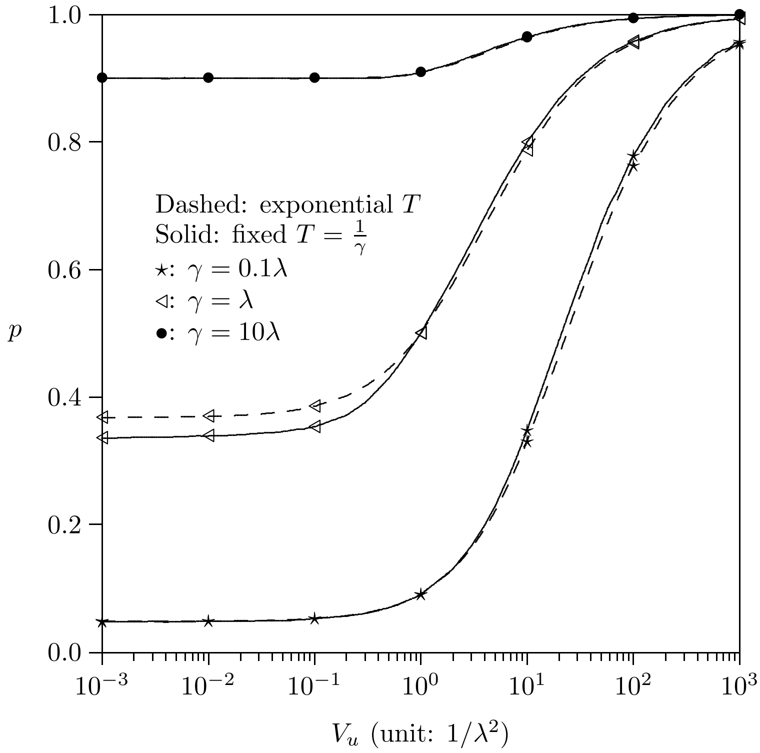

Figure 6.1: Effects of γ/λ and V_u on p

other hand, the fixed delayed threshold is slightly better than the exponential delayed threshold when V_u is large. For the $E[N]$ performance, Figure 6.2 indicates that the fixed delayed threshold slightly outperforms the exponential delayed threshold when V_u is large.

Effect of the τ_a distribution on p: Simulation experiments indicate that the distribution of the inter-access interval τ_a does not affect p. Assume that τ_u has the Gamma distribution with the mean $1/\lambda$ and the variance V_u. We use the Gamma, Weibull, and lognormal distributions for τ_a with the mean $1/\mu$ and the variance V_a. Table 6.1 shows the effect of the τ_a distributions on p, where $\lambda = \gamma$ and $V_u = 100/\lambda^2$. From this table, p is independent of the τ_a distributions.

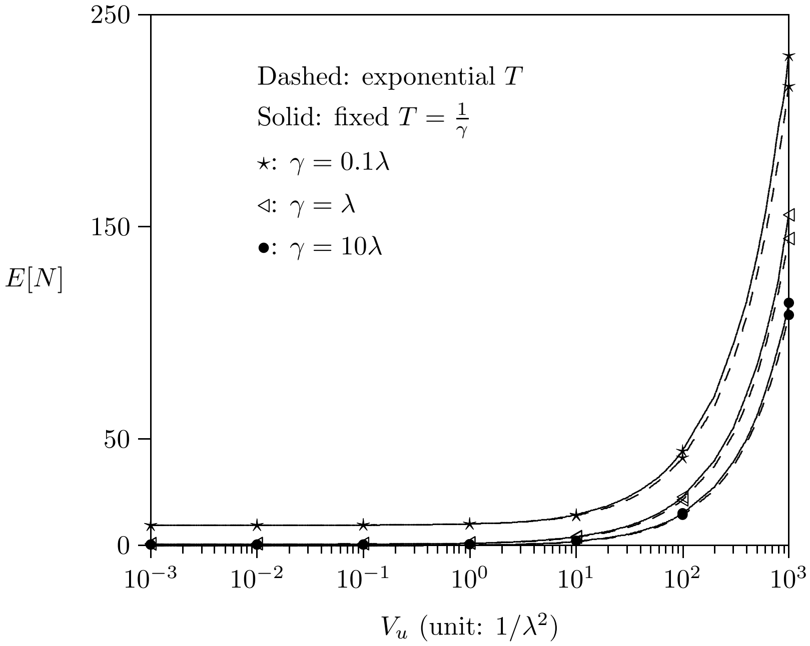

Figure 6.2: Effects of γ/λ and V_u on $E[N]$

Table 6.1: Effect of the τ_a distribution on p ($\lambda = \gamma$ and $V_u = 100/\lambda^2$)

Gamma distribution				
V_a	$0.001/\mu^2$	$0.1/\mu^2$	$10/\mu^2$	$1000/\mu^2$
p (fixed delayed threshold)	0.958406	0.957799	0.958335	0.957274
p (exponential delayed threshold)	0.955652	0.955313	0.953606	0.955345
Weibull distribution				
V_a	$0.001/\mu^2$	$0.1/\mu^2$	$10/\mu^2$	$1000/\mu^2$
p (fixed delayed threshold)	0.957609	0.957626	0.957932	0.957613
p (exponential delayed threshold)	0.954867	0.954526	0.954771	0.954768
Lognormal distribution				
V_a	$0.001/\mu^2$	$0.1/\mu^2$	$10/\mu^2$	$1000/\mu^2$
p (fixed delayed threshold)	0.957667	0.957586	0.957652	0.957407
p (exponential delayed threshold)	0.954767	0.954666	0.954458	0.954906

Chapter 7

Conclusions

This chapter describes our contribution and future work. This thesis investigated the performance of delayed update for IMS presence service. Both the fixed and the exponential delayed thresholds are considered in our study. The performance is measured by the valid access probability p and the expected number $E[N]$ of the update messages occurring in T.

7.1 Contribution

Our study indicated that delayed update can effectively improve p and $E[N]$ when the variance V_u of the inter-update interval is large (when the update behavior is irregular). Furthermore, it is appropriate to select the exponential delayed threshold when V_u is small, and the fixed delayed threshold should be selected when V_u is large.

We also note that the amount of notification traffic to be reduced heavily depends on the operation status of the mobile networks, and is determined in network planning of mobile operators. On the other hand, the valid access probability p is determined by the *Quality of Service* (QoS) policy. By considering both network planning and the QoS policy, a Taiwan's mobile operator, for example, decides that it is acceptable if inaccuracy of the presence information caused by delayed update is less than 10%. For a VIP subscriber, $T = 0$ (delayed update is not exercised). For a flat-rate subscriber, T is set such that p is around 90%-95%. Also, in a heterogeneous wireless network environment (e.g., in the Taipei city), delayed update is triggered through tier-switch (from WiFi to 2.5G/3G). When a subscriber is connected to WiFi (where wireless resources are abundant), delayed update is not exercised ($T = 0$). When the subscriber is switched from WiFi to 2.5G/3G (with higher wireless cost), T is set such that p is around 90%-95%.

7.2 Future Work

In this thesis, we implemented an IMPS system based on 3GPP standards. Because the *Global Positioning System* (GPS)-enabled mobile phones become more popular in recent years, we can enhance the IMPS client with the GPS receiver to obtain the location information. Therefore the presence server can provide the location-based services (e.g., mobile advertising and traffic jam warning). However, due to the UE's mobility, the IMPS client may update the location information too frequently, which wastes the limited radio resources. Therefore, we will investigate how to maintain the accuracy of the UE's position with less bandwidth consumption.

Bibliography

[1] 3GPP. 3rd Generation Partnership Project; Technical Specification Group Services and System Aspects; Presence Service; Architecture and functional description. Technical Specification 3G TS 23.141 version 8.1.0 (2008-06), 2008.

[2] 3GPP. 3rd Generation Partnership Project; Technical Specification Group Services and System Aspects; IP Multimedia Subsystem (IMS); Stage 2. Technical Specification 3G TS 23.228 version 8.5.0 (2008-06), 2008.

[3] 3GPP. 3rd Generation Partnership Project; Technical Specification Group Core Network and Terminals; Presence service using the IP Multimedia (IM) Core Network (CN) subsystem; Stage 3. Technical Specification 3G TS 24.141 version 8.1.0 (2008-06), 2008.

[4] 3GPP. 3rd Generation Partnership Project; Technical Specification Group Core Network and Terminals; IP multimedia call control protocol based on Session Initiation Protocol (SIP) and Session Description Protocol (SDP); Stage 3. Technical Specification 3G TS 24.229 version 8.4.1 (2008-06), 2008.

[5] Escalle, P.G., Giner, V.C., and Oltra, J.M. Reducing Location Update and Paging Costs in a PCS Network, *IEEE Transactions on Wireless Communications*, vol. 1, no. 1, pp. 200-209, January 2002.

[6] Kelly, F.P. *Reversibility and Stochastic Networks*. John Wiley & Sons, Inc., 1979.

[7] Lin, Y.-B., Lai, W.-R., and Chen, J.-J. Effects of cache mechanism on wireless data access, *IEEE Transactions on Wireless Communications*, vol. 2, no. 6, pp. 1247-1258, November 2003.

[8] Lin, Y.-B. and Pang, A.-C. *Wireless and Mobile All-IP Networks*. John Wiley & Sons, Inc., 2005.

[9] Liou, R.-H., and Chen, W.-E. Message Reducing Mechanisms for Presence Service in 3GPP IP Multimedia Subsystem (IMS), *The 13th Mobile Computing Workshop*, April 2007.

[10] Niemi, A. Session Initiation Protocol (SIP) Extension for Event State Publication. RFC 3903, IETF, October 2004.

[11] Pang, A.-C., Liu, C.-H., Liu, S.P., and Hung, H.-N. A Study on SIP Session Timer for Wireless VoIP, *IEEE Wireless Communications and Networking Conference (WCNC)*, vol. 4, pp. 2306-2311, March 2005.

[12] Roach, A. B. Session Initiation Protocol (SIP)-Specific Event Notification. RFC 3265, IETF, June 2002.

[13] Rosenberg, J. A Presence Event Package for the Session Initiation Protocol (SIP). RFC 3856, IETF, August 2004.

[14] Ross, S.M. *Stochastic Processes*. John Wiley & Sons, Inc., 1996.

[15] Sou, S.-I. and Lin, Y.-B. Modeling Mobility Database Failure Restoration using Checkpoint Schemes. *IEEE Transactions on Wireless Communications*, Vol. 6, No. 1, pp. 313-319, January 2007.

[16] Wang, T.-P. and Chiu, K. An Efficient Scheme for Supporting Personal Mobility in SIP-based VoIP Services, *IEICE Transactions on Communications (Special Section on Mobile Multimedia Communications)*, Vol. E89-B, No. 10, pp. 2706-2714, October 2006.

[17] Yang, S.-R. Dynamic Power Saving Mechanism for 3G UMTS System, *ACM/Springer Mobile Networks and Applications (MONET)*, Vol. 12, No. 1, pp. 5-14, January 2007.

[18] Yang, S.-R., Lin, P., and Huang, P.-T. Modeling Power Saving for GAN and UMTS Interworking. Accepted for Publication in *IEEE Transactions on Wireless Communications.*

Appendix A

An Alternative Derivation for p

This appendix shows an alternative for deriving p to double check that (4.8) is correct. To compute p, two cases are considered.

Case I. Under the condition that no update message occurs in T, the expected delayed threshold and the expected observation interval are $E[T|T < \tau_u]$ and $E[\tau_u|T < \tau_u]$, respectively.

Case II. Under the condition that one or more update messages occur in T, the expected delayed threshold and the expected observation interval are $E[T|\tau < T < \tau + \tau_u]$ and $E[\tau + \tau_u|\tau < T < \tau + \tau_u]$, where τ is the interval between the first update message and the last update message occurring in T.

We note that probability $1 - p$ is proportional to the expected delayed threshold $E[T]$, and is inversely proportional to the expected observation interval. Therefore, p can be expressed as

$$p = 1 - \frac{E[T|T < \tau_u]\Pr[T < \tau_u] + E[T|\tau < T < \tau + \tau_u]\Pr[\tau < T < \tau + \tau_u]}{E[\tau_u|T < \tau_u]\Pr[T < \tau_u] + E[\tau + \tau_u|\tau < T < \tau + \tau_u]\Pr[\tau < T < \tau + \tau_u]} \quad \text{(A.1)}$$

Since $E[X|Y] = E[X \,\&\, Y]/\Pr[Y]$, (A.1) is re-written as

$$p = 1 - \frac{E[T \,\&\, T < \tau_u] + E[T \,\&\, \tau < T < \tau + \tau_u]}{E[\tau_u \,\&\, T < \tau_u] + E[\tau + \tau_u \,\&\, \tau < T < \tau + \tau_u]} \quad \text{(A.2)}$$

Based on (A.2), we derive p for fixed and exponential T assuming that τ_u is exponentially distributed.

A.1 Fixed Delayed Threshold

When T is fixed, we have

$$\begin{aligned} E[T \ \& \ T < \tau_u] &= \int_{\tau_u=T}^{\infty} T\lambda e^{-\lambda\tau_u} d\tau_u \\ &= \left(\frac{1}{\gamma}\right) e^{-\frac{\lambda}{\gamma}} \end{aligned} \tag{A.3}$$

$$\begin{aligned} E[T \ \& \ \tau < T < \tau+\tau_u] &= \sum_{N=1}^{\infty} \left\{ \int_{\tau=0}^{T} T \left[\frac{(\lambda\tau)^{N-1}}{(N-1)!}\right] \lambda e^{-\lambda\tau} \int_{\tau_u=T-\tau}^{\infty} \lambda e^{-\lambda\tau_u} d\tau_u d\tau \right\} \\ &= Te^{-\lambda T} \sum_{N=1}^{\infty} \left\{ \int_{\tau=0}^{T} \left[\frac{(\lambda\tau)^{N-1}}{(N-1)!}\right] \lambda d\tau \right\} \\ &= Te^{-\lambda T} \int_{\tau=0}^{T} e^{\lambda\tau} \lambda d\tau \\ &= \left(\frac{1}{\gamma}\right) (1 - e^{-\frac{\lambda}{\gamma}}) \end{aligned} \tag{A.4}$$

$$\begin{aligned} E[\tau_u \ \& \ T < \tau_u] &= \int_{\tau_u=T}^{\infty} \tau_u \lambda e^{-\lambda\tau_u} d\tau_u \\ &= \left(\frac{1}{\gamma} + \frac{1}{\lambda}\right) e^{-\frac{\lambda}{\gamma}} \end{aligned} \tag{A.5}$$

and

$$\begin{aligned} E\left[\begin{array}{c} \tau+\tau_u \ \& \\ \tau < T < \tau+\tau_u \end{array}\right] &= \sum_{N=1}^{\infty} \left\{ \int_{\tau=0}^{T} \left[\frac{(\lambda\tau)^{N-1}}{(N-1)!}\right] \lambda e^{-\lambda\tau} \int_{\tau_u=T-\tau}^{\infty} (\tau+\tau_u)\lambda e^{-\lambda\tau_u} d\tau_u d\tau \right\} \\ &= \left(T + \frac{1}{\lambda}\right) e^{-\lambda T} \int_{\tau=0}^{T} e^{\lambda\tau} \lambda d\tau \\ &= \left(\frac{1}{\gamma} + \frac{1}{\lambda}\right) (1 - e^{-\frac{\lambda}{\gamma}}) \end{aligned} \tag{A.6}$$

From (A.3), (A.4), (A.5), and (A.6), (A.2) is re-written as

$$p = \frac{\gamma}{\lambda+\gamma} \tag{A.7}$$

A.2 Exponential Delayed Threshold

If T has the exponential distribution with the mean $1/\gamma$, then

$$\begin{aligned} E[T \ \& \ T < \tau_u] &= \int_{T=0}^{\infty} T\gamma e^{-\gamma T} \int_{\tau_u=T}^{\infty} \lambda e^{-\lambda\tau_u} d\tau_u dT \\ &= \frac{\gamma}{(\lambda+\gamma)^2} \end{aligned} \tag{A.8}$$

$$E\left[\begin{array}{c} T \ \& \\ \tau < T < \tau + \tau_u \end{array}\right] = \sum_{N=1}^{\infty}\left\{\int_{T=0}^{\infty} T\gamma e^{-\gamma T}\int_{\tau=0}^{T}\left[\frac{(\lambda\tau)^{N-1}}{(N-1)!}\right]\lambda e^{-\lambda\tau}\int_{\tau_u=T-\tau}^{\infty}\lambda e^{-\lambda\tau_u}d\tau_u d\tau dT\right\}$$

$$= \frac{1}{\gamma} - \frac{\gamma}{(\lambda+\gamma)^2} \quad (A.9)$$

$$E[\tau_u \ \& \ T < \tau_u] = \int_{T=0}^{\infty}\gamma e^{-\gamma T}\int_{\tau_u=T}^{\infty}\tau_u\lambda e^{-\lambda\tau_u}d\tau_u dT$$

$$= \frac{\gamma}{(\lambda+\gamma)^2} + \frac{\gamma}{\lambda(\lambda+\gamma)} \quad (A.10)$$

and

$$E\left[\begin{array}{c} \tau + \tau_u \ \& \\ \tau < T < \tau + \tau_u \end{array}\right] = \sum_{N=1}^{\infty}\left\{\int_{T=0}^{\infty}\gamma e^{-\gamma T}\int_{\tau=0}^{T}\left[\frac{(\lambda\tau)^{N-1}}{(N-1)!}\right]\lambda e^{-\lambda\tau}\int_{\tau_u=T-\tau}^{\infty}(\tau+\tau_u)\lambda e^{-\lambda\tau_u}d\tau_u d\tau dT\right\}$$

$$= \frac{1}{\gamma} + \frac{1}{\lambda} - \frac{\gamma}{(\lambda+\gamma)^2} - \frac{\gamma}{\lambda(\lambda+\gamma)} \quad (A.11)$$

From (A.8), (A.9), (A.10), and (A.11), (A.2) is re-written as

$$p = \frac{\gamma}{\lambda+\gamma} \quad (A.12)$$

Both (A.7) and (A.12) indicate that p for fixed delayed threshold is the same as that for exponential delayed threshold.

Appendix B

Simulation Model

This appendix describes the discrete event simulation model for delayed update to validate against the proposed analytic model. The following attributes are defined for an event e in the simulation model:

- The *type* attribute indicates the event type. An **Access** event represents that the watcher accesses the presence information. A **Notification** event represents that the presence server notifies the watcher of the presence information update. An **Update** event represents that the presence information is updated.
- The ts attribute indicates the timestamp when the event occurs.

The inter-arrival period τ_a between two consecutive **Access** events e_1 and e_2 is a random number drawn from a random number generator G_A, where $e_2.ts = e_1.ts + \tau_a$. The delayed threshold T is a random number drawn from a random number generator G_N. The inter-arrival period τ_u between two consecutive **Update** events e_1 and e_2 is a random number drawn from a random number generator G_U, where $e_2.ts = e_1.ts + \tau_u$. In this simulation model, arbitrary distributions of the random number generators can be used to generate τ_a, T, and τ_u. In our numerical example, G_A is the Gamma, Weibull, or lognormal random number generator with mean $1/\mu$ and the variance V_a, G_N is a fixed or an exponential random number generator with mean $1/\gamma$, and G_U is a Gamma random number generator with mean $1/\lambda$ and variance V_u. A flag $Valid$ is used in the simulation to indicate if the presence information is valid. The output measures of the simulation are listed below:

- N_a: the number of the presence information accesses.
- N_i: the number of the invalid presence information accesses.
- N_n: the number of the notifications.
- N_u: the number of the updates occurring in all T periods in the simulation run.

From the above output measures, we compute

$$p = 1 - \frac{N_i}{N_a} \text{ and } E[N] = \frac{N_u}{N_n} \tag{B.1}$$

In every simulation run, a simulation clock ck is maintained to indicate the simulation progress, which is the timestamp of the event being processed. All events are inserted into the event list, and are deleted/processed from the event list in the non-decreasing timestamp order. Figure B.1 illustrates the simulation flow chart for delayed update with the following steps:

Step 1: The simulation clock ck is set to 0, the output measures are initialized to 0, and the flag $Valid$ is set to $TRUE$.

Step 2: The first **Access** event e_1 and **Update** event e_2 are generated, where $e_1.ts = ck + \tau_a$ and $e_2.ts = ck + \tau_u$. These events are inserted into the event list.

Step 3: The first event e in the event list is deleted, and is processed based on its type at Step 4. The clock ck is set to $e.ts$.

Step 4: If $e.type$ is **Access**, then Step 5 is executed. If $e.type$ is **Notification**, the simulation flow proceeds to Step 11. Otherwise, $e.type$ is **Update**, and the flow goes to Step 13.

Steps 5-7: N_a is incremented by one at Step 5. If $Valid = TRUE$ at Step 6, then Step 8 is executed. Otherwise N_i is incremented by one at Step 7.

Step 8: If one millions of **Access** events have been processed, then Step 9 is executed. Otherwise, the simulation proceeds to Step 10.

Step 9: Output measures are computed, and the simulation terminates.

Step 10: The next **Access** event e_1 is generated, and is inserted into the event list, where $e_1.ts = ck + \tau_a$. The simulation flow proceeds to Step 3.

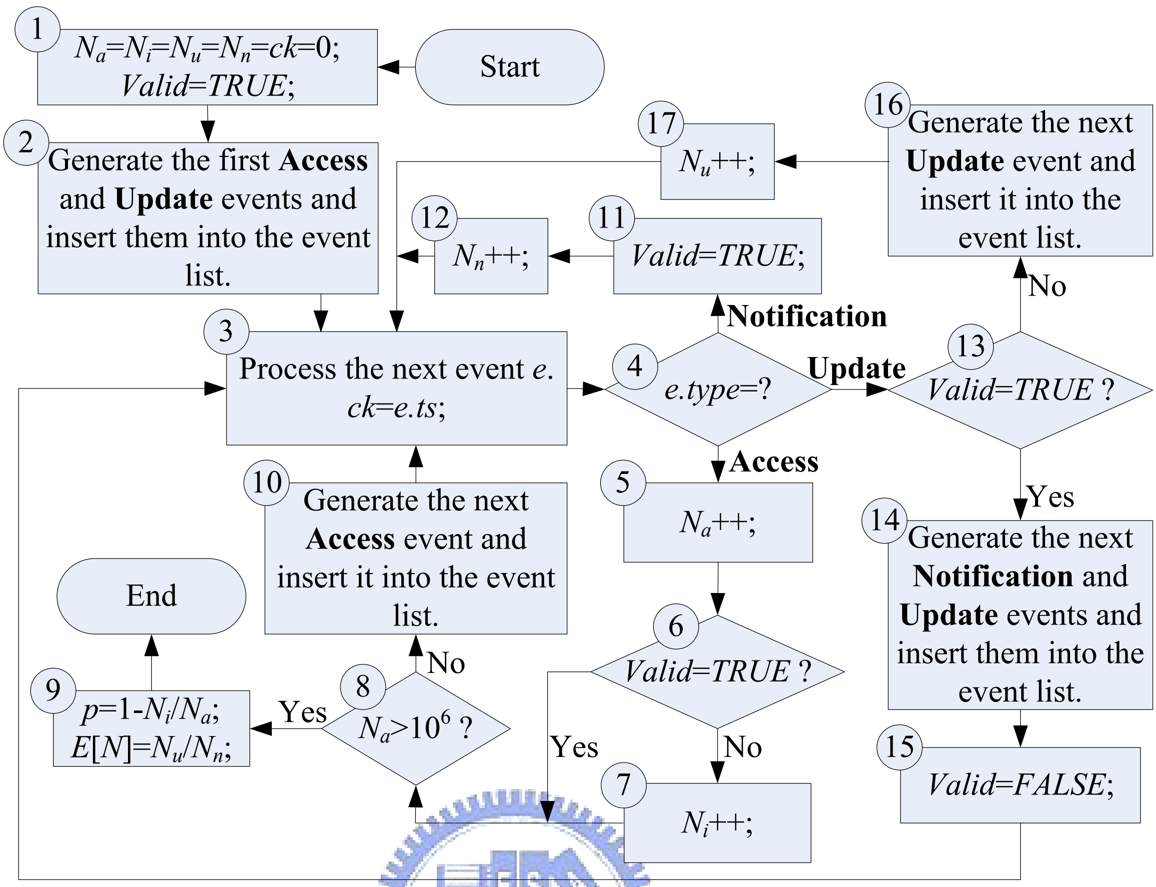

Figure B.1: Simulation Flow Chart for Delayed Update

Steps 11-12: $Valid$ is set to $TRUE$ at Step 11, N_n is incremented by one at Step 12, and the flow goes to Step 3.

Step 13: If $Valid = TRUE$, then Step 14 is executed. Otherwise Step 16 is processed.

Steps 14-15: Step 14 generates the next **Notification** event e_1 and **Update** event e_2, and inserts them into the event list, where $e_1.ts = ck + T$ and $e_2.ts = ck + \tau_u$. At Step 15, the flag $Valid$ is set to $FALSE$, and the flow goes to Step 3.

Steps 16-17: The next **Update** event e_1 is generated, and is inserted into the event list at Step 16, where $e_1.ts = ck + \tau_u$. Then N_u is incremented by one at Step 17. The simulation flow jumps to Step 3.

Appendix C

The Simulation Program

The simulation codes for Figure B.1 is listed in this appendix. The library of random number generator is proprietary and is not listed.

```
#include <iostream>
#include <event.h>
#include <e_list.h>
#include <random_2.h>
#include <random_tsaimh.h>
using namespace std;

/*Event type*/
enum
{
  ACCESS,
  NOTIFICATION,
  UPDATE
};
/**
 * compute_p_and_expected_N: compute probability p and
 *                           expected N value.
 * 1/lambda: mean inter-update interval
 * 1/gamma: mean mean delayed threshold
 * 1/mu: mean inter-access interval
 * Va: the variance for the tau_a distribution
 * Vu: the variance for the tau_u distribution.
```

```
 * if use_exp_T==true, then the delayed threshold T has the
 * exponential distribution with the mean 1/gamma. Otherwise,
 * T is fixed.
 */
void compute_p_and_expected_N(double lambda, double gamma,
         double mu, double Va, double Vu, bool use_exp_T);

int main(int argc, char* argv[])
{
  if(argc!=6)
  {
    cout<<"Usage: "<<argv[0]<<" <lambda> <gamma> <mu>
                   <Va> <Vu>"<<endl;
    return 0;
  }
  compute_p_and_expected_N(atof(argv[1]),atof(argv[2]),
          atof(argv[3]),atof(argv[4]),atof(argv[5]),false);
  compute_p_and_expected_N(atof(argv[1]),atof(argv[2]),
          atof(argv[3]),atof(argv[4]),atof(argv[5]),true);
  return 0;
}
void compute_p_and_expected_N(double lambda, double gamma,
          double mu, double Va, double Vu, bool use_exp_T)
{
  /**
   * Na: the number of the presence information accesses
   * Ni: the number of the invalid presence information
   *     accesses
   * Nn: the number of the notifications
   * Nu: the number of the updates occurring in all T periods
   * in the simulation run
   * ck: a simulation clock
  */
  /*initialization*/
  double interarrival_time;
  double Na=0, Ni=0, Nn=0, Nu=0, ck=0;
  bool Valid=true;

```

```
/**
 * tau_a: the inter-access interval of the watcher
 * tau_u: the inter-update interval of the presence
 *         information.
 * T: the delayed threshold
 */
Gamma tau_a(1/mu,Va);
Gamma tau_u(1/lambda,Vu);
Expon T(1/gamma);

E_List* E_List_ptr=new E_List;
Event *e,*e1,*e2;

/**
 * The first Access event e1 and Update event e2 are
 * generated, where e1.ts = ck+tau_a and
 * e2.ts = ck + tau_u. These events are inserted into
 * the event list.
 */
e1 = new Event();
interarrival_time = tau_a++;
e1 -> setTimeStamp(ck + interarrival_time);
e1 -> setEventType(ACCESS);
*E_List_ptr << *e1 ;

e2 = new Event();
interarrival_time = tau_u++;
e2 -> setTimeStamp(ck + interarrival_time);
e2 -> setEventType(UPDATE);
*E_List_ptr << *e2 ;

while(true)
{
  /**
   * The first event e in the event list is deleted, and is
   * processed based on its type.
   * The clock ck is set to e.ts.
   */
```

```
*E_List_ptr >> e;
ck = e -> getTimeStamp() ;

switch(e->getEventType())
{
case ACCESS:
  /* Na  is incremented by one */
  Na++;
  /*if Valid==false, Ni is incremented by one*/
  if(Valid!=true)
  {
    Ni++;
  }
  /**
   * If one millions of Access events have been processed,
   * the simulation terminates
   */
  if(Na>1000000)
  {
    break;
  }
  /**
   * The next Access event e1 is generated, and is
   * inserted into the event list, where
   * e1.ts = ck + tau_a.
   */
  e1 = new Event();
  interarrival_time = tau_a++;
  e1 -> setTimeStamp(ck + interarrival_time);
  e1 -> setEventType(ACCESS);
  *E_List_ptr << *e1 ;
  break;
case NOTIFICATION:
  /* Valid is set to true, and Nn is incremented by one */
  Valid=true;
  Nn++;
  break;
case UPDATE:
```

```
if(Valid==true)
{
  /**
   * Generates the next Notification event e1 and Update
   * event e2, and inserts them into the event list,
   * where e1.ts = ck + T and e2.ts = ck + tau_u.
   */
  e1 = new Event();
  if(use_exp_T==true)
  {
    interarrival_time = T++;
  }
  else
  {
    interarrival_time=1/gamma;
  }
  e1 -> setTimeStamp(ck + interarrival_time);
  e1 -> setEventType(NOTIFICATION);
  *E_List_ptr << *e1 ;

  e2 = new Event();
  interarrival_time = tau_u++;
  e2 -> setTimeStamp(ck + interarrival_time);
  e2 -> setEventType(UPDATE);
  *E_List_ptr << *e2 ;
  /*the flag Valid is set to false*/
  Valid=false;
}
else
{
  /**
   * The next Update event e1 is generated, and is
   * inserted into the event list, where
   * e1.ts = ck + tau_u.
  */
  e1 = new Event();
  interarrival_time = tau_u++;
  e1 -> setTimeStamp(ck + interarrival_time);
```

```
        e1 -> setEventType(UPDATE);
        *E_List_ptr << *e1 ;
        /*Nu  is incremented by one*/
        Nu++;
      }
      break;
    }
    delete e;
    /**
     * If one millions of Access events have been processed,
     * the simulation terminates
     */
    if(Na>1000000)
    {
      break;
    }
  }
  delete E_List_ptr;

  /* Output measures are computed*/
  double p=1-Ni/Na;
  double Expected_N=Nu/Nn;
  if(use_exp_T==true)
  {
    cout<<"For exponential delayed threshold"<<endl;
  }
  else
  {
    cout<<"For fixed delayed threshold"<<endl;
  }
  cout<<"p:"<<p<<endl;
  cout<<"E[N]:"<<Expected_N<<endl;
}
```