Per-User Checkpointing for Mobility Database Failure Restoration

Yi-Bing Lin, *Fellow*, *IEEE*

Abstract—This paper studies the failure restoration of mobility database for *Universal Mobile Telecommunications System* (UMTS). We consider a per-user checkpointing approach for the *Home Location Register* (HLR) database. In this approach, individual HLR records are saved into a backup database from time to time. When a failure occurs, the backup record is restored back to the mobility database. We first describe a commonly used basic checkpoint algorithm. Then, we propose a new checkpoint algorithm. An analytic model is developed to compare these two algorithms in terms of the checkpoint cost and the probability that a HLR backup record is obsolete. This analytic model is validated against simulation experiments. Numerical examples indicate that our new algorithm may significantly outperform the basic algorithm in terms of both performance measures.

Index Terms—Checkpoint, failure restoration, General Packet Radio Service (GPRS), Home Location Register, Universal Mobile Telecommunications System (UMTS).

1 Introduction

This paper studies failure restoration of mobility databases for *Universal Mobile Telecommunications System* (UMTS) and/or *General Packet Radio Service* (GPRS) [13], [1]. UMTS and GPRS support wireless Internet applications [2]. In these networks, the *Home Location Register* (HLR) is a database used for mobile user information management. All permanent subscriber data are stored in this database. An HLR record consists of three types of information: **Mobile Station (MS) Information** such as the telephone number and the *International Mobile Subscriber Identity* (used by the MS to access the network), **Service Information** such as service subscription, service restrictions, and supplementary services, and **Location Information** such as the address of the *Serving GPRS Support Node* (SGSN) where the MS resides. The location information in the HLR is updated whenever the MS moves to a new SGSN. To access the MS, the HLR is queried to find the current SGSN location of the MS. Note that both the MS and service information items are only occasionally updated. On the other hand, an MS may move frequently and the location information is often modified. Details of HLR operations due to call delivery can be found in [13].

If the HLR fails, one will not be able to access the MSs. To guarantee service availability to the MSs, database recovery is required after an HLR failure. In UMTS/GPRS [1], the HLR recovery procedure works as follows: The HLR database is periodically checkpointed. After an HLR failure, the database is restored by reloading the backup information. There are several approaches to checkpointing the HLR database. In the *all-record checkpoint* approach, all HLR records are saved into the backup at the same times [8], [5], [11]. The checkpoint overhead for this approach is very high and is typically performed at midnight when the HLR activities are infrequent. Alternatively, checkpointing can be exercised for individual mobile users, which is referred to as *per-user checkpointing* [3], [9], [18], [12], [16]. We describe two algorithms for the per-user checkpoint approach. The first algorithm (referred to as *Algorithm I*) is the same as all-record checkpointing, except that the checkpoint frequencies for individual MSs may be different. The second algorithm (referred to as *Algorithm II*) is a new approach proposed in this paper.

Algorithm I (The Basic Algorithm). For every MS, we define a timeout period t_p. In Fig. 1, the t_p timeouts occur at time t_0, t_1, t_2, t_4, and t_9. When this timer expires, checkpoint is performed to save the HLR record of the MS. Therefore, the checkpoint interval t_c is equal to the timeout period t_p. After a failure (see t_6 in Fig. 1), the HLR record in the backup database is restored to the HLR. The backup copy is *obsolete* if the HLR record is updated between the last checkpoint and when the failure occurs (i.e., a registration occurs in $[t_4, t_6]$ in Fig. 1). After the HLR record is restored, one of the following two events may occur next:

- The record may be updated again if the MS issues a registration (i.e., $t_8 < t_7$ in Fig. 1) or
- the record may be accessed due to an incoming call to the MS (i.e., $t_7 < t_8$ in Fig. 1).

After a failure, if the backup record is obsolete and the next event to the MS is an incoming call ($t_7 < t_8$ in Fig. 1), then the call is lost. On the other hand, if the next event is a registration, then the location information of the HLR record is modified and the record is up to date again.

Algorithm II (Lin's Algorithm). The intuition behind our algorithm is simple: If registration activities are very frequent (i.e., a registration always occurs before the t_p timer expires), then Algorithm II behaves exactly the same as Algorithm I. On the other hand, if no registration has occurred before the t_p timer expires, then there is no need to checkpoint the record (because the backup copy is still valid). In this case, checkpoint is performed when the next registration occurs.

In Algorithm II, a three-state finite state machine (FSM) is implemented for an HLR record. The state diagram for the FSM is shown in Fig. 2. Initially, the FSM is in **state 0**, and

• *The author is with the Deparment of Computer Science and Information Engineering, National Chiao Tung University, Hsinchu, Taiwan, R.O.C. E-mail: liny@csie.nctu.edu.tw.*

Manuscript received 31 Oct. 2003; revised 3 Jan. 2004; accepted 9 Feb. 2004; published online 27 Jan. 2005.
For information on obtaining reprints of this article, please send e-mail to: tmc@computer.org, and reference IEEECS Log Number TMC-0182-1003.

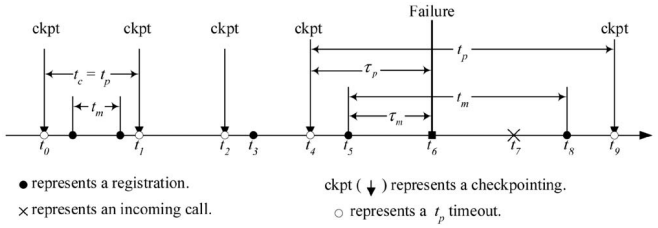

Fig. 1. The timing diagram for Algorithm I.

the t_p timer starts to decrement. If a registration event occurs before the t_p timer expires, the FSM moves to **state 1**, and remains in **state 1** until the t_p timeout event occurs. Then, the FSM moves back to **state 0**, the HLR record is checkpointed into the backup, and the t_p timer is restarted. If the timeout event occurs at **state 0**, then the FSM moves to **state 2**, and the t_p timer is stopped. If a registration event occurs at **state 2**, the FSM moves to **state 0**, a checkpoint is performed, and the t_p timer is restarted.

Consider the timing diagram in Fig. 3. At time t_0, the FSM is at **state 0** (when a registration occurs). At time t_1, the next registration occurs and the FSM moves from **state 0** to **state 1** (where $t_m = t_1 - t_0$ is the interregistration interval). At time t_2, the t_p timer expires and the FSM moves from **state 1** to **state 0** (where $t_c = t_p = t_2 - t_0$). At time t_3, the t_p timer expires again and the FSM moves from **state 0** to **state 2**. At time t_4, a registration occurs. The FSM moves from **state 2** to **state 0**, and $t_c = \tau_m^* = t_4 - t_2$, where τ_m^* is the *excess life* or *residual time* of t_m.

Two output measures are considered to investigate the checkpoint performance:

- $E[t_c]$: the expected checkpoint interval. The larger the $E[t_c]$ value, the lower the checkpoint overhead. That is, the checkpoint cost is proportional to the checkpoint frequency $1/E[t_c]$. We use $E_I[t_c]$ and $E_{II}[t_c]$ to represent the $E[t_c]$ values for Algorithms I and II, respectively.
- α: the probability that the HLR record in the backup is obsolete when a failure occurs. The smaller the α value, the better the checkpoint performance. We use α_I and α_{II} to represent the α values for Algorithms I and II, respectively.
- I_c: Improvement of Algorithm II over Algorithm I in terms of the $E[t_c]$ measure.
- I_α: Improvement of Algorithm II over Algorithm I in terms of the α measure.

In a typical checkpoint approach, fixed t_p is selected [7], [8]. Since many HLR records will be performed per-user checkpointing, Exponential t_p distribution is selected in our study to avoid congestion caused by a large number of simultaneous checkpoints. In [14], we showed that similar performance results were observed for both fixed and Exponential checkpoint approaches. On the other hand, Exponential checkpointing exhibits Exponential backoff property for resolving contention of checkpoint traffic [10]. Such an advantage is not found in the fixed checkpoint approach. The Exponential random variable t_p has the density function

$$f_p(t_p) = \lambda e^{-\lambda t_p}$$

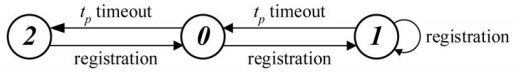

Fig. 2. The state diagram for Algorithm II.

Since $t_c = t_p$ in Algorithm I, the expected checkpoint interval is

$$E_I[t_c] = E[t_p] = \frac{1}{\lambda}. \tag{1}$$

After a failure, the HLR record is restored from the backup. This backup copy is obsolete if the record in the HLR has been modified since last checkpoint. In Fig. 1, a failure occurs at time t_6 which is a random observer of the intercheckpoint interval $[t_4, t_9]$ and the interregistration interval $[t_5, t_8]$. In this figure, the interval $t_m - \tau_m$ is the residual time of t_m, and τ_m is the *reverse residual time*. Similarly, τ_p is the reverse residual time of t_p. Consider a random variable t with the probability density function $f(t)$, the distribution function $F(t) = \int_{y=0}^{t} f(y)dy$, the expected value $E[t]$, and the Laplace transform $f^*(s) = \int_{t=0}^{\infty} f(t)e^{-st}dt$. Let τ be the residual time of t with the density function $r(\tau)$, the probability distribution function $R(\tau)$, and the Laplace transform $r^*(s)$. From [17],

$$r(\tau) = \frac{1 - F(\tau)}{E[t]} \quad \text{and} \quad r^*(s) = \frac{1 - f^*(s)}{E[t]s}. \tag{2}$$

Note that the reverse residual time distribution is the same as the residual time distribution [10], and (2) also holds for the reverse residual time. From (2), the density function $r_p(t)$ is the same as $f_p(t)$ for the Exponential distribution. That is,

$$r_p(t) = f_p(t) = \lambda e^{-\lambda t}.$$

Let α_I be the probability that the HLR backup is obsolete when a failure occurs in Algorithm I. In Fig. 1, the backup copy is obsolete if $t_4 < t_5 < t_6$. Let $\tau_c = \tau_p$ with the density function $r_p(\tau_c)$ be the reverse residual time $t_c = t_p$. Then,

$$\begin{aligned} \alpha_I &= \Pr[\tau_c > \tau_m] \\ &= \int_{\tau_m=0}^{\infty} r_m(\tau_m) \int_{\tau_c=\tau_m}^{\infty} \lambda e^{-\lambda\tau_c} d\tau_c d\tau_m \\ &= r_m^*(\lambda) = \left(\frac{1}{\lambda E[t_m]}\right)\left[1 - f_m^*(\lambda)\right]. \end{aligned} \tag{3}$$

2 Modeling of Algorithm II

This section derives the expected checkpoint interval $E_{II}[t_c]$ and the probability α_{II} of obsolete HLR backup record for

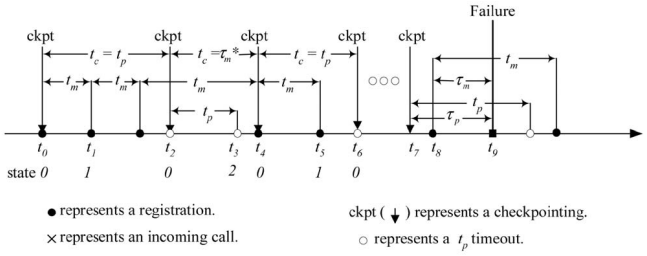

Fig. 3. The timing diagram for Algorithm II.

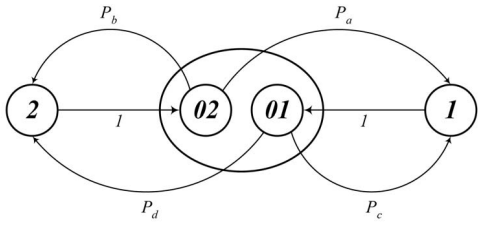

Fig. 4. The modified state diagram for Algorithm II.

Algorithm II. Consider the timing diagram in Fig. 3. In Algorithm II, if the t_p timer is restarted due to the t_p timeout event (i.e., a transition from **state 1** to **state 0**; see t_2 in Fig. 3), then the next checkpoint interval is $t_c = \max(\tau_m^*, t_p)$. On the other hand, if the t_p timer is restarted due to a registration event (i.e., a transition from **state 2** to **state 0**; see t_4 in Fig. 3), then the next checkpoint interval is $t_c = \max(t_m, t_p)$. The two random variables $\max(\tau_m^*, t_p)$ and $\max(t_m, t_p)$ are not the same in general. To distinguish the above two cases, **state 0** is split into **state 01** and **state 02**. If a checkpoint occurs due to a registration event, then the FSM moves from **state 2** to **state 02**. If a checkpoint occurs due to a t_p timeout event, then the FSM moves from **state 1** to **state 01**. Fig. 4 redraws the state diagram in Fig. 2 with these two new states. Let π_x be the probability that the FSM is in **state x**. Then, with probabilities

$$p_1 = \frac{\pi_{01}}{\pi_{01} + \pi_{02}} \quad \text{and} \quad p_2 = \frac{\pi_{02}}{\pi_{01} + \pi_{02}}, \tag{4}$$

the random variable t_c can be expressed as

$$t_c = p_1 \max(\tau_m^*, t_p) + p_2 \max(t_m, t_p). \tag{5}$$

In Fig. 4, it is clear that the transition probability from **state 1** to **state 01** is 1. Similarly, the transition probability from **state 2** to **state 02** is 1. Let the transition probabilities from **state 02** to **state 1** and **state 2** be p_a and p_b, respectively. Similarly, let the transition probabilities from **state 01** to **state 1** and **state 2** be p_c and p_d, respectively. These transition probabilities are derived as

$$\left.\begin{array}{l} p_a = f_m^*(\lambda), p_b = 1 - f_m^*(\lambda) \\ p_c = r_m^*(\lambda), p_d = 1 - r_m^*(\lambda) \end{array}\right\}. \tag{6}$$

From Fig. 4, the limiting probabilities π_x are expressed as follows:

$$\left.\begin{array}{l} 1 = \pi_{01} + \pi_{02} + \pi_1 + \pi_2 \\ \pi_1 = \pi_{01} \\ \pi_2 = \pi_{02} \\ \pi_1 = p_c \pi_{01} + p_a \pi_{02} \\ \pi_2 = p_d \pi_{01} + p_b \pi_{02} \end{array}\right\}. \tag{7}$$

With the above equations, (4) is solved to yield

$$p_1 = \frac{f_m^*(\lambda)}{1 + f_m^*(\lambda) - r_m^*(\lambda)} \quad \text{and} \quad p_2 = \frac{1 - r_m^*(\lambda)}{1 + f_m^*(\lambda) - r_m^*(\lambda)}. \tag{8}$$

From (5), the density function for t_c is

$$f_c(t_c) = p_1\left[\lambda e^{-\lambda t_c} R_m(t_c) + r_m(t_c) - r_m(t_c) e^{-\lambda t_c}\right] \tag{9}$$

$$+ p_2\left[\lambda e^{-\lambda t_c} F_m(t_c) + f_m(t_c) - f_m(t_c) e^{-\lambda t_c}\right]. \tag{10}$$

Based on the relationship between t_p, t_m, and τ_m^*, $f_c(t_c)$ can be reinterpreted in two cases:

Case 1. The first term in the second bracket of (9) (and (10)) represents the situation when $t_p > \tau_m^*$ (or $t_p > t_m$).

Case 2. The second and third terms in the second bracket of (9) (and (10)) represent the situation when $t_p < \tau_m^*$ (or $t_p < t_m$).

Therefore, (9) and (10) can also be reexpressed as

$$f_c(t_c) = f_{c1}(t_c) + f_{c2}(t_c),$$

where

$$f_{c1}(t_c) = \frac{f_m^*(\lambda)\lambda e^{-\lambda t_c} R_m(t_c)}{1 + f_m^*(\lambda) - r_m^*(\lambda)} + \frac{[1 - r_m^*(\lambda)]\lambda e^{-\lambda t_c} F_m(t_c)}{1 + f_m^*(\lambda) - r_m^*(\lambda)} \tag{11}$$

is the density function for Case 1, and

$$\begin{aligned} f_{c2}(t_c) = {} & \frac{f_m^*(\lambda)\left[r_m(t_c) - r_m(t_c) e^{-\lambda t_c}\right]}{1 + f_m^*(\lambda) - r_m^*(\lambda)} \\ & + \frac{[1 - r_m^*(\lambda)]\left[f_m(t_c) - f_m(t_c) e^{-\lambda t_c}\right]}{1 + f_m^*(\lambda) - r_m^*(\lambda)}. \end{aligned} \tag{12}$$

is the density function for Case 2.

From (9) and (10), the expected checkpoint interval for Algorithm II is

$$\begin{aligned} E_{II}[t_c] &= \int_{t_c=0}^{\infty} t_c f_c(t_c) dt_c \\ &= \frac{A_1 f_m^*(\lambda)}{1 + f_m^*(\lambda) - r_m^*(\lambda)} + \frac{A_2[1 - r_m^*(\lambda)]}{1 + f_m^*(\lambda) - r_m^*(\lambda)}, \end{aligned} \tag{13}$$

where

$$\begin{aligned} A_1 = {} & \int_{t_c=0}^{\infty} t_c \lambda e^{-\lambda t_c} R_m(t_c) dt_c + \int_{t_c=0}^{\infty} t_c r_m(t_c) dt_c \\ & - \int_{t_c=0}^{\infty} t_c r_m(t_c) e^{-\lambda t_c} dt_c \end{aligned} \tag{14}$$

$$= -\lambda \left\{ \frac{d\left[\frac{r_m^*(s)}{s}\right]}{ds} \right\}\Bigg|_{s=\lambda} + E[\tau_m] + \left[\frac{dr_m^*(s)}{ds}\right]\Bigg|_{s=\lambda} \tag{15}$$

$$= \frac{r_m^*(\lambda)}{\lambda} + E[\tau_m]. \tag{16}$$

The first term in (15) is derived from the first term in (14) using the single integral rule and the linear scaling rule of Laplace transform [19]. The third term in (15) is derived from the third term in (14) using the linear scaling rule of Laplace transform. Similarly,

$$\begin{aligned} A_2 = {} & \int_{t_c=0}^{\infty} t_c \lambda e^{-\lambda t_c} F_m(t_c) dt_c + \int_{t_c=0}^{\infty} t_c f_m(t_c) dt_c \\ & - \int_{t_c=0}^{\infty} t_c f_m(t_c) e^{-\lambda t_c} dt_c \\ = {} & \frac{f_m^*(\lambda)}{\lambda} + E[t_m]. \end{aligned} \tag{17}$$

From (13), (16), and (17), we have

$$E_{II}[t_c] = p_1\left\{\frac{r_m^*(\lambda)}{\lambda} + E[\tau_m]\right\} + p_2\left\{\frac{f_m^*(\lambda)}{\lambda} + E[t_m]\right\}. \tag{18}$$

For the probability α_{II} of obsolete HLR backup record in Algorithm II, there is no close-form expression when arbitrary $f_m(t_m)$ is used. In this paper, we consider mix-Erlang density function for t_m, which is expressed as

$$f_m(t_m) = \sum_{i=1}^{j} q_i \left[\frac{(\mu_i t_m)^{n_i-1}}{(n_i-1)!}\right]\mu_i e^{-\mu_i t_m}, \tag{19}$$

where $\sum_{i=1}^{j} q_i = 1$, μ_i are the scale parameters and n_i are the shape parameters. The mix-Erlang distribution is selected because it has been proven as a good approximation to many other distributions as well as measured data [6], [10]. For purposes of illustration, it suffices to derive α_{II} by considering t_m with Erlang distribution where the density function is

$$f(n,\mu,t_m) = \left[\frac{(\mu t_m)^{n-1}}{(n-1)!}\right]\mu e^{-\mu t_m}$$

(i.e., $j = 1$ in (19)). It is straightforward to extend our results with Erlang t_m distribution to the results with mix-Erlang distribution. Let $F(n,\mu,t_m)$ be the Erlang distribution function. Then,

$$F(n,\mu,t_m) = 1 - \sum_{j=0}^{n-1}\left[\frac{(\mu t_m)^j}{j!}\right]e^{-\mu t_m}, \tag{20}$$

$$= 1 - \left(\frac{1}{\mu}\right)\sum_{j=1}^{n} f(j,\mu,t_m) \tag{21}$$

and the Laplace transform $f^*(n,\mu,s)$ is expressed as

$$f^*(n,\mu,s) = \left(\frac{\mu}{\mu+s}\right)^n. \tag{22}$$

The reverse residual time τ_m of t_m has the density function $r(n,\mu,\tau_m)$, the distribution function $R(n,\mu,\tau_m)$, and the Laplace transform $r^*(n,\mu,s)$. Since $E[t_m] = \frac{n}{\mu}$, from (2) and (21),

$$r(n,\mu,\tau_m) = \left(\frac{1}{n}\right)\sum_{j=1}^{n} f(j,\mu,\tau_m),$$
$$R(n,\mu,\tau_m) = \left(\frac{1}{n}\right)\sum_{j=1}^{n} F(j,\mu,\tau_m), \tag{23}$$

and

$$r^*(n,\mu,s) = \left(\frac{\mu}{ns}\right)\left[1 - \left(\frac{\mu}{\mu+s}\right)^n\right]. \tag{24}$$

Consider a checkpoint interval t_c. Let us revisit the two cases mentioned before:

Case 1. $t_c = \max(t_m, t_p) = t_p$ or $\max(\tau_m^*, t_p) = t_p$: The HLR record is always updated in this t_c interval. The t_c density function for this case is expressed in (11).

Case 2. $t_c = \max(t_m, t_p) = t_m$ or $\max(\tau_m^*, t_p) = \tau_m^*$: The HLR record is never updated in this t_c interval. The t_c density function for this case is expressed in (12).

To derive α_{II}, we only need to consider Case 1. From (11) and (23),

$$f_{c1}(t_c) = p_1\lambda e^{-\lambda t_c}\left(\frac{1}{n}\right)\left[\sum_{m=1}^{n} F(m,\mu,t_c)\right] + p_2\lambda e^{-\lambda t_c}F(n,\mu,t_c) = \left(\frac{p_1}{n}\right)\sum_{m=1}^{n} g(m,\mu,t_c) + p_2 g(n,\mu,t_c), \tag{25}$$

where

$$g(i,\mu,t) = \lambda e^{-\lambda t}F(i,\mu,t) \tag{26}$$

$$= \lambda e^{-\lambda t} - \sum_{k=1}^{i}\left[\frac{\mu^{k-1}\lambda}{(\mu+\lambda)^k}\right]f(k,\mu+\lambda,t). \tag{27}$$

From (2) and (27), the density function of the (reverse) residual time corresponding to $g(i,\mu,t)$ is

$$h(i,\mu,t) = \lambda e^{-\lambda t} - \sum_{k=1}^{i}\left[\frac{\mu^{k-1}\lambda}{k(\mu+\lambda)^k}\right]\sum_{j=1}^{k} f(j,\mu+\lambda,t). \tag{28}$$

Let τ_c be the reverse residual time of t_c (see Fig. 3) in Case 1. The density function of τ_c is

$$r_{c1}(\tau_c) = \left(\frac{p_1}{n}\right)\left[\sum_{m=1}^{n} h(m,\mu,\tau_c)\right] + p_2 h(n,\mu,\tau_c). \tag{29}$$

From (29), α_{II} is derived as

$$\begin{aligned}\alpha_{II} &= \Pr[\tau_c > \tau_m] \\ &= \int_{\tau_m=0}^{\infty} r(n,\mu,\tau_m)\int_{\tau_c=\tau_m}^{\infty} r_{c1}(\tau_c)d\tau_c d\tau_m \\ &= \left(\frac{p_1}{n}\right)\sum_{m=1}^{n}\int_{\tau_m=0}^{\infty} r(n,\mu,\tau_m)\int_{\tau_c=\tau_m}^{\infty} h(m,\mu,\tau_c)d\tau_c d\tau_m \\ &\quad + p_2\int_{\tau_m=0}^{\infty} r(n,\mu,\tau_m)\int_{\tau_c=\tau_m}^{\infty} h(n,\mu,\tau_c)d\tau_c d\tau_m \\ &= \left(\frac{p_1}{n}\right)\sum_{m=1}^{n}\Big[A_3(m) - A_4(m)\Big] + p_2\Big[A_3(n) - A_4(n)\Big],\end{aligned} \tag{30}$$

where

$$\begin{aligned}A_3(m) &= \int_{\tau_m=0}^{\infty} r(m,\mu,\tau_m)\int_{\tau_c=\tau_m}^{\infty}\lambda e^{-\lambda\tau_c}d\tau_c d\tau_m \\ &= \left(\frac{\mu}{m\lambda}\right)\left[1 - \left(\frac{\mu}{\mu+\lambda}\right)^m\right]\end{aligned} \tag{31}$$

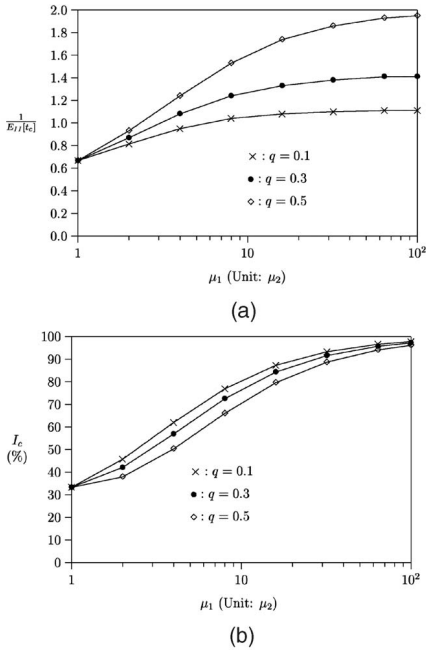

Fig. 5. The checkpoint overhead ($\lambda = (\mu_1 + \mu_2)/2$). (a) Checkpointing cost for Algorithm I. (b) Improvement of Algorithm II over Algorithm I.

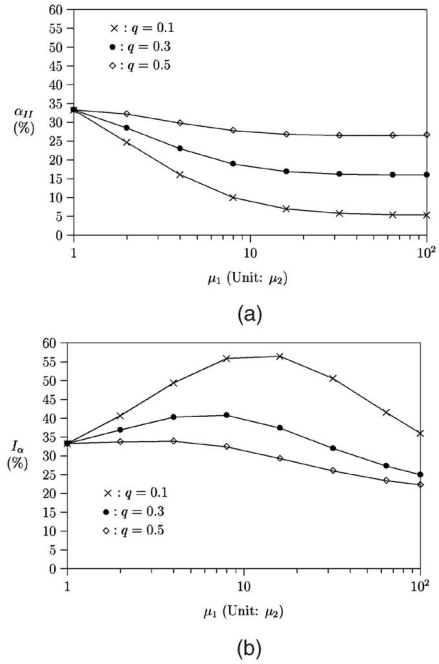

Fig. 6. The probability of obsolete HLR backup record after failure ($\lambda = (\mu_1 + \mu_2)/2$). (a) Obsolete record probability for Algorithm II. (b) Improvement of α_{II} over α_I.

and

$$A_4(m) = \int_{\tau_m=0}^{\infty} r(m,\mu,\tau_m) \int_{\tau_c=\tau_m}^{\infty} \sum_{k=1}^{m} \left[\frac{\mu^{k-1}\lambda}{k(\mu+\lambda)^k}\right] \times \sum_{j=1}^{k} f(j,\mu+\lambda,\tau_c) d\tau_c d\tau_m = \sum_{k=1}^{m} \left[\frac{\mu^{k-1}\lambda}{k(\mu+\lambda)^k}\right] \sum_{j=1}^{k} B(m,j), \quad (32)$$

where

$$B(m,j) = \int_{\tau_m=0}^{\infty} r(m,\mu,\tau_m) \int_{\tau_c=\tau_m}^{\infty} f(j,\mu+\lambda,\tau_m) d\tau_c d\tau_m = \sum_{l=1}^{j} \left[\frac{1}{m(\mu+\lambda)}\right] \sum_{i=1}^{m} \int_{\tau_m=0}^{\infty} f(i,\mu,\tau_m) f(l,\mu+\lambda,\tau_m) d\tau_m. \quad (33)$$

Our analytic model is validated against simulation experiments (the simulation model is similar to the one we developed in [8], [11], [15] and the details are omitted). In Figs. 5 and 6, the symbols ×, •, and ◇ represent simulation data, and the solid curves represent the analytic results. The figures indicate that the errors between the analytic results and the simulation data are within 2 percent.

3 Performance Evaluation of Algorithms I and II

This section uses numerical examples to investigate the performance of Algorithms I and II. For purposes of illustration, we consider the simplest mix-Erlang format for the t_m distribution (i.e., $j = 2$ and $n_1 = n_2 = 1$ in (19)):

$$f_m(t_m) = q\mu_1 e^{-\mu_1 t_m} + (1-q)\mu_2 e^{-\mu_2 t_m}. \quad (34)$$

This density function can be used to approximate the fast and slow movement behaviors of a mobile user. When $\mu_1 >> \mu_2$, it means that with probability q, the user moves very fast (i.e., crossing the SGSN areas with high frequency μ_1); and with probability $1-q$, the user moves very slowly (i.e., crossing the SGSN areas with low frequency μ_2).

Based on (34), we compute $E_I[t_c]$, α_I, $E_{II}[t_c]$, and α_{II}. For Algorithm I, $E_I[t_c]$ is expressed in (1), which is not affected by the t_m distribution. From (3),

$$\alpha_I = \left(\frac{1}{\lambda}\right)\left(\frac{q}{\mu_1} + \frac{1-q}{\mu_2}\right)^{-1}\left[1 - \frac{q\mu_1}{\mu_1+\lambda} - \frac{(1-q)\mu_2}{\mu_2+\lambda}\right]. \quad (35)$$

From (18),

$$E_{II}[t_c] = p_1 C_1 + (1-p_1)C_2, \quad (36)$$

where

$$p_1 = \left[\frac{q\mu_1}{\mu_1+\lambda} + \frac{(1-q)\mu_2}{\mu_2+\lambda}\right]\left\{1 + \frac{q\mu_1}{\mu_1+\lambda} + \frac{(1-q)\mu_2}{\mu_2+\lambda} - \alpha_I\right\}^{-1},$$

$$C_1 = \left(\frac{q}{\mu_1} + \frac{1-q}{\mu_2}\right)^{-1}\left\{\frac{1}{\lambda^2} + q\left[\frac{1}{\mu_1^2} - \frac{\mu_1}{\lambda^2(\mu_1+\lambda)}\right] + (1-q)\left[\frac{1}{\mu_2^2} - \frac{\mu_2}{\lambda^2(\mu_2+\lambda)}\right]\right\},$$

and

$$C_2 = q\left[\frac{1}{\mu_1} + \frac{\mu_1}{\lambda(\mu_1+\lambda)}\right] + (1-q)\left[\frac{1}{\mu_2} + \frac{\mu_2}{\lambda(\mu_2+\lambda)}\right].$$

From the derivation in the previous section,

$$\alpha_{II} = \frac{2q\mu_1^2}{(2\mu_1+\lambda)(\mu_1+\lambda)} + \frac{2(1-q)\mu_2^2}{(2\mu_2+\lambda)(\mu_2+\lambda)}. \quad (37)$$

To compare Algorithm II with Algorithm I, we define two *improvement* indicators. The indicator I_c represents the percentage of checkpoint cost saved by Algorithm II over Algorithm I. Since the checkpoint cost is proportional to the checkpoint frequency, I_c is defined as

$$I_c = \frac{\frac{1}{E_I[t_c]} - \frac{1}{E_{II}[t_c]}}{\frac{1}{E_I[t_c]}} = \frac{E_{II}[t_c] - E_I[t_c]}{E_{II}[t_c]}. \quad (38)$$

Another indicator I_α represents the percentage of reduction for α provided by Algorithm II over Algorithm I. That is

$$I_\alpha = \frac{\alpha_I - \alpha_{II}}{\alpha_I}. \quad (39)$$

Fig. 5a plots the checkpoint frequency (i.e., $1/E_{II}[t_c]$) curves based on (36). Fig. 5b plots the I_c curves based on (38). In these figures, $\lambda = (\mu_1 + \mu_2)/2$, and the checkpoint frequency is normalized by μ_2. Fig. 5a shows the intuitive result that the checkpoint cost for Algorithm II is an increasing function of the registration frequency. The nontrivial result is that the checkpoint cost can be quantitatively computed through our model. Fig. 5b shows that Algorithm II significantly outperforms Algorithm I in terms of reducing the checkpoint cost.

Based on (37), Fig. 6a plots α_{II} against μ_1/μ_2. When μ_1/μ_2 increases, the variance of interregistration interval increases. Therefore, we will observe a small number of checkpoint intervals that experience many registrations and a large number of checkpoint intervals that experience no registration. From the description of Algorithm II, it is also clear that the checkpoint intervals without registration are longer than the intervals with registrations. Since a failure is a random observer of the checkpoint intervals, the failure time is more likely to fall in a checkpoint interval without any registration. Therefore, α_{II} decreases as μ_1/μ_2 increases (this phenomenon is also true for Algorithm I). Based on (39), Fig. 6b indicates that Algorithm II provides 20-55 percent improvement over Algorithm I in terms of α performance.

4 Conclusions

We studied failure restoration of Home Location Register (HLR) for UMTS and GPRS. By utilizing per-user checkpoint, an HLR record is saved into a backup database from time to time. When a failure occurs, the backup record is restored to the HLR. We first described a commonly used basic checkpoint algorithm (referred to as Algorithm I). Then, we proposed a new checkpoint algorithm called Algorithm II. An analytic model was developed to compare these two algorithms in terms of the checkpoint cost and the probability α of obsolete HLR backup record. The analytic model was validated against simulation experiments. For all input parameter values considered in this paper, Algorithm II can save more than 50 percent of the checkpoint cost over Algorithm I. For the performance of α, Algorithm II demonstrates 20-55 percent improvement over Algorithm I. As a final remark, we note that failure restoration for a SGSN (or a visitor location register in the circuit switched service domain) is very different from HLR failure restoration described in this paper. No checkpointing is performed for a SGSN because all MS records in the SGSN are temporary, and it is useless to store these temporary records into backup. Details of SGSN failure restoration can be found in [7], [4], [11].

Acknowledgments

This work was sponsored in part by Chair Professorship of Providence University, IIS/Academia Sinica, FarEastone, and CCL/ITRI. Equations (35) and (36) were contributed by Sok-Ian Sou under NSC Excellence project NSC93-2752-E-0090005-PAE.

References

[1] 3GPP, 3rd Generation Partnership Project; Technical Specification Group Services and Systems Aspects; General Packet Radio Service (GPRS); Service Descripton; Stage 2, Technical Specification 3G TS 23.060 version 4.1.0 (2001-06), 2001.
[2] P. Agrawal, G. Omidyar, and A. Wolisz, *IEEE Wireless Communications Magazine,* special issue on mobile and wireless Internet: architectures and protocols, 2002.
[3] G. Cao, "Proactive Power-Aware Cache Management for Mobile Computing Systems," *IEEE Trans. Computers,* vol. 51, no. 6, pp. 608-621, 2002.
[4] M.-F. Chang, Y.-B. Lin, and S.-C. Su, "Improving Fault Tolerance of GSM Network," *IEEE Network,* vol. 1, no. 12, pp. 58-63, 1998.
[5] ETSI/TC, Restoration Procedures, Version 4.2.0, Technical Report Recommendation GSM 03.07, ETSI, 1993.
[6] Y. Fang and I. Chlamtac, "Teletraffic Analysis and Mobility Modeling for PCS Networks," *IEEE Trans. Comm.,* vol. 47, no. 7, pp. 1062-1072, July 1999.
[7] Y. Fang, I. Chlamtac, and H. Fei, "Analytical Results for Optimal Choice of Location Update Interval for Mobility Database Failure Restoration in PCS Networks," *IEEE Trans. Parallel and Distributed Systems,* 2000.
[8] Z. Haas and Y.-B. Lin, "On Optimizing the Location Update Costs in the Presence of Database Failures," *ACM/Baltzer Wireless Networks J.,* vol. 4, no. 5, pp. 419-426, 1998.
[9] A. Kahol, S. Khurana, S. Gupta, and P. Srimani, "An Efficient Cache Management Scheme for Mobile Environment," *Proc. IEEE Int'l Conf. Distributed Computing Systems (ICDCS),* 2000.
[10] F.P. Kelly, *Reversibility and Stochastic Networks.* John Wiley & Sons, 1979.
[11] Y.-B. Lin, "Failure Restoration of Mobility Databases for Personal Communication Networks," *ACM-Baltzer J. Wireless Networks,* vol. 1, pp. 365-372, 1995.
[12] Y.-B. Lin, "A Cache Approach for Supporting Life-Time Universal Personal Telecommunication Number," *ACM-Baltzer Wireless Networks,* vol. 2, pp. 155-160, 1996.
[13] Y.-B. Lin and I. Chlamtac, *Wireless and Mobile Network Architectures.* John Wiley & Sons, 2001.
[14] Y.-B. Lin and P. Lin, "Performance Modeling of Location Tracking Systems," *ACM Mobile Computing and Comm. Rev.,* vol. 2, no. 3, pp. 24-27, 1998.
[15] Y.-B. Lin and V.K. Mak, "Eliminating the Boundary Effect of a Large-Scale Personal Communication Service Network Simulation," *ACM Trans. Modeling and Computer Simulation,* vol. 4, no. 2, 1994.
[16] Y.-B. Lin, W.-R. Lai, and J.-J. Chen, "Effects of Cache Mechanism on Wireless Data Access," *IEEE Trans. Wireless Comm.,* vol. 2, no. 6, 2003.
[17] S.M. Ross, *Simulation.* Academic Press, 1996.
[18] J. Shim, P. Scheuermann, and R. Vingralek, "Proxy Cache Algorithms: Design, Implementation, and Performance," *IEEE Trans. Knowledge and Data Eng.,* vol. 11, no. 4, pp. 549-562, July/Aug. 2000.
[19] E.J. Watson, *Laplace Transforms and Applications.* Birkhauserk, 1981.

Yi-Bing Lin received the BSEE degree from the National Cheng Kung University in 1983 and the PhD degree in computer science from the University of Washington in 1990. He is chair professor in the Department of Computer Science and Information Engineering (CSIE), National Chiao Tung University (NCTU). Dr. Lin is a fellow of the IEEE and the ACM.