Restoration Scheme of Mobility Databases by Mobility Learning and Prediction in PCS Networks

Joon-Min Gil, *Member, IEEE*, Chan Yeol Park, Chong-Sun Hwang, Doo-Soon Park, *Member, IEEE*, Jin Gon Shon, and Young-Sik Jeong

***Abstract*—This paper proposes a restoration scheme based on mobility learning and prediction in the presence of the failure of mobility databases in personal communication systems (PCSs). In PCSs, mobility databases must maintain the current location information of users to provide a fast connection for them. However, the malfunction of mobility databases may cause some location information to be lost. As a result, without an explicit restoration procedure, incoming calls to users may be rejected. Therefore, an explicit restoration scheme against the malfunction of mobility databases is needed to guarantee continuous service availability to users. Introducing mobility learning and prediction into the restoration process allows systems to locate users after a failure of mobility databases. In failure-free operations, the movement patterns of users are learned by a Neuro-Fuzzy Inference System (NFIS). After a failure, an inference process of the NFIS is initiated and the users' future location is predicted. This is used to locate lost users after a failure. This proposal differs from previous approaches using a checkpoint because it does not need a backup process nor additional storage space to store checkpoint information. In addition, simulations show that our proposal can reduce the cost needed to restore the location records of lost users after a failure when compared to the checkpointing scheme.**

***Index Terms*—Mobility database failure restoration, mobility learning and prediction, personal communication systems.**

I. Introduction

PERSONAL communication systems (PCSs) [1], [2] enable users to communicate with each other at any time from any location. Thus, users no longer need to remain at a fixed location to receive messages. However, to do this, the location information of users should first be identified to systems before a connection is established. In PCSs this location information is usually maintained in mobility databases. When there are calls for users, the calls are delivered using the location information.

Manuscript received December 21, 2000; revised June 15, 2001. This work was supported in part by the Ministry of Information and Communication of Korea ("Support Project of University Foundation Research ⟨1997–1999⟩" supervised by IITA).

J.-M. Gil and C.-S. Hwang are with the Department of Computer Science and Engineering, Korea University, Seoul 136-701, Korea (e-mail: jmgil@disys.korea.ac.kr; hwang@disys.korea.ac.kr).

C. Y. Park is with Research and Development, ECO Corporation, Seoul 135-010, Korea (e-mail: chan@eco.co.kr).

D.-S. Park is with the Division of Information Technology Engineering, Soonchunhyang University, Choongnam-Do 336-745, Korea (e-mail: parkds@asan.sch.ac.kr).

J. G. Shon is with the Department of Computer Science, Korea National Open University, Seoul 110-791, Korea (e-mail: jgshon@mail.knou.ac.kr).

Y.-S. Jeong is with the School of Computer and Communication Engineering, Wonkwang University, Chollabuk-Do 570-749, Korea (e-mail: ysjeong@wonnms.wonkwang.ac.kr).

Publisher Item Identifier S 0733-8716(01)08480-3.

Typically, mobility databases are hierarchically composed in PCS networks. When users move across a registration area boundary, mobility databases are updated to include the users' new location. This procedure, which is known as *location registration*, is initiated by users. When users are called, the mobility databases are queried to find the location where users currently visiting. This procedure, referred to as *location query*, is performed by systems.

Over the years, several strategies have been devised for minimizing the cost of the location management in the existing standards, IS-41 in the United States and GSM in Europe. Most of them have focused on optimizing the registration and location query costs [3]–[6]. However, there have been a few works on the reliability aspect of mobility databases. When the mobility databases fail, the location records stored in the databases are lost and incoming calls to users may be rejected. This results in a large service delay and a serious deterioration in the performance of systems. Thus, there is a need for an explicit restoration procedure for mobility databases to guarantee continuous service availability to users.

Without an explicit restoration procedure, the delay in restoring location information after a failure depends on the length of the users' silence period. In IS-41 [7], there is no explicit expedient to restore mobility databases. After a failure, mobility databases incrementally reconstruct the location information of a user each time the user sends a registration message. Before the users originate any message, all incoming calls to them are rejected. In GSM [8], mobility databases are backed up periodically, i.e., a checkpointing procedure is performed for the recovery of failed mobility databases. After a failure, the databases are immediately restored from a stable storage. However, some backup records may be obsolete. If the interval between the checkpointing time and the failure time of mobility databases is relatively long, the users' latest location information may not be updated, in which case, location records during this period may be rendered obsolete.

Users' mobility plays an important role in locating lost users after a failure. To verify the obsoleteness of backup records, lost users should be paged, starting from the area recorded on the backup. Unfortunately, unsuccessful paging occurs when users move far away from the area recorded on the backup. This results in deteriorated system performances and service quality. If systems can predict the probable location of users after a failure, it would cut the paging cost tremendously. User mobility prediction would be useful in enhancing the performance of systems during the restoration of failed mobility databases. In fact, user mobility prediction has been used in various application areas, such as service pre-allocation, resource pre-assignment, and so

on [9]–[11]. In this paper, we consider user mobility prediction for the restoration of failed mobility databases in PCSs.

Our restoration scheme is based on learning and predicting the movement patterns of users. Whenever users register a new location, their movement patterns are learned by a Neuro-Fuzzy Inference System (NFIS). When a failure occurs in mobility databases, the probable location of users may be predicted by the NFIS. This predicted location is used to find the users' current location. Contrary to other approaches using checkpoint, our restoration scheme does not need any backup process. Thus, it has less costs during failure-free operations. In addition, there's no need for additional storage space to store checkpoint information and users experience only slight delays in service.

This paper is organized as follows. In Section II, we briefly describe location management and fault tolerance in PCS networks. Section III provides an overview of related works on the restoration of mobility databases. In Section IV, we present a brief introduction to NFIS and model mobility learning and prediction in mobile environments. In Section V, we propose a restoration scheme based on mobility learning and prediction. Procedures for the restoration of mobility databases are also presented in this section. Section VI shows the performance evaluation of our restoration scheme with simulations. This section also compares our scheme with the periodic checkpointing scheme. Finally, the conclusion of this paper is given in Section VII.

II. Location Management and Fault Tolerance in PCS Networks

A. System Architecture

The network architecture of PCS is composed of two sets of entities: one is the fixed network and the other is mobile units (or mobile users). At the end of the fixed network, a base station (BS) is augmented to provide users with an interface. Users can only communicate with a BS through a wireless link. The service area is partitioned into cells. A cell is the geographical area covered by a BS. Several cells compose a registration area (RA). Each RA is connected to a mobile switching center (MSC) through a wired network. An MSC typically provides switching functions which coordinate location registration and call delivery. The MSC has access to the mobility databases in the network. These mobility databases are used to store the location and service information for each registered user in PCS. The architecture of mobility databases in PCS is typically based on a two-tier hierarchy structure. With this structure, the mobility databases consist of the home location register (HLR) and the visitor location register (VLR). The HLR is a global database which stores information about all users registered in PCS. A VLR is a local database usually associated with the MSC. It stores information about users visiting the MSC's RA. Whenever a handover occurs during the wireless connection of users, the RA maintains information about the current cell location of users. Fig. 1 shows the mobility database architecture of PCS used in this paper.

B. Location Registration and Location Query

The location registration procedure is used to update location records in mobility databases when users move into a new RA. The location query procedure is used for establishing an incoming call to users. Below is a description of the location registration and query procedures.

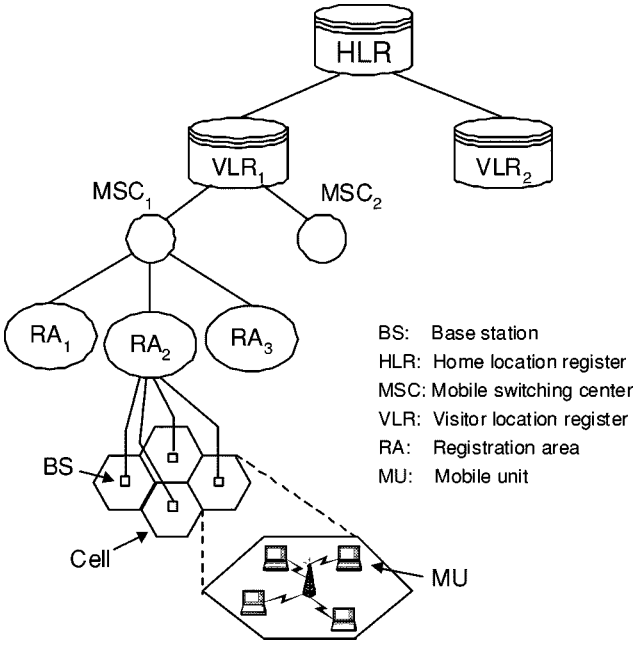


Fig. 1. Mobility database architecture.

The location registration procedure works as follows: when a user, m_1 move from a RA RA_1 with MSC MSC_1 to another RA RA_2 with MSC MSC_2, m_1 sends a location registration message to MSC_2 [Step 1 in Fig. 2(a)]. MSC_2 delivers the registration message to the HLR [Step 2 in Fig. 2(a)] and confirms the receipt of the registration message from MSC_2 [Step 3 in Fig. 2(a)]. The HLR also informs MSC_1 about the relocation of m_1. MSC_1 deletes m_1's entry in VLR_1 on receiving this message [Step 4 in Fig. 2(a)]. MSC_2 also updates VLR_2 with m_1's entry.

The location query procedure works as follows: when a user m_1 calls another user m_2, m_1 sends a location query message to MSC_1 [Step 1 in Fig. 2(b)]. MSC_1 then queries the VLR_1 to find the location information of the called user m_2. If the information is retrieved, MSC_1 establishes a connection between m_1 and m_2. This case indicates that two users m_1 and m_2 reside in RA_1. If the information does not exist in VLR_1, MSC_1 queries the HLR to find m_2's location information [Step 2 in Fig. 2(b)]. The HLR determines the RA of the m_2 and sends a route request message to MSC_2 [Step 3 in Fig. 2(b)]. MSC_2 then determines a temporary location directory number for m_2 and transfers the information to the HLR [Step 4 in Fig. 2(b)]. The HLR delivers this information to MSC_1 [Step 5 in Fig. 2(b)], and establishes a connection between two users m_1 and m_2 [step 6 in Fig. 2(b)].

C. Failure of Location Databases

When a failure occurs in mobility databases, location records stored in them are lost. In addition, location registration and location query procedures can no longer be performed. So, the exact location of users may not be identified, and the latest location information of users may not be updated. In this situation, a connection may not be established for incoming calls to users.

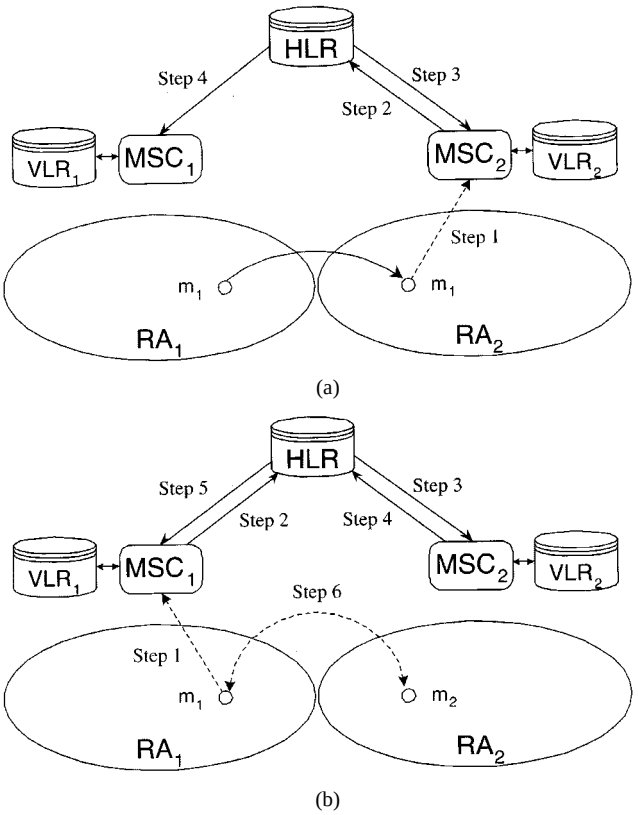


Fig. 2. (a) Registration and (b) query procedure.

Typically, without an explicit restoration procedure, the location information can be restored after a failure by means of one of the following events.

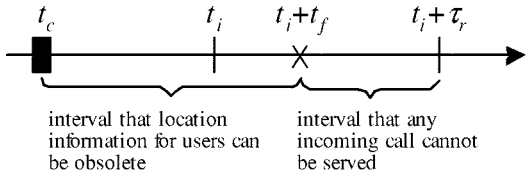
- Autonomous location registration: Users periodically inform networks of their current location.
- Call origination: The current location of users is identified to networks at a call origination.
- Registration boundary crossing: Users necessarily send a registration message to networks when they cross a registration boundary.

The three events described above are used to restore mobility databases in the absence of an explicit restoration procedure. The delay and cost in restoring mobility databases depend on the occurrence of three events by users. If any of these three events does not occur after a failure, the location information of users are permanently not restored. After a long time passes, if one of these events occurs, the location information of users can be restored by it. However, services may not be served to users for the failure period.

Apart from the implicit restoration procedure in PCS, an explicit restoration procedure is required after a failure to reconstruct lost location information. Consequently, the restoration of mobility databases performed only by systems will guarantee service availability to users.

III. Related Works

The existing two standards, IS-41 and GSM, are based on a two-tier hierarchical structure with two types of mobility databases, specifically HLR and VLR. Unlike the IS-41 system, GSM has provided an explicit restoration procedure [8] to recover mobility databases after a failure occurs. The following procedure is performed for the recovery of failed mobility databases. During failure-free operations, the location information recorded in the mobility databases, HLR and VLR, is periodically checkpointed into a stable storage co-located in the mobility databases. After a failure occurs, the location information is immediately reconstructed from the backup. However, backup records are not always exact; some backup records may be obsolete, because the location information may have changed during the interval between checkpoint time and failure time. In such instances, mobility databases are restored from incorrect location information. Incoming calls to users may be lost due to the absence of location information for the users. Furthermore, users often experience delays in service using this checkpointing scheme, not to mention it takes up additional storage space. The bigger the number of users, the larger the amount of records needed to store the location information of users.



- t_c : time at which a checkpoint is taken
- t_i : time at which the last location update occurs
- t_i+t_f : time at which a failure occurs in the mobility database
- $t_i+\tau_r$: time at which a user registers his location

Fig. 3. Time flow example.

To overcome these problems, several restoration schemes have been proposed in [12], [13]. These works have focused on deriving the optimal checkpoint interval to balance the checkpointing cost against the paging cost. However, it is not adequate to apply the interval to all users due to a different mobility of each user. The movement behavior of users is closely related to the cost and delay required to locate users after a failure. For example, fast-moving users may quickly move out of the last checkpointed location. In this case, systems should page entire cells within a RA due to the first unsuccessful paging in the last checkpointed location. Moreover, if users move far away from the last checkpointed location, more successive paging must be executed, resulting in redundant paging. Thus, the checkpointing of mobility databases does not seem to be an appropriate approach for improving the efficiency of a restoration procedure.

Let us consider an illustrative example. Fig. 3 shows a time flow for such a situation. In this figure, t_c represents the time at which a checkpoint for mobility databases is taken. After checkpointing, a user moves out of the checkpointed RA and moves to a new RA at time t_i. τ_r represents a residence time in the new RA. That is, the user moves to the new RA at time t_i and moves out of the RA at time $t_i+\tau_r$. We assume that the failure of mobility databases occurs at the time t_i+t_f between t_i and $t_i+\tau_r$.

After a failure occurs, any incoming calls to the user cannot be delivered to a target user during the time between t_i+t_f and $t_i+\tau_r$. If τ_r is relatively long, the no-service time for incoming

calls will be also long. When a failure occurs at the time $t_i + t_f$, mobility databases are restored from a stable storage immediately. At this time, the system starts to page the area recorded on the checkpoint. If an interval between t_c and $t_i + t_f$ is very long, the delay time for checkpointing will be short. However, most users may move far away from checkpointed location at time t_c. Since users may not reside in the cell which corresponds to the checkpointed location, first paging for the cell may be unsuccessful, at which time the system must perform entire paging within the new RA. As user mobility grows larger, the redundant paging becomes more. On the other hand, if an interval between t_c and $t_i + t_f$ becomes shorter, the restored location from backup may be exact. However, the location information is frequently saved into a stable storage and so the delay time for checkpointing will be larger.

So far, we have described some restoration schemes in PCS. As shown in these schemes, an explicit restoration procedure after a failure is needed to provide users with continuous service availability. In addition, the failed mobility databases should be aggressively restored by systems rather than by users. So, restoration delay should not be dependent on user-initiated events. Moreover, it is important that user mobility be considered during failure-free operations. To reduce obsolete location information minimally, the failed mobility database should be reconstructed with the location where users probably reside after a failure. This paper introduces the mobility learning of users' moving trajectory in failure-free operations. After a failure, mobility prediction is used to restore failed mobility databases. The next section describes mobility learning and prediction.

IV. Mobility Learning and Prediction

Mobility learning and prediction plays an important part in our restoration procedure. User mobility is a key to locating users after a failure occurs in mobility databases. In existing checkpointing schemes, there is a danger for some backup records to be obsolete because the user mobility between checkpoint time and failure time is not updated. The more complex the user mobility, the greater the chances for obsolete information to occur. On the other hand, our approach to mobility learning and prediction extrapolates probable user locations using the previously known mobility information by an NFIS. This approach promises better adaptability for various types of user mobility.

A. Neuro-Fuzzy Inference Model

In this paper, we use the simplified fuzzy inference model form among various fuzzy models. The simplified fuzzy inference model, referred to as the zero-order Sugeno' model in [14], is based on the fuzzy IF–THEN rules whose consequence is a real number. Thus, this model provides the inference structure that avoids the time-consuming process of defuzzification in an inference procedure. The form of fuzzy IF–THEN rules is as follows:

$$\text{Rule } i\text{: IF } x_1 \text{ is } A_1^i \text{ and, } \ldots, \text{ and } x_d \text{ is } A_d^i, \text{ THEN } y \text{ is } \omega_i \tag{1}$$

where

- i a fuzzy rule number;
- $x_1, \ldots, x_d$ input variables;
- A_j^i a fuzzy set for input variable x_j in the ith fuzzy rule;
- ω_i a real number for output variable y in the ith fuzzy rule ($i = 1, 2, \ldots, n$, $j = 1, 2, \ldots, d$).

Given the real-valued input vector $\vec{x} = [x_1, x_2, \ldots, x_d]$, the real-valued output of the fuzzy model is inferred as follows:

$$f(\vec{x}) = \frac{\sum_{i=1}^{n} \mu_i \cdot \omega_i}{\sum_{i=1}^{n} \mu_i}, \qquad \mu_i = \prod_{j=1}^{d} \mu_{A_j^i}(x_j). \tag{2}$$

Here, $\mu_{A_j^i}$ is a fuzzy membership function of the fuzzy set A_j^i and μ_i is a membership value of ith fuzzy rule. This mapping is performed using a singleton fuzzifier as fuzzification, a product-inference operator as fuzzy inference procedure, and a weighted-average technique as defuzzification. The fuzzy membership function can employ various forms such as triangular, trapezoid, and Gaussian according to application problems. Because of its smoothness and concise notation, the Gaussian function is becoming increasingly popular for specifying fuzzy sets. Moreover, since the Gaussian function has a continuous and differentiable property, it suitably applies to the learning rules. Thus, we employ the Gaussian function as the fuzzy membership function

$$\mu_{A_j^i}(x_j) = \exp\left[-\frac{1}{2} \cdot \left(\frac{x_j - c_j^i}{\sigma_j^i}\right)^2\right] \tag{3}$$

where c_j^i is the central value of the fuzzy membership function and σ_j^i is the variance of the central value of the fuzzy membership function.

The parameters c_j^i and σ_j^i in (3) and the parameter ω_i in (2) are iteratively adjusted by backpropagation learning algorithms [14] in order to minimize a learning error ($e = 1/2 \sum_p (y_p - \hat{y}_p)^2$) and produce a better output. The learning rules for these parameters are

$$\begin{aligned} c_j^i(t+1) &= c_j^i(t) - \eta \cdot \frac{\partial e}{\partial c_j^i} = c_j^i(t) - \eta \cdot \frac{\partial e}{\partial \mu_i} \cdot \frac{\partial \mu_i}{\partial \mu_{A_j^i}} \cdot \frac{\partial \mu_{A_j^i}}{\partial c_j^i} \\ &= c_j^i(t) - \eta \cdot (y_p - \hat{y}_p) \cdot \left(\frac{(\omega_i(t) - \hat{y}_p)}{\sum_{k=1}^{n} \mu_k}\right) \\ &\quad \cdot \mu_i \cdot \left(\frac{x_j - c_j^i(t)}{\left(\sigma_j^i(t)\right)^2}\right) \end{aligned} \tag{4}$$

$$\begin{aligned} \sigma_j^i(t+1) &= \sigma_j^i(t) - \eta \cdot \frac{\partial e}{\partial \sigma_j^i} = \sigma_j^i(t) - \eta \cdot \frac{\partial e}{\partial \mu_i} \cdot \frac{\partial \mu_i}{\partial \mu_{A_j^i}} \cdot \frac{\partial \mu_{A_j^i}}{\partial \sigma_j^i} \\ &= \sigma_j^i(t) - \eta \cdot (y_p - \hat{y}_p) \cdot \left(\frac{(\omega_i(t) - \hat{y}_p)}{\sum_{k=1}^{n} \mu_k}\right) \\ &\quad \cdot \mu_i \cdot \left(\frac{\left(x_j - c_j^i(t)\right)^2}{\left(\sigma_j^i(t)\right)^3}\right) \end{aligned} \tag{5}$$

$$\omega_i(t+1) = \omega_i(t) - \eta \cdot \frac{\partial e}{\partial \omega_i} = \omega_i(t) - \eta \cdot \left(\frac{\mu_i}{\sum_{k=1}^{n} \mu_k} \right) \cdot (y_p - \hat{y}_p) \quad (6)$$

where y_p is the actual output for the pth input–output data, $\hat{y}_p$ is the output of the NFIS for the pth input–output data, and η is a learning rate.

B. Mobility Learning and Prediction System

We have devised a mobility learning and prediction system to predict the probable location of users. The mobility learning and prediction system expresses user mobility as the movement velocity and direction. It predicts a future location from the movement factors of current and past locations. Let S_k^t be the state k at time t of the movement of users. S_k^t represents the movement factors that determine the next location to which users will move. In this paper, $S_k^t = (v_k^t, \theta_k^t)$ such that $k \leq K$ and $t \in T$. Here, K is the number of states in the state space, T is the set of time, v_k^t is the kth movement velocity at time t, and θ_k^t is the kth movement direction at time t. We assume that a movement state can be changed when a period of time τ elapses.

Using the movement states defined above, the movement function which maps current and past movement states into a future movement state can be denoted by

$$\text{NFIS}\left(S_k^t, S_k^{t-1}, \ldots, S_k^{t-(h-1)}\right) \mapsto S_k^{t+1} \quad (7)$$

where NFIS is a neuro-fuzzy inference system which is based on the neuro-fuzzy model described in Section IV-A and h is the number of current and past movement states. Equation (7) uses the current and past movement states $S_k^t, S_k^{t-1}, \ldots, S_k^{t-(h-1)}$ to predict the future movement state S_k^{t+1}. $S_k^t, S_k^{t-1}, \ldots, S_k^{t-(h-1)}$ represents the location history which includes the movement patterns accumulated in the previous days or months. They are used for the input of the NFIS. The output of (7) is the future movement state inferred by the NFIS. The probable location of users is obtained from the velocity and direction of the future movement state S_k^{t+1}.

In order to construct fuzzy rulebase from movement states for several days or months, we use clustering-based techniques [14] that can automatically generate fuzzy rules according to the degree of similarity of movement states. The clustering-based techniques partition movement states into some clusters so that the similarity within a cluster is larger than that within others. Each cluster has a cluster center. The cluster center is used as the central value of the fuzzy membership function in the antecedent part of a fuzzy rule. Let $\vec{R}_i (= [R_i^1, R_i^2, \ldots, R_i^h])$ be the ith cluster center among M clusters and $\vec{S}_k^t (= [S_k^t, S_k^{t-1}, \ldots, S_k^{t-(h-1)}])$ be the kth movement state vector at time t. In order to measure a similarity degree between $\vec{R}_i$ and $\vec{S}_k^t$, a distance function can be defined by

$$D_i = \left| \vec{S}_k^t - \vec{R}_i \right| \quad (8)$$

where D_i is a distance between $\vec{R}_i$ and $\vec{S}_k^t$. In (8), as $\vec{R}_i$ and $\vec{S}_k^t$ become more similar, D_i grows smaller. Then, it is necessary to determine the degree of similarity between $\vec{R}_i$ and $\vec{S}_k^t$. To determine this similarity, we define a radius r. By using (8) and the radius r, a similarity degree between a movement state vector and each cluster is defined as follows: if $D_i \leq r$, the similarity is high, otherwise it is low.

Based on this criterion, the movement state vector of users can be subdivided into three classes:

- not defining the movement state vector as a cluster (fuzzy rule);
- defining the movement state vector as a cluster;
- defining the movement state vector as a cluster, but considering the movement state vector as an unnecessary cluster.

The first class indicates that a movement state vector is added to fuzzy rulebase as a new cluster (fuzzy rule) because the movement state vector does not correspond to any cluster among predefined ones. In this case, the antecedent and consequence of a new fuzzy rule are organized as follows:

$$\mu_{A_j^i}(x_j) = \exp\left[-\frac{1}{2} \cdot \left(\frac{x_j - S_k^{t-(j-1)}}{r} \right)^2 \right], \qquad \omega_i = S_k^{t+1} \quad (9)$$

where $S_k^{t-(j-1)}$ and r correspond to the central value (c_j^i) and the variance (σ_j^i) of a fuzzy membership function, respectively, and S_k^{t+1} corresponds to the real number (ω_i) of consequence part.

The second class indicates that a movement state vector has a high degree of similarity to several clusters among predefined ones. In this case, by increasing the importance of the clusters, the NFIS results in a more exact prediction when a similar movement state vector with predefined movements states appears in the movement path of users. In order that the movement state can affect the real value of the consequence part in fuzzy rules, ω_i is updated as follows:

$$\omega_i = \frac{\sum_{k=1}^{m_i} S_k^{t+1}}{m_i} \quad (10)$$

where m_i is the number of movement state vectors within the ith cluster. Using (10), we get the updated weight which corresponds to the mean of future movement states within a cluster.

The third class indicates that a movement state vector had been previously defined as a fuzzy rule, but is currently considered unnecessary. This means the wrong reflection of the latest movement state in fuzzy rulebase. This situation happens when the past movement of users differs from their ordinary movement patterns. The cluster (fuzzy rule) made from this situation causes incorrect predictions. So, the unnecessary fuzzy rule should be eliminated from fuzzy rulebase according to an appropriate mobile situation. In other words, those fuzzy rules that suitably reflect the latest movement state should be stored in

the fuzzy rulebase. Thus, it is necessary for a criterion to determine the elimination of an unnecessary fuzzy rule from the fuzzy rulebase. Such a criterion is given as

$$L(age_i) = \exp[-\alpha \cdot age_i]$$
$$\text{Rule } i = \begin{cases} young & \text{if } L(age_i) \geq \beta \\ old & \text{if } L(age_i) < \beta \end{cases} \quad (11)$$

where

- age_i the age of the ith fuzzy rule;
- α the parameter which controls the age of a fuzzy rule;
- β the parameter which determines the elimination of a fuzzy rule.

The age of a fuzzy rule is updated as follows. If a distance between a current movement state vector and a cluster center is less than the radius, the age of a fuzzy rule becomes zero. On the other hand, if the distance is larger than the radius, the age of a fuzzy rule is increased by one. To determine how much the movement state of users affects fuzzy rules, we introduce the parameter β. According to the value of β, fuzzy rules are subdivided into two groups: young fuzzy rules and old fuzzy rules. Since the young fuzzy rules suitably reflect the latest movement state, it is desirable that they be used as fuzzy rules. The old fuzzy rules are built from the movement state which is found just once in the movement path of users during β. There is a very strong possibility that the old fuzzy rules bring about incorrect prediction. Thus, the old fuzzy rules should be eliminated from the fuzzy rulebase. In our system, if the age of a fuzzy rule is more than β (the old fuzzy rules), this fuzzy rule is eliminated as an unnecessary fuzzy rule.

So far, we have described the mobility learning and prediction system based on the NFIS for modeling and predicting the movement patterns of users. The next section proposes the restoration scheme by mobility learning and prediction.

V. Restoration of Mobility Database by Mobility Learning and Prediction

In this section, we describe the restoration scheme to provide service availability to users even after the failure of mobility databases. Our restoration scheme consists of two primary operations: failure-free operations and failure recovery operations. Fig. 4 shows the overall structure for the restoration of mobility databases under our scheme. The following subsections present a detailed description for each operation.

A. Failure-Free Operations

A single fault model is assumed in our restoration scheme. The mobility databases are assumed to be fail-stop and so the lost location records are restored immediately after a failure by the mobility learning and prediction system described in Section IV.

During failure-free operations, our restoration scheme learns the users' moving trajectory. We consider current movement direction and velocity as user mobility parameters. Basically, our restoration scheme assumes that users inform mobility databases of their current location every time interval τ_r. Also, mobility databases update their location records when users originate a call, when a handover occurs, and so on. These assumptions reduce wireless link consumption. The movement patterns based on the format (*current exact location, current time, current movement direction and velocity*) are gathered at the MSC and periodically transferred to the HLR via the VLR. Then, the NFIS learns the movement patterns using the latest location information.

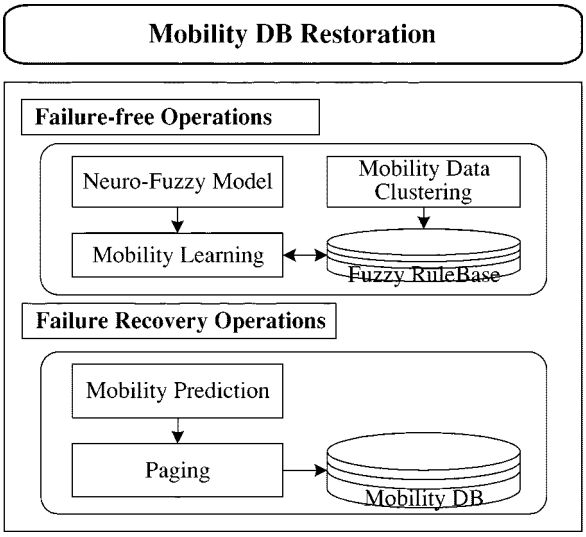


Fig. 4. The overall structure of our restoration scheme.

Here, we describe an overall algorithm for mobility learning and prediction. The algorithm performs a learning procedure for the movement patterns of users in order to construct more an elaborate fuzzy rulebase. It also performs a prediction procedure to predict the future location. The various steps are described as follows.

Algorithm 1: Mobility Learning and Prediction

Let $\vec{S}_k^t (= [S_k^t, S_k^{t-1}, \ldots, S_k^{t-(h-1)}])$ be the movement state vector containing the current and past movement information at time t ($k = 1, 2, \ldots, K$, $t = 1, 2, \ldots, T$). We predict the future movement state S_k^{t+1} from $\vec{S}_k^t$ at time t by the NFIS.

Step 1: Starting with the first movement state vector $\vec{S}_1^t$ and the first future movement state S_1^{t+1}, establish a cluster center $\vec{R}_1$ at $\vec{S}_1^t$, set $\omega_1 = S_1^{t+1}$, and assign zero to the age of the cluster age_1. Select a radius r.

Step 2: Suppose that, when we consider the kth movement state vector $\vec{S}_k^t$, there are M clusters with centers at $\vec{R}_1, \vec{R}_2, \ldots, \vec{R}_M$. Compute the similarity of $\vec{S}_k^t$ to these M cluster centers by (8) and let the highest similarity be $D_{k'}$, that is, the nearest cluster to $\vec{S}_k^t$ is $\vec{R}_{k'}$ ($k' = 1, 2, \ldots, M$).

a) If $D_{k'} > r$, establish $\vec{S}_k^t$ as a new cluster center $\vec{R}_{M+1} = \vec{S}_k^t$, set $\omega_{M+1} =$

S_k^{t+1}, and assign zero to the age of the new cluster age_{M+1}.

b) If $D_{k'} \leq r$, update $\omega_{k'}$ by Equation (10) and set $age_l = age_l + 1$ for $l = 1, 2, \ldots, M$ with $l \neq k'$.

Step 3: Eliminate unnecessary clusters by (11).

Step 4: Compute the future movement state $\hat{S}_k^{t+1}$ at the kth movement state vector $\vec{S}_k^t$ by (2).

Step 5: Adjust the parameters c_j^i, σ_j^i, and ω_i as much as the number of learning iterations by (4)–(6).

Step 6: Execute Steps 2–5 iteratively.

B. Failure Recovery Operations

After a failure, mobility databases must restore the same location information as before. Our approach to restore failed mobility databases is based on mobility learning and prediction. When a failure occurs, the mobility learning and prediction system is initiated to predict a user's probable location. Let t_f be the time at which a failure occurs in mobility databases, and $\vec{S}_k^{t_f}$ be the kth movement state vector at time t_f. After time t_f, the movement state vector at future time t cannot employ current and past movement information any more ($t_f < t$). Thus, in our scheme, to predict the kth movement state at current time t_p ($t < t_p$), future movement state vectors are recursively constructed with the location information predicted by NFIS up to time t_p. The kth movement state vector at time t, $\vec{S}_k^t$, is organized as follows:

$$\vec{S}_k^t = \begin{cases} [\hat{S}_k^t, \hat{S}_k^{t-1}, \ldots, \hat{S}_k^{t_f+1}, \\ \quad S_k^{t_f}, S_k^{t_f-1}, \ldots, \\ \quad S_k^{t-(h-1)}], & \text{if } t_f < t < t_f + h \\ \left[\hat{S}_k^t, \hat{S}_k^{t-1}, \ldots, \hat{S}_k^{t-(h-1)}\right], & \text{if } t \geq t_f + h \end{cases} \tag{12}$$

where S_k^* and $\hat{S}_k^*$ represent the movement states included with $\vec{S}_k^{t_f}$ and the predicted movement states by NFIS, respectively.

Using (12), we can get different movement state vectors whether or not S_k^* is included with the movement state vector at time t. $\vec{S}_k^t$ is used for predicting the movement state $\hat{S}_k^{t+1}$ at time $t+1$ and $\hat{S}_k^{t+1}$ is used for organizing the movement state vector $\vec{S}_k^{t+1}$ at time $t+1$. This process is repeated up to time t_p by NFIS in order to predict the movement state $\hat{S}_k^{t_p}$ at time t_p. The prediction information derived in this way is used as the first paging cell to locate lost users. This prediction information represents the probable cell of users at the failure time of the mobility databases. The accuracy of predicting the probable cell location of users depends on the time interval between t_f and t_p. In general, users may move farther and farther away from their location at time t_f, as the time interval grows longer. However, if the mobility learning and prediction system correctly learns user mobility during failure-free operations, it can predict the probable cell of users at time t_p, with a high rate of accuracy. In this case, only one paging is needed to locate lost users after a failure. So, as the rate of accuracy in predicting location increases, paging cost goes down.

Our restoration scheme consists of the HLR failure restoration procedure and the VLR failure restoration procedure. Fig. 5(a) and (b) shows the overall flow of the failure restoration of two databases, the HLR and the VLR, respectively. Algorithms 2 and 3 also show the restoration step of two mobility databases after a failure.

Algorithm 2: **HLR Failure Restoration by Mobility Prediction**

Step 1: The failed HLR sends a restoration initiation message to all VLR's in order to gather the lost location information of users.

Step 2: Each VLR that receives the restoration initiation message is queried to search the lost location information of users.

Step 3: The failed HLR receives an exact location information of users from all VLR's. The HLR is reconstructed using the location information.

Algorithm 3: **VLR Failure Restoration by Mobility Prediction**

Step 1: The failed VLR sends a restoration initiation message to the HLR in order to gather the lost location information of users.

Step 2: The HLR that receives the restoration initiation message predicts the location of the requested users.

Step 3: The failed VLR receives the predicted location from the HLR.

Step 4: The predicted location is paged to locate users.

Step 4.1: A cell corresponding to the predicted location is paged. If any response from the paged user is returned, the location information for the user is reconstructed.

Step 4.2: If there is no response within a specific time, the adjacent cells of the predicted location are paged.

Step 4.3: If there's still no response from the adjacent cell paging, entire cells within a RA are paged.

Step 5: The failed VLR receives an exact location information of the paged users. The VLR is reconstructed using the location information. If any response from the paged users is not returned after three successive pagings, the failed VLR waits until the users originate a call.

The checkpointing scheme is a passive restoration strategy in which the location information of users is backed up into a stable

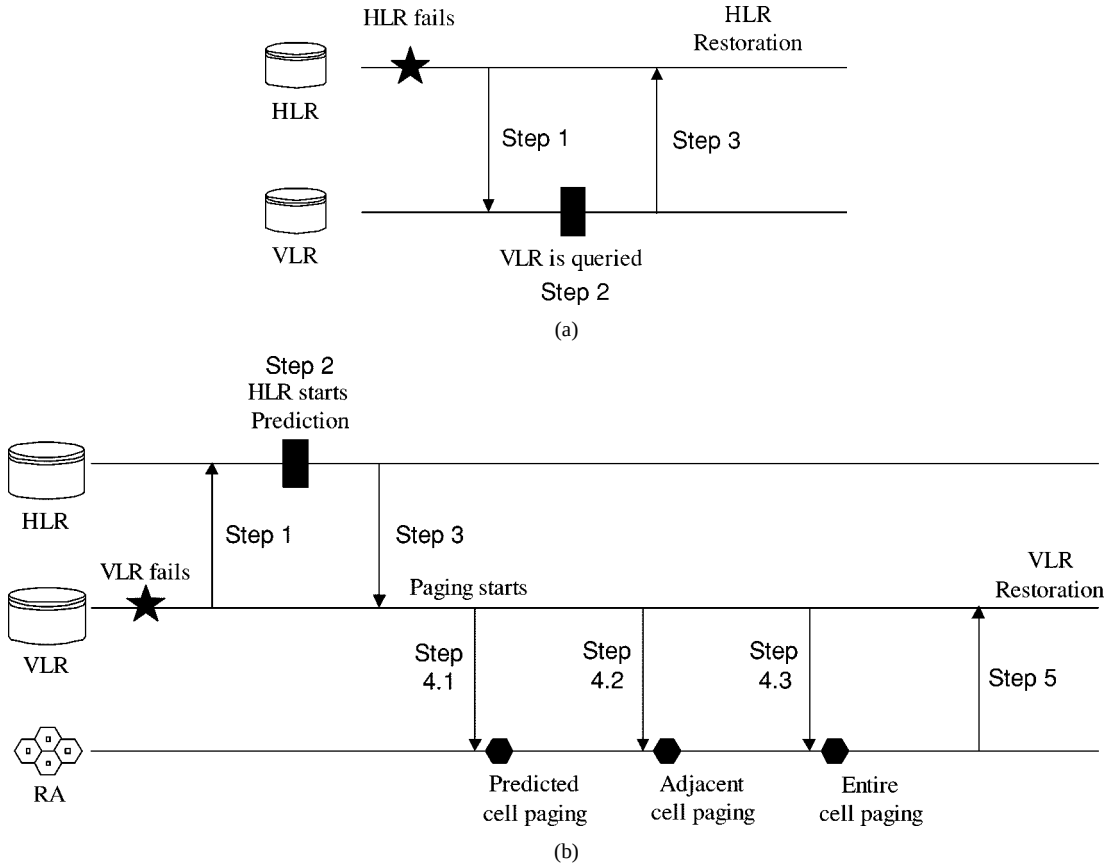


Fig. 5. Mobility database restoration by mobility prediction: (a) HLR and (b) VLR.

storage during failure-free operations. Because failed databases are restored from storage, restored location information may be obsolete. So, restoration efficiency under the checkpointing scheme is very sensitive to user mobility. On the other hand, our scheme is an aggressive restoration strategy in which the lost location information is recovered by predicting the users' movements at the failure time. It predicts the probable location of users after a failure and uses this predicted location information as a starting paging cell. Since the predicted location information is based on the movement patterns of users during failure-free operations, we expect it to be more accurate and therefore entail less number of paging.

VI. Performance Evaluation

In this section, we evaluate the performance of our restoration scheme through simulations. Our simulations are performed with the mobility data, which are approximated to users' real movements. We first describe how to generate mobility data. Then, to show an efficiency of our restoration scheme, we compare our scheme with a restoration scheme that uses a periodic checkpointing scheme [13].

A. Mobility Data Generation

Mobility data could be collected from real mobile environments. However, these data are too complicated to be collected directly from the real mobile environments. The procedure to generate mobility data is presented here. This procedure consists of a user mobility model and user mobility data generation. The user mobility model presents a method of modeling users' movement behavior in terms of their velocity and direction. Based on the user mobility model, mobility data are generated.

Users have their own movement behavior. The movement behavior is defined as the act of moving from one given geographical location to another, over a spatial dimension. According to their movement behavior, users have specific movement patterns. Many previous researches [15]–[17], have used some version of a random-walk mobility model. However, the random-walk mobility model fails to take into account certain aspects of movement behavior, such as users' movement patterns. In [18], the traveling demand model is proposed to represent observed travel patterns by travel mode. This is based on transportation planning and traffic engineering models that describe and predict travel demand, typically concentrating on urban areas [19]. In this model, it is assumed that individual travelers make travel choices which are "best" for them. Based on the traveling demand model, we extract the mobility data in terms of users' movement velocity and direction.

The velocity of users can be interpreted in two ways: using mean velocity and an arbitrary velocity variation. In a real situation, when users move at a mean velocity, the velocity variation would be nearly zero. If users move without any periodicity for a velocity per period of time, they move at a random velocity in the range of a mean velocity. Thus, at time $t+1$, users have the following velocity:

$$v_{t+1} = \overline{v}_t + \tilde{v}_t \tag{13}$$

TABLE I
USER CLASSES AND PARAMETER VALUES

Velocity			*Direction*		
Class	Meaning	Value ($\tilde{v}_c$)	Class	Meaning	Value ($\vec{p}$)
A	regular	(5, 25)	X	almost straight	[0.0 0.1 0.8 0.1 0.0]
B	moderate	(10, 40)	Y	moderate	[0.1 0.2 0.4 0.2 0.1]
C	fluctuating	(20, 60)	Z	staggering	[0.2 0.2 0.2 0.2 0.2]

where $\overline{v}_t$ is a mean velocity at time t; $\tilde{v}_t$ is the variation of a mean velocity at time t and it is selected using a uniform distribution in the velocity variation range $[-\tilde{v}_c, \tilde{v}_c]$. In (13), as the velocity variation range increases, the velocity grows more random.

On the assumption that users move toward a destination along their route, users are differentiated using their target direction toward the destination and a variation of the target direction. Thus, the movement direction at time $t+1$, θ_{t+1} can be defined by

$$\theta_{t+1} = \overline{\theta}_t + \frac{2\pi}{d} \cdot k \tag{14}$$

where

$\overline{\theta}_t$ current direction toward a destination at time t;
d number of movement directions;
k movement direction index and is selected with the probability p_k, where $k \in \{0, 1, \ldots, d-1\}$.

In (14), if the probability of the direction k toward a destination p_k is larger than that of other directions, users would move into a destination without abrupt direction changes. On the other hand, if all probabilities are equal, users would move into a destination with a random direction.

Through the mobility model described above, users are categorized into several movement types. In this paper, we will make the mobility data per each movement type and employ them in evaluating the performance of our restoration scheme.

B. Simulation Environments

When carrying out the simulations, we assume that users move to any cell in the hexagon cell environments along unknown movement paths according to their movement properties. The assumed total number of cells is 256. Each VLR has the same number of cells. It is also assumed that the radius of a cell is 500 m. Within these geographical parameters, the location information of users is measured at every 0.0835 h for 30 days.

In order to extract mobility data, we classify users into nine user types by the combination of three kinds of velocities (A, B, and C) and three kinds of directions (X, Y, and Z). Table I shows the user classes for velocity and direction. Among nine user types, the users of the type AX always move toward a destination, keeping a mean velocity and direction. On the other hand, the users of the type CZ display a random velocity and direction in their movement paths.

For the sake of simplicity, we assume that $\overline{v}_t = 5$ km/h and $d = 5$ for all time t. Thus, users have different movement paths according to the velocity variation range ($\tilde{v}_c$) and the direction probability vector ($\vec{p} = [p_0, p_1, \ldots, p_{d-1}]$ such that $\sum_{k=1}^{d-1} p_k = 1$). Parameters values for each user class are also shown in Table II.

TABLE II
SIMULATION PARAMETERS

Parameter Description	Values
The number of mobility data (*days*)	30
The number of current and past movement states (h)	4
The change period of movement states (τ)	0.0835
Radius (r)	0.3
Learning rate (η)	0.05
The number of learning iterations	50
The control parameter for the age of fuzzy rules (α)	0.1
The remove parameter for unnecessary fuzzy rules (β)	0.005

With mobility data for each user type, we evaluated the performance of our restoration scheme. During failure-free operation, NFIS learns the movement patterns of users with the mobility data. For the sake of simplicity, it was assumed that the failure of mobility databases happens after it passes 30 days. Then, a hit ratio between a predicted location and actual location was collected. These results were compared to those gathered using the periodic checkpointing scheme.

The performance of a restoration scheme with periodic checkpointing is shown in [13]. After failed mobility databases are restored from a stable storage, lost users are paged to find their exact locations. If lost users are not found by the paging, entire paging within an RA should be performed. This renders the checkpointed location information obsolete because users have moved away from the checkpointed location. Thus, the performance of a restoration scheme with periodic checkpointing can be determined by a hit ratio between a restored location and the probable location of lost users after a failure. Accordingly, it is shown in [13] that users' residence time in an RA follows an exponential distribution with mean $1/\lambda_m$, where λ_m is a mean probability that users move out of the RA. Under this observation, the probability P_{T_V} which users move in an RA at a time between 0 and T_V before checkpointing and move out of the RA at the time T_V after a failure is derived as

$$P_{T_V} = \frac{1}{T_V} \int_0^{T_V} e^{-\lambda_m (T_V - s)}\, ds = \frac{1 - e^{-\lambda_m \cdot T_V}}{T_V \cdot \lambda_m} \tag{15}$$

where T_V is an average checkpoint interval. From (15), we can deduce that users do not move out of the RA of a checkpointed location after a failure occurs at mobility databases. It should be noted that, as the probability becomes lower, the paging area to locate lost users grows wider. In our simulations, (15) was used to compare the performance of our restoration scheme with that of a restoration scheme with periodic checkpointing. For demonstration purposes, we assume that $\lambda_m = 0.4$.

The performance of our restoration scheme is determined by the degree of accuracy of actual location vis-à-vis predicted location by the NFIS. If our scheme can predict the probable location of users after a failure, only one paging is needed to locate the users. This dramatically decreases paging cost. In order to analyze the performance of our restoration scheme, we evaluate how many location records are exactly restored after the failure of mobility databases elapses. As the criterion of the performance, we use an exact cell hit ratio (ECHR) and an adjacent cell hit ratio (ACHR). The ECHR is defined as a ratio of the correctly restored location information by a single paging to the

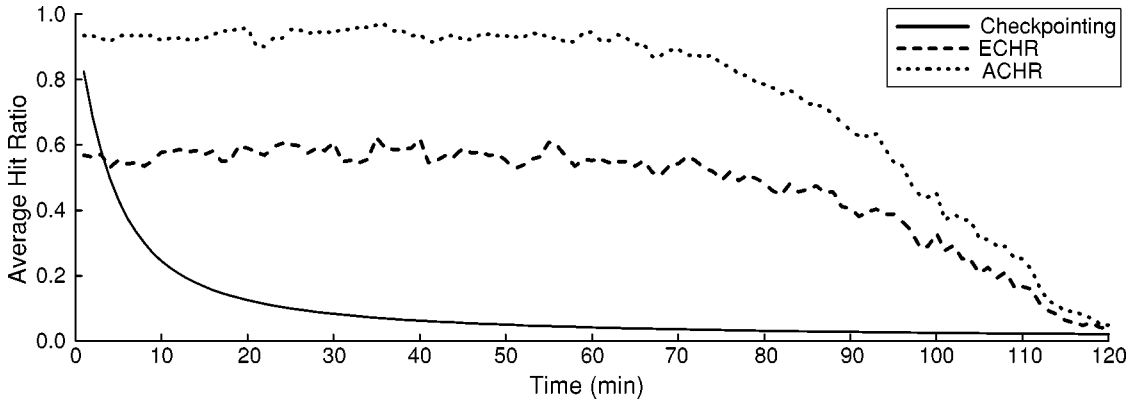


Fig. 6. Average hit ratio as a time after a failure elapses.

total number of lost location information at a future time after a failure. The ACHR is defined as a ratio of correctly restored location information by the paging of adjacent cells around predicted cell to the total number of lost location information at a future time after a failure.

C. Simulation Results

Fig. 6 shows the average hit ratio of all user types as a time after the failure of mobility databases elapses. In this figure, the solid line indicates the average hit ratio of the checkpointing scheme. The dotted line and the dashed line indicate the average ECHR and ACHR of our restoration scheme, respectively. The checkpointing scheme shows a steep descent as a time elapses. Besides, we can see that the curve sharply goes down at the initial stage of the failure. On the other hand, our restoration scheme maintains a high average hit ratio for all user types until a time after a failure elapses in the range from 50 to 70 min. Even though the ECHR is somewhat low, the ACHR, which is a hit ratio for the adjacent cells of a predicted cell, is considerably high. Thus, this figure shows that the performance of the checkpointing scheme is very sensitive to the time elapsed after a failure. On the other hand, the performance of our restoration scheme does not depend on the elapsed time. Even though a lot of time has elapsed after a failure, our restoration scheme maintains a high average hit ratio.

From the results of Fig. 6, we observe that our restoration scheme can reduce the cost needed to restore location records in the presence of the failure of mobility databases. This reduction indicates that our restoration scheme is able to predict the probable location of lost users after a failure. In addition, our restoration scheme can sufficiently locate lost users using less paging, resulting in a dramatic decrease in paging costs. On the other hand, the checkpointing scheme can locate lost users by paging an entire RA after the first unsuccessful paging. In practical systems, an RA consists of dozens or hundreds of cells. So, the checkpointing scheme would page a lot of cells to locate lost users, as compared to our restoration scheme.

We now examine the effect of the users' random movement on the ECHR. Fig. 7 shows the ECHR of nine user types as the elapsed time after a failure. It shows the probability that users exactly reside on the predicted cell by the NFIS after a failure occurs. In Fig. 7, the users of the type AX have the highest hit ratio from among nine user types because they have fairly regular movement patterns. The ECHR of the users is in the range of 80%–90% up to a failure duration time of roughly 110 min. On the other hand, the users of the type CZ who have somewhat random movement patterns have the lowest hit ratio. Even though the ECHR of this type is not high as that of the other types, there is no steep slope during an initial stage after a failure. We can see in Fig. 7 that the users of this type maintain a uniform ECHR up to a failure duration time of roughly 60 min.

Upon analyzing the ECHR of Fig. 7, we can find that, as the variation of velocity and direction increases, the ECHR decreases. Nevertheless, for all user types, there is no steep slope during the initial stage of a failure. This means that the location records of lost users can be restored with uniform paging costs in the initial state of a failure regardless of users' random movements.

Our restoration scheme is based on user mobility learning and prediction to locate lost users after a failure. To get a higher restoration accuracy, it would require a number of learning iterations for the past movement patterns of users during failure-free operations. The learning rules [see (4)–(6)] are iteratively performed as much as needed in order to optimize the parameters c_j^i, σ_j^i, and ω_i. This can be an overhead for our restoration scheme. However, excessive learning for the parameters may recklessly learn noise patterns in movement states, and moreover, it requires a great deal of learning time. So, even though the learning phase is performed using many learning iterations, the performance of our restoration scheme should not always be enhanced in proportion to the increase of the learning iterations. On the other hand, scarce learning for the parameters may require less learning time, but suitable values for the parameters cannot be derived. Thus, it is important to choose the number of learning iterations suitably so that, during failure-free operations, learning time is reduced as much as possible. In our simulation, we used the number of learning iterations at 50 and discovered that this value is quite suitable.

VII. Conclusion

We have proposed an approach based on mobility learning and prediction to failure restoration for PCS mobility databases. Whenever users register a new location, their movement patterns are learned by an NFIS. When a failure occurs in mobility databases, the location information of users is predicted by the

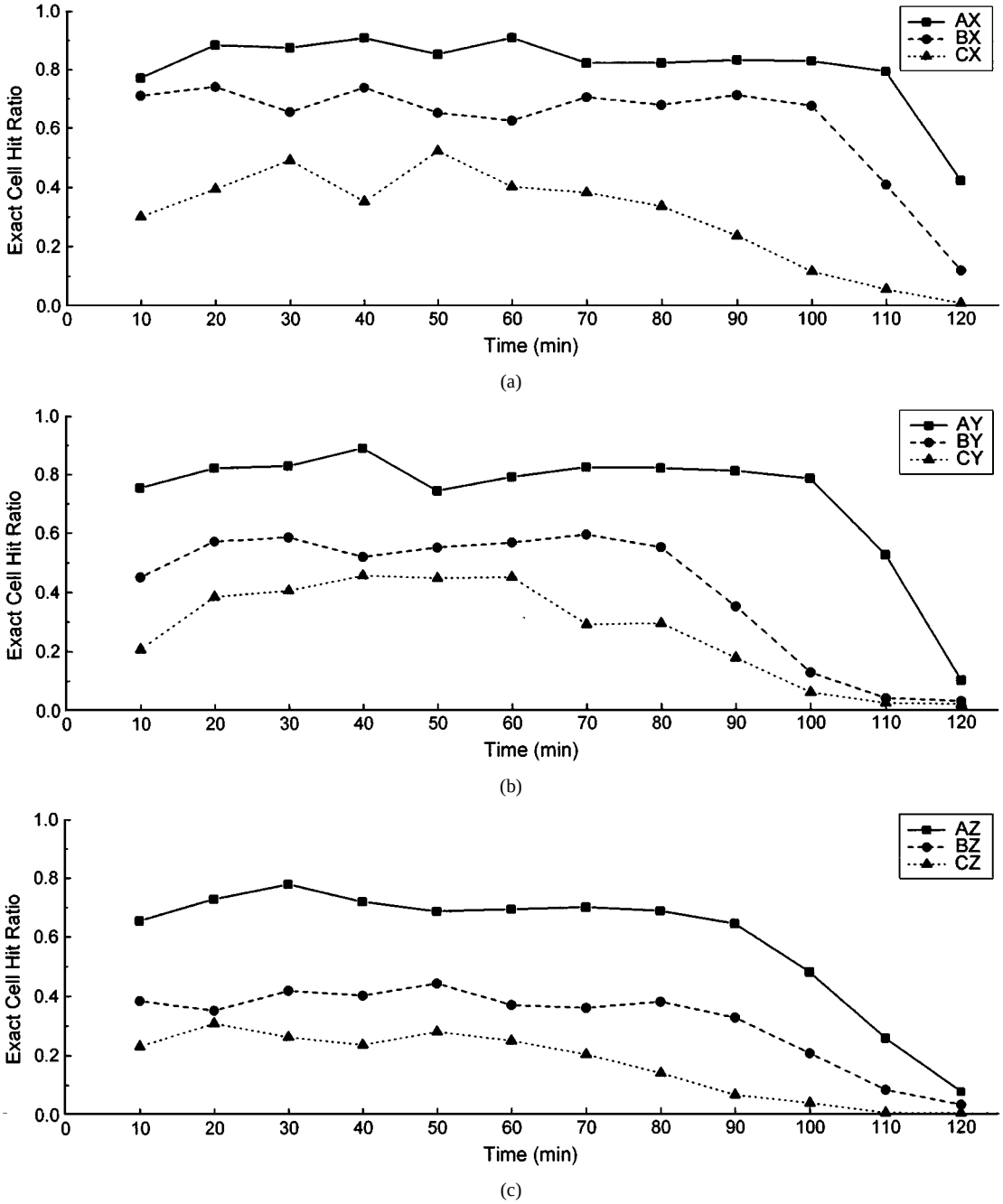


Fig. 7. ECHR for all users. (a) Almost straight direction. (b) Moderate direction. (c) Staggering direction.

NFIS. The predicted location is used to determine the probable location of lost users after a failure. Unlike previous approaches using the checkpointing scheme, our restoration scheme can reduce the cost needed to restore the location records of lost users after a failure by using less paging.

The simulations showed that mobility prediction provides considerably high restoration accuracy. While the checkpointing scheme becomes less accurate as the checkpoint interval becomes larger, our restoration scheme with mobility learning and prediction capability maintains a considerably high restoration rate regardless of the time elapsed after the failure of mobility databases. We also examined through simulations an effect of users' random movements. The results showed that the performance of our restoration scheme is somewhat sensitive to the variation of movement velocity and direction, but it can restore the location records of lost users with a high restoration rate after a failure.

In our restoration scheme, the NFIS replaces the role of the checkpoint as mobility learning and prediction. Thus, our restoration scheme has no backup process and therefore fewer failure-free operation costs. In addition, it needs no storage space to store the checkpoints. We conclude that our scheme produces a more accurate restoration rate than the checkpointing scheme, at less delay and lower cost.

In our paper, we applied the mobility prediction to mobility database restoration in PCS networks. The mobility prediction would have applicability to various areas, such as service preallocation, resource pre-assignment, seamless handovers, con-

text transfers, and so on. In particular, IP networking like IETF SeaMoby [20] seems to need this mobility prediction to provide mobile users with a high quality of services.

REFERENCES

[1] T. Imielinski and H. F. Korth, *Mobile Computing*. Norwell, MA: Kluwer, 1996.

[2] E. Pitoura and G. Samaras, *Data Management for Mobile Computing*. Norwell, MA: Kluwer, 1998.

[3] I. F. Akyildiz and J. S. M. Ho, "A mobile user location update and paging mechanism under delay constraints," *Comput. Commun. Rev.*, vol. 25, no. 4, pp. 244–255, Oct. 1995.

[4] B. Awerbuch and D. Peleg, "Online tracking of mobile users," *J. ACM*, vol. 42, no. 5, pp. 1021–1058, Sept. 1995.

[5] A. Bar-Noy and I. Kessler, "Tracking mobile users in wireless communication networks," *IEEE Trans. Inform. Theory*, vol. 39, pp. 1877–1886, Nov. 1995.

[6] S. Tabbane, "An alternative strategy for location tracking," *IEEE J. Select. Areas Commun.*, vol. 13, pp. 880–892, June 1995.

[7] EIA/TIA, "Cellular radio-telecommunications intersystems operations: Automatic roaming," Tech. Rep. IS-41.3-B, EIA/TIA, 1991.

[8] ETSI/TC, "Restoration procedures, version 4.2.0," Tech. Rep. Recommendation GSM 03.07, ETSI/TC, 1993.

[9] G. Liu and G. Maguire, Jr., "A class of mobile motion prediction algorithms for wireless mobile computing and communications," *Mobile Networks and Applications*, vol. 1, pp. 113–121, 1996.

[10] T. Liu, P. Bahl, and I. Chlamtac, "Mobility modeling, location tracking, and trajectory prediction in wireless ATM networks," *IEEE J. Select. Areas Commun.*, vol. 16, pp. 922–936, Aug. 1998.

[11] J. Chan and A. Seneviratne, "A practical user mobility prediction algorithm for supporting adaptive QoS in wireless networks," in *Proc. IEEE Int. Conf. on Networks (ICON '99)*, Sept. 1999, pp. 104–111.

[12] Y.-B. Lin, "Failure restoration of mobility databases for personal communication networks," *ACM-Baltzer J. Wireless Networks*, vol. 1, no. 3, pp. 365–372, 1995.

[13] T.-P. Wang, C.-C. Tseng, and W.-K. Chou, "An aggressive approach to failure restoration of PCS mobility databases," *Mobile Comput. Commun. Rev.*, vol. 1, no. 3, pp. 21–28, Sept. 1997.

[14] J.-S. R. Jang, C.-T. Sun, and E. Mizutani, *Neuro-Fuzzy and Soft Computing*. Englewood Cliffs, NJ: Prentice-Hall, 1997.

[15] R. Thomas, H. Gilbert, and G. Mazziotto, "Influence of the movement of mobile station on the performance of the radio cellular network," in *Proc. 3rd Nordic Seminar*, Sept. 1988.

[16] D. Hong and S. S. Rappaport, "Traffic model and performance analysis for cellular mobile radio telephone systems with prioritized and nonprioritized handoff procedure," *IEEE Trans. Veh. Technol.*, vol. VT-35, pp. 77–91, Aug. 1986.

[17] D. Lam, D. C. Cox, and J. Widom, "Teletraffic modeling for personal communications services," *IEEE Commun. Mag. Special Section on Teletraffic Modeling*, vol. 35, pp. 79–87, Feb. 1997.

[18] J. Scourias and T. Kunz, "A dynamic individualized location management algorithm," in *Proc. 8th IEEE Int. Symp. on Personal, Indoor, and Mobile Radio Communications*, Sept. 1997, pp. 1004–1008.

[19] N. Oppenheim, *Urban Travel Modeling*. New York: Wiley, 1994.

[20] P. Calhoun and O. Lewkowetz, "Problem Description: Reasons for Performing context transfers", to be published.

Joon-Min Gil (S'99–M'00) received the M.S. degree in computer science and the Ph.D. degree in computer science and engineering from Korea University, Seoul, Korea, in 1996 and 2000, respectively.

From 1998 to 2000, he was a Research Associate in the Institute of Basic Science at Korea University. He is currently a Visiting Research Associate in the Department of Computer Science at the University of Illinois, Chicago. His recent research interests include distributed and mobile computing, moving object databases, fuzzy systems, and neural networks.

Chan Yeol Park received the B.S. degree in mathematics and the M.S. and Ph.D. degrees in computer science and engineering from Korea University, Seoul, Korea, in 1993, 1995, and 2000, respectively.

He is currently working for ECO Corporation in Seoul, Korea as a Manager of an R&D team. His recent research interests include distributed and mobile computing and fault tolerances.

Chong-Sun Hwang received the M.S. degree in mathematics from Korea University, Seoul, Korea, in 1970 and the Ph.D. degree in statistics and computer science from the University of Georgia, Athens, in 1978.

From 1978 to 1980, he was an Assistant Professor at South Carolina Lander State University. He is currently a Full Professor in the Department of Computer Science and Engineering at Korea University, Seoul, Korea. Since 1995, he has been a Dean in the Graduate School of Computer Science and Technology at Korea University. His recent research interests include distributed systems, distributed algorithms, and mobile computing systems.

Doo-Soon Park (M'90) received the M.S. degree in computer science from Chungnam National University, Daejon, Korea, in 1983 and the Ph.D. degree in computer science from Korea University, Seoul, Korea, in 1988.

From 1992 to 1993, he was a Visiting Professor in the Center of Supercomputing Research and Development at the University of Illinois at Urbana-Champaign. Since 1985, he has been a Professor in the Division of Information Technology Engineering at Soonchunhyang University, Asan, Korea. He is currently a Dean of Computer Education at Soonchunhyang University. His recent research interests include parallel processing compiler, parallel and mobile computing, and computational theory.

Dr. Park is a member of the ACM.

Jin Gon Shon received the B.S. degree in mathematics and the M.S. and Ph.D. degrees in computer science and engineering from Korea University, Seoul, Korea, in 1984, 1988, and 1991, respectively.

Since 1991, he has been with the Department of Computer Science, Korea National Open University, Seoul, Korea, where he is a Professor and former Department Chairman, working in the areas of distributed and mobile computing, fault-tolerant systems and ITLET (Information Technology for Learning, Education, and Training) as a member of JTC1/SC36.

Young-Sik Jeong received the M.S. and Ph.D. degrees in computer science and engineering from Korea University, Seoul, Korea. in 1989 and 1993, respectively.

He is currently an Associate Professor in the School of Computer and Communication Engineering at Wonkwang University, Iksan, Korea. His recent research interests include distributed and parallel computing, mobile computing, and hypermedia systems.