Wireless Personal Communications **28:** 17–33, 2004.

Replicated Object Management with Periodic Maintenance in Mobile Wireless Systems

DING-CHAU WANG [1], ING-RAY CHEN [2], CHIH-PING CHU [3] and I-LING YEN [4]

[1] *Department of Information Management, Southern Taiwan University of Technology, Tainan, Taiwan*
E-mail: wangdc@csie.ncku.edu.tw
[2] *Department of Computer Science, Virginia Tech., 7054 Haycock Road, Falls Church, VA 22043, U.S.A.*
E-mail: irchen@cs.vt.edu
[3] *Department of Computer Science and Information Engineering, National Cheng Kung University, Tainan, Taiwan*
E-mail: chucp@csie.ncku.edu.tw
[4] *Department of Computer Science, University of Texas at Dallas, Richardson, TX 75083-0688, U.S.A.*
E-mail: ilyen@utdallas.edu

Abstract. In this paper we analyze periodic maintenance strategies for managing replicated objects in mobile wireless environments. Under periodical maintenance strategies, the system periodically checks local cells to determine if a replicated object should be allocated or deallocated in a cell to reduce the access cost. We develop a performance model based on Petri nets that considers the missing-read cost, write-propagation cost and the periodic maintenance cost with the objective to identify optimal periodic maintenance intervals to minimize the overall cost. The analysis results show that the overall cost is high when the user arrival-departure ratio and the read-write ratio work against each other and is low otherwise. In all cases, there exists an optimal periodic maintenance interval that would yield the minimum cost. Further, the optimal periodic maintenance interval increases as the arrival-departure ratio and the read-write ratio work in harmony.

Keywords: data replication, mobile systems, primary copy, periodic maintenance, performance evaluation, Petri net.

1. Introduction

Data replication is a proven technique for improving fault tolerance and performance of distributed and database systems [7]. In particular, determining the optimum number of replicated items and their placement in the system can greatly improve system efficiency. Research has been conducted that analyzes both static and dynamic replicated data management schemes [1, 2, 4, 5, 8–10]. In a static replicated data management scheme, the number of replicas and their placement is pre-determined and fixed. The number or placement of the replicas does not change as user patterns change. In contrast, a dynamic replicated data management scheme adjusts both the number of replicas and their placement in response to the demands on the system. The dynamic approach is particularly relevant to a mobile wireless environment where users are frequently moving from one location to another. This paper concerns dynamic replicated data management in mobile wireless environments.

Wolfson et al. propose a dynamic replication scheme known as adaptive data replication (ADR) [9] where the read-write patterns of an object changes in the network. ADR operates in a distributed environment, where each processor analyzes the read-write demand on an object and makes replica adjustments as opposed to having a centralized processor determine all

replica placement changes. ADR adapts its configuration to handle the changes in the read-write pattern periodically and is proven to converge to the optimal scheme in the steady state within a number of time periods. The issue of optimal periods, however, is not addressed. Sistla et al. consider a data replication scheme [6] in mobile environments where a mobile computer or a cell[1] collects the read/write access information of an object in a window period of k read/write requests, thus forming a class of window(k) data replication schemes. Although the notion of competitiveness is used to analyze the worst-case behavior of the scheme, the issue of what the optimal k value should be is not addressed. Also, the analysis is based on the assumption of a single-stream client-server relationship between the primary copy at the server and the replica copy at the mobile machine to handle all read/write operations of an object.

In general, for an object suppose that N_R and N_W are the numbers of local reads and remote writes performed on a data object, respectively, in the last window period. Also suppose that C_R is the cost saved due to a read if the cell has a replica of the object and C_W is the extra cost incurred due to a write if the cell has a replica of the object locally. Then it is worthwhile to maintain a replica locally if $N_R C_R \geq N_W C_W$. The above formula can be used in combination with constraints such as a maximum number of replicas per object and/or a maximum number of replicas per cell. Shivakumar et al. propose the maintenance of a centralized maximum-flow, minimum cost flow diagram to maximize $N_R C_R - N_W C_W$ while satisfying the imposed constraints [5]. It is suggested that the flow diagram be constructed periodically and analyzed dynamically at run time to determine the placement of replicas in the system. Wu et al. also propose a distributed dynamic replication scheme [10] based on the same formula. In addition to tracking the read/write histories of objects, each site (i.e., base station) also tracks the user's access history, daily schedule, and the user's demand by item with the goal of increasing data availability while reducing response time. This approach uses the changing user profiles to predict the future demands and then reconfigures the replicas in the system to meet those predictions. Although these research efforts consider the need to dynamically alter the number and replacement of replicas in the network, they do not address how often periodic maintenance events should be invoked. The goal of this paper is to analyze periodic maintenance for replicated data management in mobile systems. In particular, it addresses *when* and *how* to use periodic maintenance as a mechanism to optimize the cost of replicated data management.

There are several costs involved in managing replicas of an object in mobile environments. There is the cost of updating data replicas at numerous cells when a write operation occurs. There is the cost of reading a copy from a neighboring cell when the local cell does not contain a replica. There is the cost of allocating/deallocating replicas and adjusting the number of replicas based on usage. The optimum solution involves balancing these costs and understanding the tradeoff involved in different scenarios. This paper develops analytical models based on Petri nets to provide data for analysis. The evaluations were done using the TimeNET version 3.0 software [11]. TimeNET was chosen based on its capability to allow general distribution for time events to be specified for state transitions.

The rest of the paper is organized as follows. Section 2 describes the system model and assumptions. Section 3 develops Petri net models to describe the behavior of the system

[1] A cell refers to the area governed by a base station in the PCS system. We will use the term "cell" interchangeably with the term "base station". The original scheme proposed by Sistla et al. applies to the mobile computer level but can be applied to the cell/base station level as well.

and individual cells operating under data replication with periodic maintenance. Section 4 presents analysis results, identifies optimal intervals over which to perform periodic checking to minimize the maintenance cost as a function of system parameters, and provides physical interpretations of the results. Finally, Section 5 concludes the paper and outlines future research areas.

2. System Model and Assumption

This system model considers a wireless environment composed of a primary cell, neighboring cells, and a local cell. The primary cell keeps the authoritative copy of an object. The primary cell periodically checks the status of the network to determine if service could be more efficiently provided by having data replicas of the object held at various cells throughout the system. The local cell is the cell of interest. When a user in the local cell reads a data object that is not currently in the local cell, it needs to read from the nearest neighbor that possesses a copy. This action incurs a cost that can be minimized by having lots of replicas throughout the system. However, whenever a user writes to a data object, those changes are recorded by the primary copy and then propagated to all of the existing replicas. The more replicas that exist, the more cells that need to be updated by the authoritative copy in the primary cell. Thus, the greater the number of cells that need to be updated, the higher the cost of write operations.

This paper investigates the factors involved in managing a replicated data object. This object could be an item such as a database table. Users need to be able to read this object as well as write to this object. Additionally, this paper considers the interaction between the local cell and the remainder of the network. For the network to operate efficiently, the local cell may or may not need to possess a replica of the desired data. The factors that influence this determination are listed in Table 1.

The user arrival rate (λ) is the rate at which a user moves from the network at large to the local cell. The user departure rate (μ) is the rate at which a user moves from the local cell back into the network. Both the arrival time and the departure time are assumed to be exponentially distributed to simplify the analysis. The user disconnection rate (σ_d) is the rate at which a connected user disconnects from the network. The user reconnection rate (σ_r) is the rate at which a disconnected user reconnects to the network. Both parameters are also assumed to be exponentially distributed.

The write rate (δ_W) is the average rate at which a user in the network will write to the data object. The read rate (δ_R) is the average rate at which a user in the network will read the data object. It is assumed that both the write rate and the read rate for a particular data object can be determined from historical information.

In order for the system to operate efficiently, the primary cell must periodically check the status of a remote cell to determine whether or not a copy is justified at that location. This periodic checking occurs on a deterministic basis. The time interval over which this checking occurs is denoted by a fixed periodic maintenance interval T. Note that a read and a write each have an identical cost because they both rely on the nearest neighbor that possesses a replica of the data item. Though the distance between the local cell and the nearest neighbor cell is unknown, it is immaterial because both the read and write operations must travel the same distance. That is, a read operation that occurs in the local cell without a replica must go to the nearest neighbor possessing a copy. Similarly, a write operation that occurs outside the local cell which possesses a replica is propagated from the primary cell to the nearest neighbor

Table 1. Notation.

Symbol	Definition
λ	User arrival rate to a local cell
μ	User departure rate out of a local cell
δ_R	Read rate at which a user needs to read the data item in a local cell
δ_W	Write rate at which any user in the system modifies the existing data item
σ_r	Reconnection rate of a disconnected user
σ_d	Disconnection rate of a connected user
T	Time interval over which the primary cell performs periodic checks on replicas to determine if a local cell should contain a replica
C_T	Cost incurred to perform a periodic check
N	Number of users in the system

cell and then to the local cell. Hence the cost analysis is based on a normalized cost of 1 for each missing read or remote write operation. We do not consider the extra time spent by a read operation during a write operation waiting for all replicas to be updated (e.g., through locking) in the cost analysis because the extra time spent is not part of the communication overhead to the network. For systems that allow the primary cell to propagate updates to all replicas lazily, this extra waiting "time" cost would not even exist as reading temporarily inconsistent objects would be allowed.

The number of users in the system is denoted by N corresponding to the sum of the number of users outside of the local cell (denoted by a variable n_1) and the number of users at the local cell (denoted by another variable n_2). Determining when a replica is needed at the local cell is dependent upon the number of users at the local cell reading the replicated object compared with the number of users in the system writing to the replicated object. There are two basic conditions that occur:

1. A replica is maintained or created in the local cell when the number of users at the local cell (n_2) multiplied by the read rate of the data item is equal to or exceeds the number of users in the system outside the local cell (n_1) multiplied by the write rate of the data item, i.e.,

$$n_2\delta_R \geq n_1\delta_W \,. \tag{1}$$

2. A local replica is eliminated from the local cell when the number of users at the local cell multiplied by the data item read rate is less than the number of users in the system outside the local cell multiplied by their data item write rate, i.e.,

$$n_2\delta_R < n_1\delta_W \,. \tag{2}$$

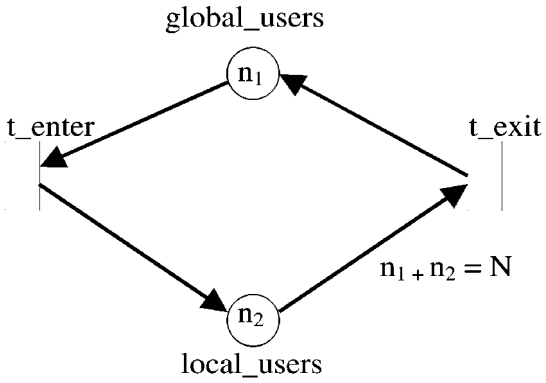

Figure 1. Enter and exit events.

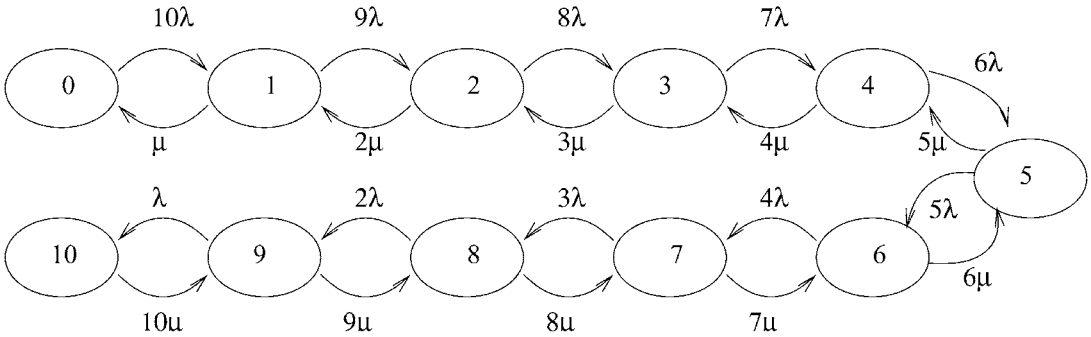

Figure 2. Corresponding Markov model for enter and exit events for $N = 10$.

3. Performance Model

To analyze the cost associated with replica management schemes in wireless mobile environments, we develop a Petri net model consisting of 2 subnets.

3.1. Enter and Exit Events

The subnet in Figure 1 models the movement of users from the network into the local cell. This local cell is the specific cell of interest and is labeled `local_users` in Figure 1. The users are initially found in a place called `global_users`. Figure 1 shows a system in which the number of users in the system is N. Each token represents a potential data consumer or user. The users move from the `global_users` place to the `local_users` place via a transition called `t_enter`. The transition rate of `t_enter` is the product of the number of users in the place called `global_users` (n_1) and the per-user arrival-rate λ. Users leave the `local_users` place via a transition called `t_exit` and return to the place called `global_users`. The transition rate of `t_exit` is the product of the number of users in the place called `local_users` (n_2) and the per-user departure rate μ.

Figure 2 shows the underlying Markov model corresponding to the SPN model in Figure 1 for modeling the user arrival/departure behavior for the case in which the number of users N is 10. We use a single-component notation n_2 to label a state, with n_2 denoting the number of users inside the local cell (and thus the number of users out of the local cell is $n_1 = 10 - n_2$ in this case). This Markov model explicitly shows the transition rates at which the system

Table 2. Steady state probability for states under different λ/μ ratios.

λ/μ	P_0	P_1	P_2	P_3	P_4	P_5	P_6	P_7	P_8	P_9	P_{10}
0.1	0.3855	0.3855	0.1735	0.0426	0.0081	0.0010	0.0001	0.0000	0.0000	0.0000	0.0000
0.5	0.0173	0.0867	0.1951	0.2601	0.2276	0.1366	0.0569	0.0163	0.0030	0.0003	0.0000
1	0.0010	0.0098	0.0439	0.1172	0.2051	0.2461	0.2051	0.1172	0.0439	0.0098	0.0010
5	0.0000	0.0000	0.0000	0.0002	0.0022	0.0130	0.0543	0.1550	0.2907	0.3230	0.1615
10	0.0000	0.0000	0.0000	0.0000	0.0001	0.0010	0.0081	0.0463	0.1735	0.3855	0.3855

goes from one state to another. For example, state 0 (no users inside the cell) goes to state 1 (one user inside the cell) with a rate of 10λ, while state 1 goes to state 2 with a rate of 9λ, etc. As a result, at equilibrium (at time infinity) there exists a steady-state probability P_i that the system can be found in a particular state i; the magnitude of P_i is dictated by the relative transition rates among states in the Markov model. Table 2 lists the steady-state probability for state i as a function of the λ/μ ratio in this Markov model. As can be seen from Table 2, when the arrival rate λ is much higher than the departure rate μ, say $\lambda/\mu = 10$, there is a high probability (0.771) that at least 9 users are inside the local cell. On the other hand, if the ratio is small, say, $\lambda/\mu = 1$, then the probability that at least 9 users are inside the cell is low (0.011). Essentially, when given a λ/μ ratio, the SPN model in Figure 1 implicitly decides the probability distribution of user population inside and out of the local cell at equilibrium as shown in Table 2.

When a user is in the local cell, i.e., in place `local_users`, it will read the object at the read rate of δ_R. If there is no local copy in the user cell when a user requests a read action, then a copy must be obtained from the nearest neighbor possessing a copy. This action incurs an extra cost of 1 (normalized) to obtain a copy from a neighboring cell. Similarly, when a user is out of the local cell, i.e., in place `global_users`, it will write to the object at the write rate of δ_W. If there is a replica maintained at the local cell, each external write operation will incur an extra cost of 1 (normalized) to propagate the update to the local cell. Note that the normalized cost for a local missing read operation is the same as that for a remote write operation because both operations involve an "extra" cost of propagating a replica (or its update) from the nearest neighbor who possesses a copy.

3.2. Periodic Maintenance Events

Figure 3 models periodic checking events for determining if there should exist a local replica of the data item at the local cell. There is one token either in the place called `object` or in the place called `no_object`. When the token is in place `object`, it represents the state in which the local cell possesses a replica of the data item.

Note that Figures 1 and 2 together define all possible system states each being characterized by the number of users inside the cell, the number of users out of the cell, and whether a replica exists in the local cell. Initially, there is not a replica in the local cell (`local_users`), so the token is present in the `no_object` place. The network is polled on a regular interval (T) to determine whether the token should be in place `object` or place `no_object`. This fixed interval is modeled by means of a deterministic transition called `tT` with a rate of $1/T$. When transition `tT` fires, a token goes to place `time_event`, representing the event that a periodic

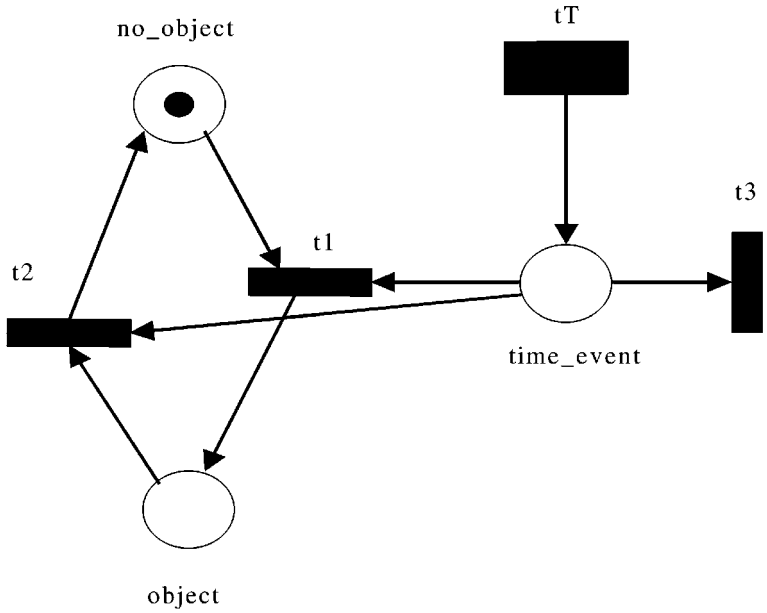

Figure 3. Periodic events that check and change object states.

maintenance check has just started. The subsequent events that will trigger depend on the current state of the system as follows:

- If there is a token in place `no_object`, then it means that a replica does not exist at the local cell. In this case, transition `t1` fires if Condition 1 is evaluated true by a guard associated with transition `t1`. Specifically, the guard associated with `t1` checks if the product of the read rate (δ_R) and the number of tokens currently in place `local_users` (n_2) is greater than or equal to the product of the write rate (δ_W) and the number of tokens currently in place `global_users` (n_1). If yes, the token at place `no_object` will move to place `object`, thus representing the case that a replica has been created at the local cell. On the other hand, if Condition 1 is evaluated false and thus `t1` is disabled, then the token in place `time_event` will vanish via transition `t3`, representing the case that the periodic check does not alter the state of the cell, i.e., the local cell still does not contain a replica. The priority for transition `t3` is set lower than that of `t1`, ensuring that `t3` fires only if `t1` fails to move the token.
- If there is a token in place `object`, i.e., a local replica exists, then when there is a token in place `time_event`, transition `t2` can fire if Condition 2 is evaluated true by a guard associated with `t2`. Specifically, the guard of `t2` enables transition `t2` if the product of the read rate and the number of tokens in the `local_users` place is less than the product of the write rate and the number of tokens in the `global_users` place, in which case the token at place `object` will move to place `no_object`. This represents the case that a replica is no longer needed at the local cell, so the local cell's copy is deallocated. However, if Condition 2 is false, then the token in place `time_event` will vanish via the immediate transition `t3`, representing the case that the periodic maintenance event does not alter the state of the cell, i.e., the local cell still contains a replica. Again the priority for transition `t3` is set lower than that of `t2` to ensure that in this case `t3` fires only if `t2` fails to move the token.

3.3. COST MODEL

There are 3 cost metrics in the unit of "cost per time unit" imposed on a local cell normalized with respect to a read/write cost of 1, accounting for the "extra" cost incurred to the network due to read, write and maintenance operations associated with the replicated object in the local cell. The first cost metric C_{read} (Equation (3)) is the average cost rate incurred because of missing reads, i.e., reading the object when there is no replica at the local cell. It is calculated by taking a weighted sum of $P_i C_{read,i}$ where P_i is the probability that the system is found in state i in the underlying semi-Markov model of the Petri net[2] and $C_{read,i}$ is the overall missing read cost per unit time for all users at the local cell when the cell is in state i. If there is a token in place `no_object` (i.e., a replica does not exist) then $C_{read,i}$ is equal to the product of the number of tokens in place `local_users` (representing the total number of users at the local cell) and the read rate per user (representing the cost per unit time per user). Otherwise, $C_{read,i}$ is 0. Specifically, the overall read cost per unit time is given by:

$$C_{read} = \sum_{i=1}^{all} P_i C_{read,i} \tag{3}$$

where

$$C_{read,i} = \begin{cases} n_2 \delta_R & \text{if place } \textbf{no\_object} \text{ contains a token in state } i \\ 0 & \text{otherwise} \end{cases}.$$

The second cost metric C_{write} (Equation (4)) is the cost rate incurred because of write propagations when a replica exists in the local cell. It is obtained by a weighted sum of $P_i C_{write,i}$ where $C_{write,i}$ is the overall write propagation cost per unit time due to writes by all users out of the cell when the system is in state i. If there is a token in place `object` (i.e., a replica exists at the local cell) then $C_{write,i}$ is equal to the product of the number of tokens in place `global_users` (representing the number of users out of the local cell) and the per-user write rate (representing the write cost per unit time per user). Otherwise $C_{write,i}$ is equal to 0. Specifically, the overall write cost per unit time is given by:

$$C_{write} = \sum_{i=1}^{all} P_i C_{write,i} \tag{4}$$

where

$$C_{write,i} = \begin{cases} n_1 \delta_W & \text{if place } \textbf{object} \text{ contains a token in state } i \\ 0 & \text{otherwise} \end{cases}.$$

The third cost metric $C_{periodic}$ (Equation (5)) is the cost rate involved in performing the periodic system check. This cost rate is calculated by multiplying the cost of each periodic check, C_T, with the checking rate. Here C_T refers to the average overhead cost associated with each periodic maintenance operation by the primary cell to communicate with the local

[2] The underlying model is referred to as a semi-Markov model because some of the time distributions in the Petri net are not exponentially distributed. The steady state probability P_i for state i in the underlying Markov model can be calculated by defining and evaluating the Petri net models shown in Figures 1 and 3 using TimeNET [11], when given a set of parameter values.

cell regarding the read/write information in order to evaluate Conditions 1 and 2 and to allocate or deallocate a replica, if necessary. The rate at which the system checks to make its decision about the placement of replica at the local copy, i.e., $1/T$, and C_T are both system parameters whose effect will be analyzed in the paper. Specifically the cost per time unit for periodic maintenance activities is:

$$C_{periodic} = \frac{C_T}{T} \tag{5}$$

where T is the periodic maintenance interval.

The overall system cost rate $C_{overall}$ imposed on a local cell (Equation (6)) is the sum of the missing-read cost rate, the write-propagation cost rate and the system periodic maintenance cost rate, i.e.,

$$C_{overall} = C_{read} + C_{write} + C_{periodic}\,. \tag{6}$$

3.4. Extension to Consider Disconnection and Reconnection Behaviors

Figure 1 assumes that all users never disconnect and reconnect so there is always a constant number of users in the system. This may not be true for mobile systems in which users may wish to disconnect and reconnect to avoid high communication cost and save battery life.

Figure 4 shows an extension to Figure 1 by considering user disconnection and reconnection behaviors. This modified SPN subnet consists of three parts, with the middle part modeling the user arrival/departure behavior of "connected" users as in Figure 1, the top part modeling the disconnection/reconnection behavior of external users (with respect to a single cell), and the bottom part modeling the disconnection/reconnection behavior of local users. Specifically, (a) two new places at the top and bottom have been created to hold the numbers of "disconnected" external and local users; (b) two transitions `t_d_enter` and `t_d_exit` have been created to model the arrival/departure behaviors of these disconnected users, with the transition rate of `t_d_enter` being the same as that of `t_enter`, i.e., λ, and the transition rate of `t_d_exit` being the same as that of `t_exit`, i.e., μ; (c) two transitions `t_g_reconnect` and `t_l_reconnect` have been created with a transition rate of σ_r to model the reconnection events of disconnected external and local users, respectively; (d) two transitions `t_g_disconnect` and `t_l_disconnect` have been created with a transition rate of σ_d to model the disconnection events of connected external and local users, respectively.

The effect is that the distribution of "connected" local and external users is now modeled by the middle part in Figure 4. With this modified model, the same analysis methodology and cost model discussed earlier can be applied (after replacing Figure 1 with this new model) to analyze the optimal checking interval for minimizing the overall cost, when given the total number of users in the system, along with the user disconnection/reconnection events characterized by the disconnection/reconnection rates (that is, σ_d and σ_r).

4. Analysis

In this Section we present numerical data based on the Petri net model developed in Section 3, namely, Figures 1 and 3, and provide physical interpretation of the results. To fairly compare the effects of different read-write ratios on the overall cost of replicated data management in mobile systems, we fix the combined read/write rate ($\delta_R + \delta_W$) to be a constant. Below we

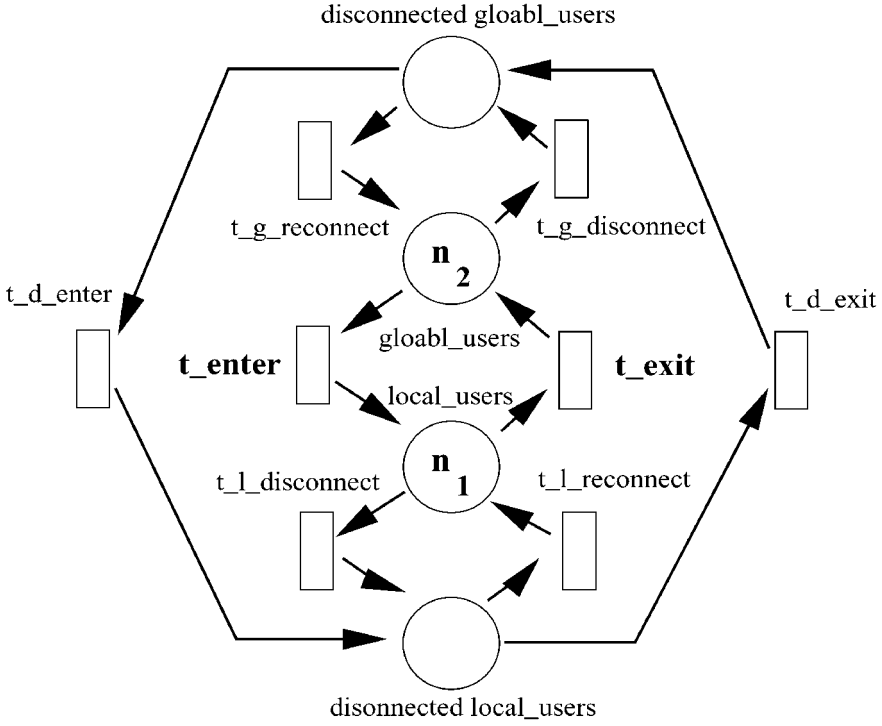

Figure 4. Enter and exit events with disconnection and reconnection behaviors.

Table 3. Working in conflict – reversing the arrival-departure and read-write rates.

	λ/μ	Read-write ratio	$C_{overall}$ with $1/T = 4$	$C_{overall}$ with $1/T = 16$
Trial #1	5:1	2:16	17.40	15.84
Trial #2	1:5	16:2	17.39	15.83

report the case in which $\delta_R + \delta_W = 18$, allowing a number of read-write rate combinations at 9-9, 12-6, 16-2, 2-16, and 6-12 to be analyzed.

4.1. Effects of the Arrival-Departure Rate and Read-Write Rate

When used in pairs, the "arrival-departure" and "read-write" rate ratios can counterbalance each other. For example, having an arrival-departure rate ratio that favors the arrivals, while simultaneously having a "read-to-write" rate ratio that favors the write action can be identically matched by reversing the ratios. Table 3 illustrates that by reversing the arrival and departure rates between Trials #1 and #2 and by reversing the read and write rates between Trials #1 and #2, the same cost value will occur. The number of users accessing the replicated object is set to 10 and $C_T = 0.1$ in these cases. Later, we will analyze the effect of changing the number of users.

For Trial #1, both the arrival rate and the write rate are high. When the arrival rate is high, there is a high probability that the local cell will contain a lot of users, thus favoring the placement of a replica at the local cell to lower the read rate cost C_{read}. When the write rate is

Table 4. Working in unison – reversing the arrival-departure and read-write rates.

	λ/μ	Read-write ratio	$C_{overall}$ when $1/T = 4$	$C_{overall}$ when $1/T = 16$
Trial #3	5:1	16:2	3.73	4.93
Trial #4	1:5	2:16	3.73	4.92

high, however, there is a greater percentage of write actions than read actions occurring in the system. All of the write actions outside the local cell would incur a cost for write propagation when a replica exists, thus increasing the write rate cost C_{write}. Thus, the effects of arrival-departure and read-write are working against each other and produce a high cost. This would occur when there are a lot of users showing up at the local cell, but the proportion of users outside the cell needing to write to the data object is very high. This same high cost is achieved in Trial #2 when the departure rate is high while the read rate is high. This would correspond to the situation where the users do not stay in the local cell very long but their need to read while in the cell is very high.

Table 4 exhibits a similar trend as that shown in Table 3, i.e., when we reverse the arrival and departure rates in Trials #3 and #4 and reverse the read and the write rates in Trials #3 and #4, the same overall cost rate will occur. However, Table 4 is different from Table 3 in that the arrival-departure ratio and the read-write ratio work in harmony.

For Trial #3, the arrival rate is high and the read rate is also high, thus favoring the placement of a local replica at the local cell. Consequently in this case the effects of arrival-departure and read-write are working together and thus produce a low cost. This would occur when there are a lot of users in the local cell and the proportion of users in the cell needing to read the data object is very high. This same low cost is achieved in Trial #4 in which the departure rate is high and the write rate is high, thus favoring the removal of the replica in the local cell. This would correspond to the situation where the users do not stay in the local cell very long and the need to write to the data object while outside the local cell is very high.

Lastly we observe that the cost values in Table 3 are much higher than the cost values in Table 4. Thus when the arrival-departure ratio and the read-write ratio work against each other, the overall cost will be higher than if the two effects work in harmony.

4.2. Optimal Periodic Maintenance Interval

The system performs a check in every fixed T period to determine if a replica should be allocated or deallocated in the local cell. The Petri net performance model developed in Section 3 can be used to determine the optimal maintenance period T. Figure 5 shows the overall cost rates under several different scenarios when the number of users accessing the replicated object is 10 and $C_T = 0.1$. The lowest point on each curve line is labeled. The highest cost occurs at the 1:5 arrival-departure rate along with the read-write rate of 16:2. This is so because these two sets of parameters conflict, representing a high overall cost scenario. The lowest periodic maintenance rate exists at $1/T = 12$. The lowest cost scenario in Figure 5 is achieved with the 7:1 arrival-departure rate and a read-write ratio of 16:2. In this case, the two sets of parameters are working in unison, representing a low overall cost rate scenario. The lowest periodic maintenance rate exists at $1/T = 0.001$. Thus, in practice no periodic

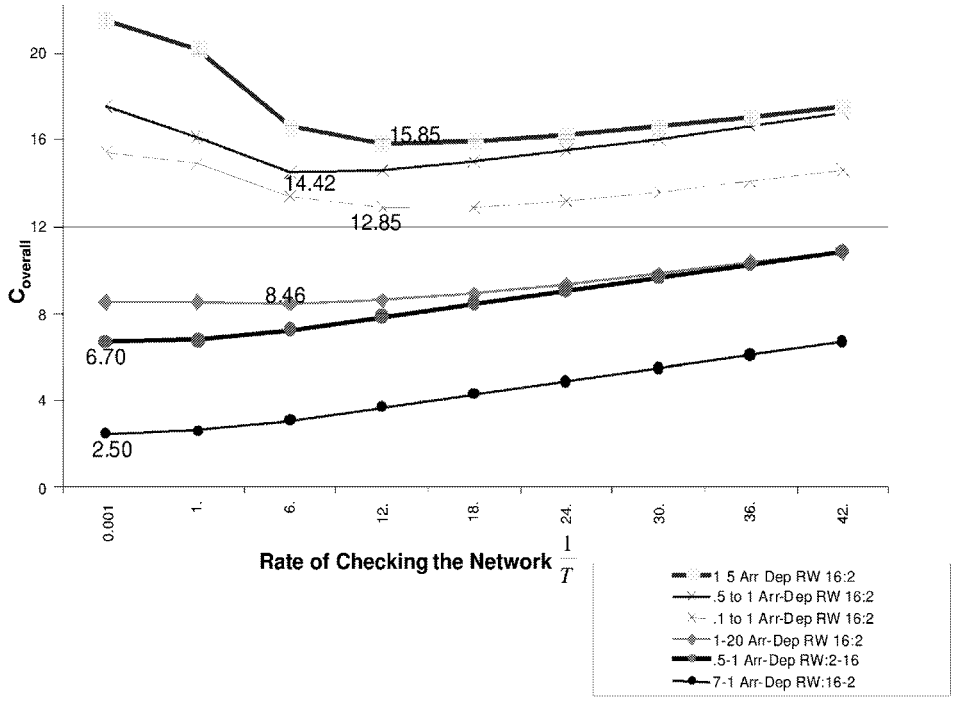

Figure 5. Optimal checking intervals for cost minimization.

maintenance of the system is required. This is because the high arrival rate and high read rate dictate a replica of the data object be allocated in the local cell virtually all the time.

The general trend for the optimal periodic maintenance rate is that the higher the overall cost rate, the higher the required periodic maintenance rate to obtain the minimum cost point on the curve. This makes sense because systems with lots of conflicts would produce high costs. To find the minimum point on their curves will require a higher rate of checking than scenarios that are stable.

4.3. Effects of Changing the Periodic Maintenance Event Cost

Figure 6 illustrates the impact of C_T on the overall system cost rate. Recall that C_T represents the average cost per periodic maintenance event initiated by the primary cell to communicate with the local cell regarding the evaluation of Conditions 1 and 2 and the replica allocation/deallocation if necessary. The scenario case is set at a 1:5 arrival-departure rate, 16:2 read-write rate, and 10 total users to illustrate the effect. There are three possible C_T values at 0.1, 0.3, and 0.5, normalized with respect to a read/write cost of 1. At very low rates of checking, the maintenance costs are the same. As the rate of checking increases, the differences in the maintenance cost become visible. The lowest system cost occurs when C_T has the lowest value of 0.1. As C_T increases to 0.3 and then to 0.5, the resulting cost curves indicate that the overall cost will increase as well. This is because a high cost associated with periodic checking increases the $C_{periodic}$ term in the overall cost equation. For all C_T values, we again observe that there always exists an optimal periodic maintenance rate at which the overall cost rate is minimized. Further, the optimal periodic maintenance rate $1/T$ will shift toward a small value

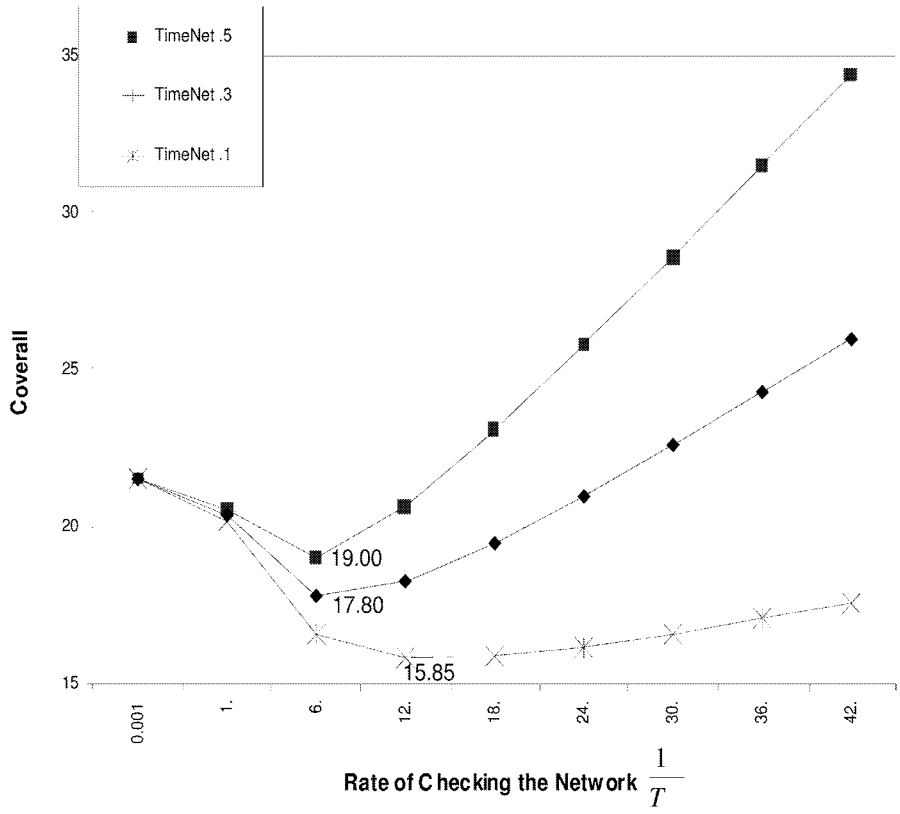

Figure 6. Effect in changing the periodic maintenance event cost.

(or toward a large periodic maintenance period T) as the cost per check increases in order to reduce the overhead associated with $C_{periodic}$.

4.4. Effects of Changing the Number of Users

The effects of increasing the number of users in the system while keeping all of the other variables constant is shown in Figure 7 where the arrival-departure ratio was held at 1:5, the read-write ratio was held at 16:2 and C_T is held at 0.1, representing a high overall cost case. Figure 7 shows that increasing the number of users in the system increases the overall cost. This result can be interpreted by considering the following two cases. The first case is when the local cell contains a replica. In this case, as the number of users in the overall system increases, the probability that a user will exist in a cell outside the local cell also increases. As the number of users outside the local cell increases, there will be a higher probability that some of those users will need to write to the same data object that is contained in the local cell. Thus, an increase in the number of write operations will cause a higher update propagation cost C_{write}. A similar argument applies to the second case in which a replica does not exist. In this case, an increase in the number of users will increase the missing read cost C_{read}. Lastly, we observe that the optimal periodic maintenance rate increases as we have more users in the system because as the system has more users with read/write actions, the system has to perform more frequent periodic maintenance activities to determine if a replica should be

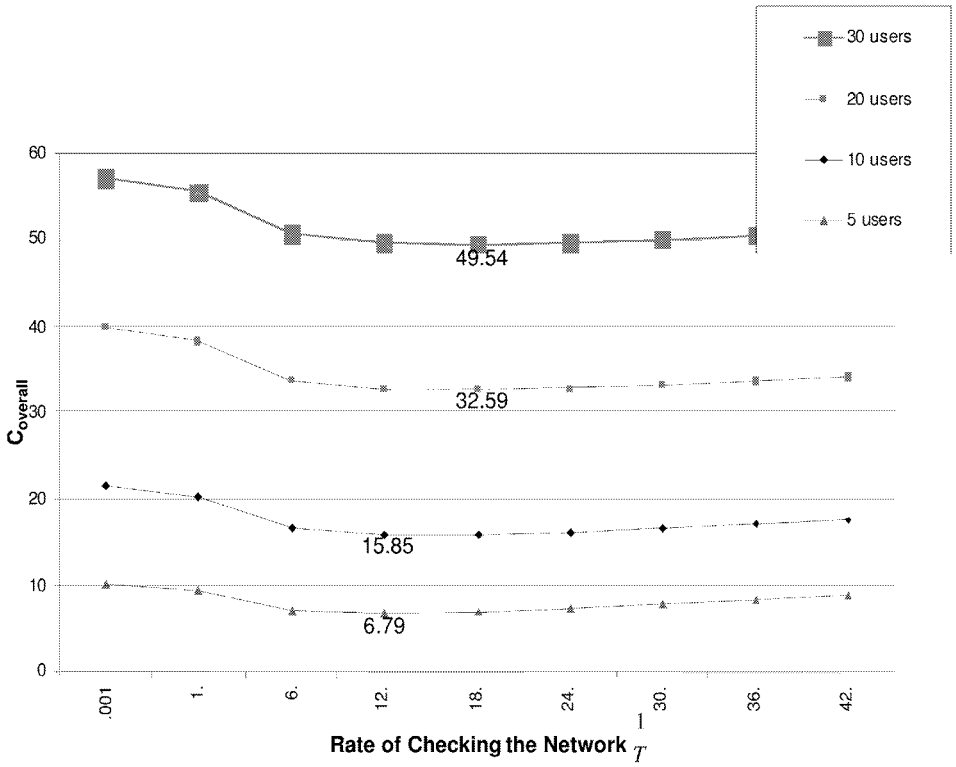

Figure 7. Effect in changing the number of users.

allocated/deallocated to reduce C_{read} and C_{write} so as to minimize $C_{overall}$, albeit at the expense of increasing $C_{periodic}$.

4.5. Sensitivity of Time Distributions

To validate the TimeNET model, the SPN model as defined by Figures 1 and 3 was additionally constructed based on SPNP version 4 [3] to test the sensitivity of the time distribution. The SPNP and TimeNET models are essentially the same except that the periodic maintenance time in the SPNP model is exponentially distributed with the average time of T. Figure 8 illustrates the close similarity between data obtained from the SPNP model data and the data obtained from the TimeNET model. The parameters used for the two higher result sets were the same: 1:5 arrival-departure rate, 16:2 read-write rate, 10 total users, and $C_T = 0.1$. The parameters used for the two lower result sets were also the same: 1:20 arrival-departure rate, 16:2 read-write rate, 10 total users, and $C_T = 0.1$.

We chose TimeNET over SPNP[3] because TimeNET has the ability to provide deterministic transitions, where our version of SPNP does not. In both cases, the TimeNET graph lines are slightly lower in overall cost. This is because the deterministic characteristics of the timer are more uniform than the exponential characteristics of the SPNP. The greater uniformity leads to more stability, less conflicts between the arrival-departure set and the read-write set and

[3] SPNP version 6 also supports the specification of non-exponential transition times. We used SPNP version 4 in this paper to validate the data obtained from TimeNET and to see the sensitivity of the result with respect to probability distribution of the periodic maintenance time.

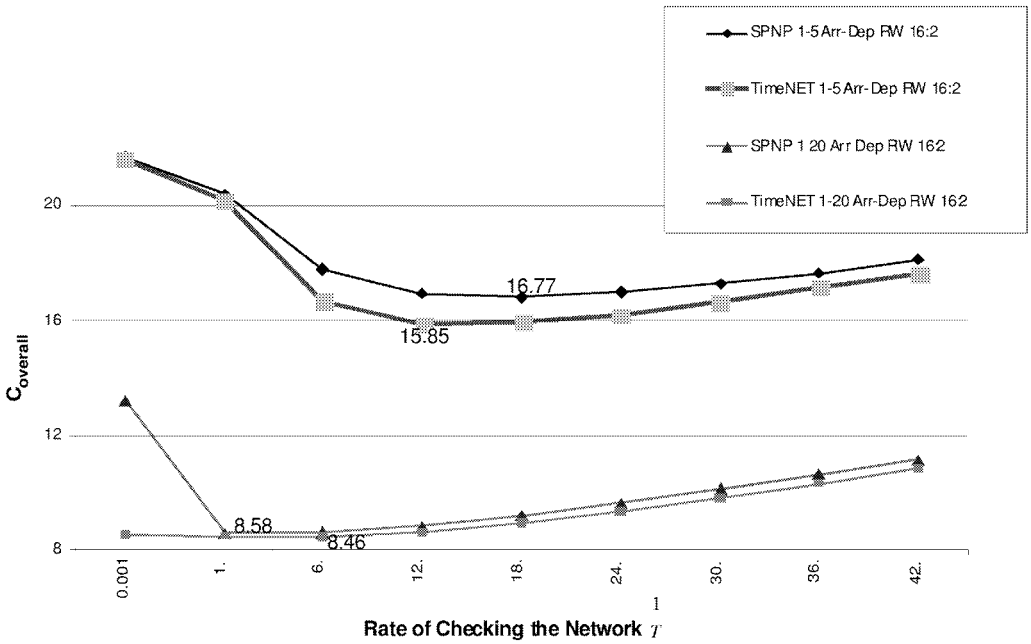

Figure 8. Sensitivity of the result toward time distributions.

thus produces a lower overall cost. A large deviation between the SPNP costs and TimeNET costs for the 1:20 arrival-departure curves occurs where the rate of checking the network is lower than 1. This deviation is caused by the variance in the periodic maintenance time being modeled as an exponentially distributed random variable with the average time T in SPNP but as a fixed constant T in TimeNet. The effect of the variance is greatly magnified at a low periodic maintenance rate.

5. Conclusions

Understanding the system environment is essential for obtaining low system costs while ensuring maximum data access speeds. The system can optimize system performance by having the primary cell check the local cells only at optimal periodic maintenance intervals so as to minimize the overall cost of managing replicated objects.

The analysis given in the paper reveals that under periodic maintenance, higher overall costs are found when the arrival-departure ratio conflicts with the read-write ratio for accessing a replicated object. For instance, a low arrival rate at the local cell with a high read rate creates a conflict, resulting in a high overall cost for maintaining the replicated object. Conversely, lower costs are found when the arrival-departure ratio works in harmony with the read-write ratio. For all cases analyzed, the analysis results show that there always exists an optimal periodic maintenance interval that minimizes the overall cost for maintaining the replicated object. Further, the higher the overall cost, the higher the periodic maintenance rate will need to be set to achieve the minimal cost.

A future research direction related to this work is to build a mobility prediction model that predicts movements of users based on the past and recent movement histories to compute the probability $\text{Prob}_i(j, t)$, i.e., the probability that a user in cell i will leave the current cell into

cell j within a period of t. This prediction model can provide an estimate of the arrival and departure rates of distinct mobile users in the system. We plan to adopt a two-level hierarchical modeling approach with the low-level model supplying predicted mobility and object access information of distinct users via simulation or statistical analysis means to a high-level model that uses the modeling approach discussed in the paper so as minimize the cost of managing replicated objects in mobile wireless systems.

References

1. D. Agrawal and A. Abbadi, "Using Reconfiguration for Efficient Management of Replicated Data", *IEEE Transactions on Knowledge and Data Engineering*, Vol. 8, No. 5, pp. 786–801, 1996.
2. D. Barbara and H. Garcia-Molina, "Replicated Data Management in Mobile Environments: Anything New under the Sun?" *IFIP Working Conf. on Applications in Parallel and Distributed Computing*, Caracas, Venezuela, April 1994, pp. 237–246.
3. G. Ciardo, R. Fricks, J. Muppala and K. Trivedi, *SPNP Reference Guide*, Version 4.0, Department of Electrical Engineering, Duke University, Durham, 1995.
4. G. Krishnamurthi, S. Chessa and A.K. Somain "Optimal Replication of Location Information in Mobile Networks", *IEEE Conference on Communications (ICC '99)*, Vancouver, June 6–10, 1999, pp. 1768–1772.
5. N. Shivakumar, J. Jannink and J. Widom "Per-user Profile Replication in Mobile Environment: Algorithms, Analysis, and Simulation Results", *ACM/Kluwer Journal on Mobile Networks and Applications*, Vol. 2, No. 2, pp. 129–140, 1997.
6. A.P. Sistla, O. Wolfson and Y. Huang, "Minimization of Communication Cost through Caching in Mobile Environments", *IEEE Trans. on Parallel and Distributed Systems*, Vol. 9, No. 4, pp. 378–390, 1998.
7. M. Wiesmann, F. Pedone, A. Schiper, B. Kemme and G. Alonso, "Understanding Replication in Databases and Distributed Systems", *20th International Conference on Distributed Computing Systems*, Taipei, Taiwan, April 2000, pp. 264–274.
8. O. Wolfson and S. Jajodia, "An Algorithm for Dynamic Data Replication in Distributed Systems", *Information Processing Letters*, Vol. 53, pp. 113–119, 1995.
9. O. Wolfson, S. Jajodia and Y. Huang, "An Adaptive Data Replication Algorithm", *ACM Transactions on Database Systems*, Vol. 22, No. 2, pp. 255–314, 1997.
10. S.Y. Wu and Y.T. Chang "An Active Replication Scheme for Mobile Data Management", *6th International Conference on Database System for Advanced Applications*, Hsinchu, Taiwan, April 1999, pp. 143–150.
11. A. Zimmerman, *TimeNET 3.0: User Manual – A Software Tool for the Performability Evaluation with Stochastic Petri Nets*, Performance Evaluation Group, TU Berlin, June 2001.

Ding-Chau Wang received the B.S. degree from Tung-Hai University, Taichung, Taiwan, and M.S. and Ph.D. degrees in computer science from National Cheng Kung University, Tainan, Taiwan. He is currently an Assistant Professor in the Department of Information Management at Southern Taiwan University of Technology. His research interests include distributed database systems, replicated systems, mobile computing and performance modeling and analysis.

Ing-Ray Chen received the B.S. degree from the National Taiwan University, Taipei, Taiwan, and the M.S. and Ph.D. degrees in computer science from the University of Houston, University Park, Houston, TX, U.S.A. He is currently an Associate Professor in the Department of Computer Science at Virginia Tech. His research interests include mobile computing, pervasive computing, multimedia, distributed systems, real-time intelligent systems, and reliability and performance analysis. Dr. Chen has served on the program committee of numerous conferences, including being as program chair of 14th IEEE International Conference on Tools with Artificial Intelligence in 2002, and 3rd IEEE Symposium on Application-Specific Systems and Software Engineering Technology in 2000. Dr. Chen currently serves as an Associate Editor for *IEEE Transactions on Knowledge and Data Engineering*, *The Computer Journal*, and *International Journal on Artificial Intelligence Tools.* He is a member of the IEEE/CS and ACM.

Chih-Ping Chu received his Ph.D. in computer science from Louisiana State University. He is currently a Professor of computer science and information engineering at the National Cheng Kung University, Taiwan. His research interests include parallel and distributed computing, object-oriented software development, and Internet applications.

I-Ling Yen received her B.S. degree from Tsing-Hua University, Taiwan, and her M.S. and Ph.D. degrees in Computer Science from the University of Houston. She is currently an Associate Professor of Computer Science at the University of Texas at Dallas. Dr. Yen's research interests are in distributed systems, fault-tolerant computing, self-stabilization algorithms, and security. Dr. Yen is a member of the IEEE/CS.