Recoverable Mobile Environments: Design and Trade-off Analysis*

Dhiraj K. Pradhan *P. Krishna* *Nitin H. Vaidya*
Department of Computer Science
Texas A&M University
College Station, TX 77843-3112
Phone: (409) 862-2438
Fax: (409) 862-2758
E-mail: {pradhan,pkrishna,vaidya}@cs.tamu.edu

Technical Report # 95-053

Abstract

The mobile wireless environment poses challenging fault-tolerant data management problems due to mobility of users, and limited bandwidth on the wireless link. Traditional fault-tolerance schemes, therefore, cannot be directly applied to these systems. Mobile systems are often subject to environmental conditions which can cause loss of communications or data. Because of the consumer orientation of most mobile systems, run-time faults must be corrected with minimal (if any) intervention from the user. The fault-tolerance capability must therefore be self-contained.

Schemes for recovery upon a failure of a mobile host are presented here. This report portrays the limitations of the mobile wireless environment, and its impact on recovery protocols. To this effect, extensions to existing traditional recovery schemes are presented which suit the mobile environment. The performance of these schemes has been analyzed to determine those environments where a particular recovery scheme is best suited. The performance of the recovery schemes mainly depends on (i) the cost of wireless transmission, (ii) the communication-mobility ratio of the user, and (iii) the failure rate of the mobile host.

*This research is supported in part by National Science Foundation grant MIP-9502563.

1 Introduction

A distributed system with mobile hosts is composed of a static backbone network and a wireless network [4]. A host that can move while retaining its network connection is a *mobile host*. The static network is comprised of the fixed hosts and communication links between them. Some of the fixed hosts, called *base stations*, are augmented with a wireless interface, providing a gateway for communication between the wireless network and the static network. Because of the limited range of the wireless transceivers, a mobile host can communicate with a *base station* only within a limited geographical region around it, referred to as a base station's *cell*. A mobile host can reside in the *cell* of only one *base station* at any given time. Because of mobility, the mobile host, while active, may cross the boundary between two cells. Thus, the task of forwarding data between the static network and the mobile host must be transferred to the new cell's *base station*. This process, known as *handoff*, is transparent to the mobile user, thus maintaining end-to-end connectivity in the dynamically reconfigured network topology.

A mobile host may become unavailable due to (i) failure of the mobile host, (ii) disconnection of the mobile host, and (iii) wireless link failure. Disconnections [4] and the weak wireless links [4] primarily delay the system response, whereas a host failure affects the system state. Strategies are developed in this report which tolerate failure of the mobile host. When a mobile host fails, it results in a loss of its volatile state. The mobile host is assumed to be *fail-stop* [9], i.e., the *base station* is able to detect the failure of the mobile host. One way to detect the failure is to require that an active mobile host send an "I am alive" message at regular intervals of time to the *base station*.

Traditional *fault-tolerance* schemes like checkpointing and message logging [6] would have served the purpose, without any modification, if the mobile hosts restricted their movements within one *cell*. But the mobile computing environment does not require a user to maintain a fixed position in the network and it allows almost unrestricted user mobility. Because of this mobility, the users cross cells, and *handoff* occurs. Thus, the abstraction of an 'immobile stable storage' cannot always be provided in mobile environments; the location of the 'stable storage' could change as the host moves. Secondly, and importantly, the traditional schemes do not consider the disparity in the bandwidths of the static network and the wireless network.

The performance of the traditional recovery schemes primarily depended on the failure rate of the host [5, 11, 12]. However, in a mobile environment, due to mobility of the hosts and the limited bandwidth on the wireless links, parameters apart from the failure rate of the mobile host also play a key role in determining the effectiveness of a recovery scheme. We identify the tradeoff parameters for a recovery scheme in a mobile environment. We analyze the performance of the recovery schemes proposed in this report and [7]. We also determine those mobile environments where a particular recovery scheme is best suited.

Other variations of the recovery schemes presented here are also plausible. Our main aim is to present the limitations of the new mobile computing environment, and its effects on recovery protocols.

In summary, this report addresses the following:

1. How do we recover the execution without user intervention, when a mobile host fails ?

2. What are the trade-offs of the recovery schemes in a mobile environment ?

3. For a given environment, which recovery scheme is preferable ?

This report is organized as follows. Section 2 presents a review of related work. Section 3 presents the recovery strategies. For sake of better understanding we also give a brief overview of the strategies presented in [7]. Section 4 presents the performance analysis of the recovery strategies, and conclusions are presented in Section 5.

2 Related Work

In the recent years, there has been a lot of work [4] dealing with mobility management, database system issues, network protocols, disconnected operation and distributed algorithms for mobile hosts. But, there is not much work on recovery protocols in a mobile environment.

Alagar et. al. [1], present schemes to tolerate base station failures. The main idea was to replicate the information stored at a base station at several "secondary" base stations. Strategies for selecting the secondary base stations were presented. These schemes can be easily integrated with the recovery schemes presented in this report to provide a system that tolerates both base station and mobile host failures.

Rangarajan et. al. [8], present a fault-tolerant protocol for location directory maintenance in mobile networks. The protocol tolerates base station failures and host disconnections. Logical timestamps are used to distinguish between an old location information and a new location information. The protocol also tolerates the corruption of these logical timestamps.

Acharya et. al. [2], identify the problems with checkpointing mobile distributed applications. They present an algorithm for recording global checkpoints for distributed applications running on mobile hosts. In this report, however, we consider protocols to recover a mobile host independent of other hosts in the system (even if the computation is distributed).

In [7], we present preliminary discussion of recovery schemes for mobile systems. This report builds on the work in [7] by presenting additional schemes, and analyzing all schemes to determine their performance.

3 Base Station-Driven Recovery Strategies

This section presents two strategies for saving the state and three strategies for *handoff* to achieve fault-tolerance. The strategies for saving the state are similar to the traditional fault-tolerance strategies. In this report, we focus our attention on schemes that can recover a faulty mobile host independent of other mobile hosts (even when the computation is distributed). Future research will deal with schemes that will be cheaper but will require more complex recovery.

3.1 State Saving

The state saving strategies presented in this report are based on the traditional checkpointing and message logging techniques. In such strategies, the host periodically saves its state at a stable storage. Thus, upon failure of the host, execution can be restarted from the last saved checkpoint.

A stable storage is necessary for saving the process checkpoint and the logs. However, a mobile host's disk storage cannot be considered stable and is vulnerable to catastrophic failures. Moreover, the mobile host can frequently be in a disconnected mode. In such cases, the stable storage of the mobile host will not be accessible. Our algorithms use the storage available at the "*local*" base station as the stable storage. The *local* base station is the base station in charge of the cell in which the mobile host is currently residing. This implies that successive checkpoints of a mobile host may be stored at different base stations.

As stated earlier, several hosts (including the mobile hosts) might be taking part in some distributed application. Such applications require messages to be transferred between the hosts, and might also require user inputs at the mobile hosts. While the user inputs may go directly to the mobile host, the messages will first reach the *base station* in charge of the *cell* in which the mobile host currently resides. The *base station* then forwards the messages to the corresponding mobile host. Likewise, all the messages sent by a mobile host, will be first sent to its *base station*, which will forward it to the destination host (static or mobile host).

Two strategies to save the process state will be discussed here [6]: (i) *No Logging* and (ii) *Logging*. During the discussion of the state saving strategies, it is assumed that the host does not move. In the next section we will deal with host mobility.

• *No Logging* Approach (denoted as N) : The state of the process can get altered either upon receipt of a message from another host or upon an user input. The messages or inputs that modify the state are called *write*[1] events. In the *No Logging* approach, the state of the mobile host is saved at the *base station* upon every *write* event on the mobile host data.

After a failure, when the mobile host restarts, the host sends a message to the *base station*, which then transfers the latest state to the mobile host. The mobile host then loads the latest state and continues from that point. (As stated earlier, it is assumed that the host does not move). The cost of frequently sending state on the wireless link is a limiting factor for this scheme.

• *Logging* Approach (denoted as L) : This approach is similar to "pessimistic" logging [3]. In this scheme a mobile host checkpoints its state periodically. To facilitate recovery, the *write* events that take place in the interval between checkpoints are also logged. As defined earlier, the messages or inputs that modify the state of the mobile host are called *write* events. If a *write* message is received from another host, the *base station* first logs it and then forwards it to the mobile host for execution. Likewise, upon an user input (write event), the mobile host first forwards a copy of the user input to the base station for logging. After logging, the *base station* sends an acknowledgment back to the mobile host. The mobile host can process the input, while waiting for the acknowledgment, but not send a response. Only upon receipt of the acknowledgment, the mobile host sends its response.

The above procedure ensures that no messages or user inputs are lost due to a failure of the mobile host. The logging of the write events continues till a new process checkpoint is backed up at the *base station*. The *base station* then purges the log of the old write events.

After a failure, when the mobile host restarts, the host sends a message to the *base station*, which then transfers the latest backed up process checkpoint of the host and the log of write events to the mobile host. (We will later consider how recovery can be achieved when a host is mobile.) The mobile host then loads the latest backed up process checkpoint and restarts executing by replaying the write events from its logs, thus reaching the state before failure.

3.2 Handoff

The two *state saving* strategies above are essentially identical to traditional fault-tolerance schemes. But, for mobile systems, in addition to above we have to deal with *handoff* process due to mobility of users. What should happen when a mobile user moves to a new cell ?

Consider Figure 1 as an example. BS_i denotes a base station, and mh_i denotes a mobile

[1]If semantics of the message is not known, in the worst case, we might have to assume that the state gets altered upon receipt of every message or an user input.

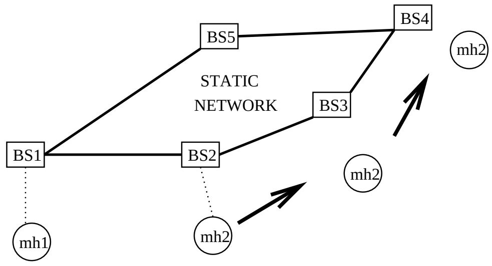

Figure 1: Handoff in the middle of an execution

host. Here, mobile hosts *mh1* and *mh2* are executing a distributed algorithm. The mobile host $mh2$ has saved its checkpoint and message log at $BS2$. In the middle of the execution, *mh2* moves to the cell of *BS3* and then to the cell of *BS4*. Handoff occurs at the boundaries of *BS2* and *BS3*, and, *BS3* and *BS4*. A failure of the mobile host *mh2* occurs on reaching the cell of *BS4*. If *mh2* had remained in the cell of *BS2* the system would have recovered because the process checkpoint and the logs are saved at *BS2*. But since no state saving took place at *BS3* or *BS4*, and since *BS4* does not know where the last checkpoint of *mh2* is stored, the recovery procedure will now have to determine the base station where the process checkpoint is saved. Thus, an additional overhead is incurred and the execution is delayed. We tackle this problem by transferring during the handoff process some *information* regarding state of the mobile host. We propose three ways to transfer this information during the *handoff* process: (i) *Pessimistic*, (ii) *Lazy*, and (iii) *Trickle*.

3.2.1 Pessimistic Strategy (P)

When a mobile host moves from one cell to another, the process checkpoint is transferred to the new cell's *base station* during handoff. If *logging* strategy is being used, then in addition to the checkpoint, the message log is also transferred to the new cell's base station. Upon receipt of the process checkpoint and/or the log, the new cell's *base station* sends an acknowledgment to the old *base station*. The old *base station*, on receiving the acknowledgment, deletes its copy of the process checkpoint and the log, since the mobile host is no longer in its cell.

The disadvantage with this approach is that there will be heavy volume of data transfer during each handoff. This can potentially cause long disruptions during handoffs. This can be avoided if we use the *Lazy* or *Trickle* strategy, as explained in the next section.

3.2.2 Lazy Strategy (L)

With *Lazy* strategy, during handoff, there is no transfer of process checkpoint and the log. Instead, the *Lazy* strategy creates a *linked list* of base stations of the cells visited by the mobile host. The mobile host may be using either one of the state saving strategies (*no logging* or *logging*) described earlier. If the mobile host is using the *No Logging* strategy, the process state is saved at the current cell's base station after every *write event*. On the other hand, if *Logging* strategy is used, a log of *write* events are maintained in addition to the last checkpoint of the mobile host at the base station. Upon a handoff, the new cell's *base station* just remembers the mobile host's last cell. Thus, as a mobile host moves from cell to cell, the corresponding *base stations* effectively form a linked list. One such linked list needs to be maintained for each mobile host.

This strategy could lead to a problem if the process checkpoint and logs of the mobile host may be unnecessarily saved at different base stations. To avoid this, upon taking a checkpoint at a base station, a notification is sent to the last cell's base station to purge the process checkpoint and logs of the mobile host, if present. If a checkpoint is not present, this base station forwards the notification to predecessor base station in the linked list. This process continues, till a base station with a old process checkpoint of the mobile host is encountered. All the base stations receiving the notification purge any state associated with the particular mobile host.

The *Lazy* strategy saves a lot of network overhead during handoff as compared to the *Pessimistic* strategy. But the recovery is more complicated. Upon a failure, if the *base station* does not have the process state, it gets the logs and the process checkpoint from the *base stations* in the linked list. The base station then transfers the process checkpoint and the log of write events to the mobile host. The host then loads the process checkpoint and replays the messages from the logs to reach the state just before failure.

3.3 Trickle Strategy (T)

In the *lazy* strategy, the scattering of logs in different base stations increases as the mobility of the host increases, potentially making recovery time-consuming. Moreover, a failure at any one base station containing the log renders the whole state information useless.

To avoid this, we propose a *trickle* strategy. In this strategy, steps are taken to ensure that the logs and the checkpoint are always at a nearby base station (which may not be the current base station). In addition, care is taken so that the handoff time is as low as the *lazy* strategy.

We make sure that the logs and the checkpoint corresponding to the mobile host are at the "predecessor base station" of the current base station[2]. The predecessor base station is the base station of the previous cell visited by the mobile host. Thus, assuming that neighboring base stations are one hop from each other (on the static network), the checkpoint and the logs are

[2]Variations of this scheme are possible where the checkpoint and logs are at *bounded* distance from current cell.

always at most one hop from the current base station.

To achieve the above, during handoff, a control message is sent to the predecessor base station to transfer any checkpoint or logs corresponding to the mobile host. Similar, to *lazy* strategy, the current base station also sends a control message to the new cell's base station indicating the last cell location of the mobile host i.e. the current cell. Thus, the new cell's base station just remembers the mobile host's last cell.

If a checkpoint is taken at the current base station, it sends a notification to the base station that has the last checkpoint and logs to purge the process state of the mobile host. During recovery, if the current base station does not have a checkpoint of the process, it gets the checkpoint and/or the logs[3] from the predecessor base station. The base station then transfers the checkpoint and/or the log to the mobile host. The host then loads the checkpoint and replays the messages from the logs to reach the state just before failure.

4 Performance Analysis

Let *wireless network factor* (denoted by α) be defined as the ratio of the cost of transferring a message over one hop of a wireless network to the cost of transferring the message over one hop of a wired network. The cost is in terms of the network bandwidth usage. Higher the value of α, costlier is the wireless transmission relative to wired transmission. Typically, the user mobility varies inversely with the available wireless network bandwidth. In Figure 2 we present a classification of wireless networks based on the user mobility and available wireless network bandwidth.

Type of Network	Bandwidth	Wireless Network Factor (α)	Typical Mobility
In-Building	> 1Mbps	Low	Pedestrian
Campus Area (Packet Relay Network)	> 64 Kbps	Moderate	Pedestrian
Wide Area	10-30 Kbps	High	Vehicular

Figure 2: A Classification of Wireless Networks

[3]If *No Logging* strategy was used for state saving, the checkpoint will be transferred. On the other hand, if *Logging* is used, the checkpoint and the log are transferred.

4.1 Terms and Notations

We use the following terminology. Significance of some of the terms will be clearer later.

• The term *operation* may refer to one of (i) checkpointing, (ii) logging, (iii) handoff, or (iv) recovery.

• **Cost** of an operation : It quantifies the network usage of the messages due to the operation.

• λ = Failure rate of the mobile host. We assume that the time interval between two failures follows an exponential distribution with a mean of $1/\lambda$.

• μ = Handoff rate of the host. We assume that the time interval between two handoffs follows an exponential distribution with a mean of $T = 1/\mu$ units of time.

• The time interval between two consecutive write events is assumed to be fixed and equal to $1/\beta$. Write events comprise of user inputs and messages from other hosts. Since we are interested only in the performance difference of the different proposed schemes, this assumption will not affect the results significantly.

• r = Communication-mobility ratio, defined as the expected number of write events per handoff. It is equal to β/μ. For a fixed β, a small value of r implies high mobility and vice-versa.

• ρ = Fraction of write events that are user inputs. If ρ is 1, then all the write events are user inputs. This means that the application is not distributed in nature, and that the mobile host is the only participant in this execution.

• T_c = Checkpoint interval, defined as the time spent between two consecutive checkpoints executing the application. T_c is fixed for all schemes under consideration. Specifically, T_c is $1/\beta$ for *no logging* schemes.

• k = Number of write events per checkpoint. For the *logging* schemes, $k = \beta T_c$. For the *no logging* schemes, k is always equal to 1.

• α = Wireless network factor. This is the ratio of the cost of transferring a message over one hop of a wireless network to the cost of transferring the message over one hop of a wired network.

• $N_c(t)$ = Number of checkpoints in t time units.

• $N_l(t)$ = Number of messages logged in t time units.

• C_c = Average cost of transferring a checkpoint state over one hop of the wired network.

• C_l = Average cost of transferring an application message over one hop of the wired network.

• $\gamma = C_l/C_c$ = Relative logging cost. It is the ratio of the cost of transferring an application message to the cost of transferring a checkpoint state over one hop of the wired network.

• C_m = Average cost of transferring a control message over one hop of the wired network. The size of a control message is assumed to be typically much less than the size of an application message.

• $\epsilon = C_m/C_c$ = Relative control message cost. It is the ratio of the cost of transferring a control message to the cost of transferring a checkpoint state over one hop of the wired network.

- C_h = Average cost of a handoff operation. (to be evaluated later)
- C_r = Average cost of a recovery operation. (to be evaluated later)
- C_t = Average total cost per handoff. It will be explained.

4.2 Modeling and Metrics

For convenience we name the interval between two handoffs as *handoff interval*. A handoff interval can be represented using a 3-state discrete Markov chain [10, 11] as presented in Figure 3.

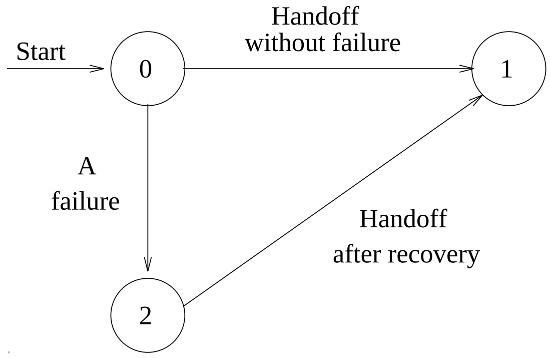

Figure 3: Markov Chain Representation

State 0 is the initial state when the handoff interval starts. During the handoff interval, the host receives messages and/or user inputs (write events). Depending on the state saving scheme the host either takes a checkpoint or logs the write events. A transition from state 0 to state 1 occurs if the handoff interval is completed without a failure. If a failure occurs during the handoff interval, a transition is made from state 0 to state 2. After state 2 is entered, a transition occurs to state 1 once the handoff interval is completed. To simplify the analysis, we have assumed that at most one failure occurs during a handoff interval. This assumption does not affect the results significantly when the average handoff interval is small compared to the mean time to failure. Our assumption is analogous to that made in [12] for evaluating rollback recovery.

Figure 4 illustrates an example of the state transitions in a handoff interval. Figure 4(a) is a case when the handoff interval is greater than a checkpoint interval. Thus, multiple checkpoint operations take place within a handoff interval. As seen in Figure 4(a), the initial state is state 0. A transition to state 2 takes place after a failure. State 2 begins with a recovery operation. A subsequent transition to state 1 takes place after a handoff operation. Figure 4(b) is a case when the checkpoint interval is greater than a handoff interval. Thus, multiple handoff operations take place within a checkpoint interval. State transitions are similar to the previous case.

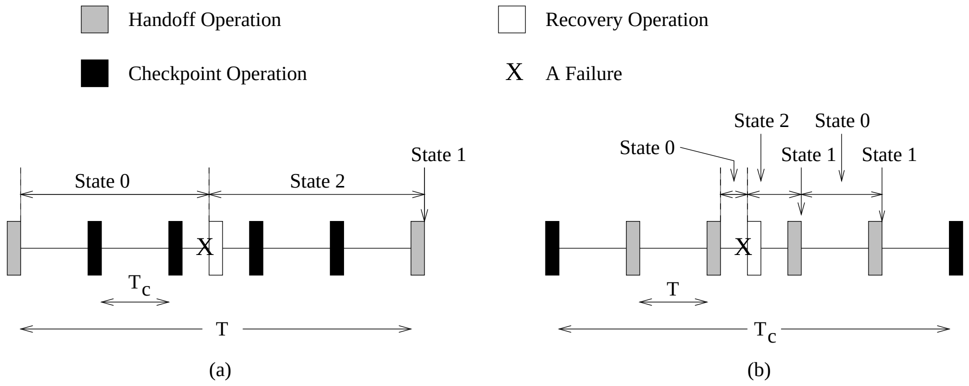

Figure 4: State Intervals

Each transition (a,b), from state a to state b in the Markov chain, has an associated transition probability P_{ab} and a cost C_{ab}. Cost C_{ab} of a transition (a,b) is the expected total cost of operations that take place during the time spent in state a before making the transition to state b.

The transition probability P_{02} is the probability that a failure occurs within a handoff interval. Let t_f be the time of failure, and t_h be the time of handoff, then,

$$P_{02} = P(t_f < t_h) = \int_0^\infty \int_{\tau_f}^\infty \lambda \mu e^{-\lambda \tau_f} e^{-\mu \tau_h} \, d\tau_h d\tau_f$$

Solving the above, we get,

$$P_{02} = \frac{\lambda}{\lambda + \mu}$$

The expected duration from the beginning of the checkpoint interval until the time when the failure occurred, given that a failure occurs before the end of the checkpoint interval is,

$$T_{cexp} = \int_0^{T_c} \frac{t\lambda e^{-\lambda t}}{1 - e^{-\lambda T_c}} \, dt = \frac{1}{\lambda} - \frac{T_c e^{-\lambda T_c}}{1 - e^{-\lambda T_c}}$$

As stated earlier, $N_c(t)$ and $N_l(t)$ denotes the number of checkpoints and messages logged in t time units respectively. Cost C_{01} of transition (0,1) is the expected total cost of operations that take place during the time spent in state 0 before making the transition to state 1. C_{01} is as follows : (Recall that T is the handoff interval.)

$$C_{01} = (\alpha C_c) * N_c(T) + (\alpha C_l) * N_l(T) + C_h \tag{1}$$

The performance metrics for the proposed schemes are :

• **Handoff Time** : The handoff time is the extra time required due to the transfer of state information by the fault tolerance scheme during handoff operation. It is basically the difference in the time duration of a handoff operation with fault tolerance and the time duration of a handoff operation without fault tolerance.

• **Recovery Cost** : Upon a failure, this is the expected cost incurred by the recovery scheme to recover the host to the state just before the failure.

• **Total Cost** : This is the expected cost incurred during a handoff interval with and without failure. The total cost is determined as follows :

$$C_t = C_{01} + P_{02} C_r \tag{2}$$

The costs will depend on the state saving and handoff scheme used. We denote the total cost of a scheme that employs a combination of state saving scheme X ($X \in \{N, L\}$) and a handoff scheme Y ($Y \in \{P, L, T\}$) as C_{tXY}. For example, C_{tLP} is the total cost of the scheme that uses a combination of *logging* scheme for state saving and *pessimistic* scheme for handoffs.

Now, we will derive the total cost, recovery cost and the handoff time for each scheme. In our analysis we assume that the cost of transmitting a message from one node to another depends on the number of hops between the two nodes. We also assume that neighboring base stations are at a distance of one network hop from each other.

4.3 No Logging - Pessimistic (NP) Scheme

A checkpoint operation takes place upon every write event. Thus upon every write event, the process state is transferred over the wireless network to the base station incurring a cost of αC_c on average. There are r write events during a handoff interval. Since there is no logging operation involved, $N_l(t) = 0, t \geq 0$. During a handoff the last checkpoint is transferred to the new base station, and in reply, an acknowledgement is sent. Thus, the cost of handoff $C_h = C_c + C_m$.

Replacing $N_c(T) = r$, $N_l(T) = 0$, and $C_h = C_c + C_m$ in Equation 1 we get,

$$C_{01} = (r\alpha + 1)C_c + C_m$$

During recovery, the process state will be present at the current base station. Therefore, the recovery cost is the cost of transmitting a request message from the mobile host to the base station and the cost of transmitting the state over one hop of the wireless link. Thus,

$$C_r = \alpha(C_m + C_c)$$

The total cost C_{tNP} can be determined by replacing the above costs in Equation 2.

4.4 No Logging - Lazy (NL) Scheme

The checkpoint and logging operations are similar to NP scheme in Section 4.3. However, upon the first checkpoint operation at the current base station, a control message is sent to the base station that has the last checkpoint, requesting it to purge that checkpoint. Let that base station be on average N_h hops from that current base station. (We will derive N_h below.) Thus, the average cost of purging is $N_h C_m$. A handoff operation includes setting a pointer at the current base station and transfer of a control message between the current and the new base station. Since setting a pointer does not involve any network usage, the cost of handoff C_h is equal to the cost C_m of transferring a control message between the two base stations.

Replacing $N_c(T) = r$, $N_l(T) = 0$, and $C_h = C_m$ in Equation 1 we get,

$$C_{01} = r\alpha C_c + N_h C_m + C_m$$

Since a checkpoint operation takes place upon every write event, and the checkpoint is not transferred to the new base station upon a handoff, the location of the last checkpoint will depend on the number of handoffs since the last write event. As stated earlier, the cost of transmitting a message from one node to another depends on the number of hops between the two nodes. The upper bound on the number of hops traversed to transfer the last checkpoint to the current base station will be the number of handoffs between two write events (or in this case checkpoints). In addition to this, the cost of transferring the process checkpoint over the wireless link is incurred – αC_c. The average number of handoff operations completed since the last write event (or checkpoint event) till the time of failure is N_h, where,

$$N_h = \mu T_{cexp} \tag{3}$$

A cost is also incurred due to the request message from the mobile host for the checkpoint. The cost is $(\alpha + N_h)C_m$. Thus, an upper bound on the recovery cost is

$$C_r = (N_h + \alpha)(C_c + C_m)$$

We will use this C_r to evaluate C_{tNL}. As this C_r estimated is an upper bound, C_{tNL} estimated here is somewhat pessimistic. The total cost C_{tNL} can be determined by replacing above costs in Equation 2.

4.5 No Logging - Trickle (NT) Scheme

The checkpoint and logging operations are same as for the NP and NL schemes described in Sections 4.3 and 4.4. As in the NL scheme, the handoff cost is the cost of transferring a control message from the current to the new base station. In addition to this, a control message is sent to the previous base station requesting it to transfer any state corresponding to the mobile host. This ensures that the maximum number of hops traversed to transfer the state during recovery is one. The cost of handoff operation is thus the sum of the cost of transferring state over one hop of wired network and the cost of sending two control messages. Thus, $C_h = C_c + 2C_m$. It should be noted however, that the handoff time is only determined by C_m for the transfer of a control message between the current and the new base station. The time spent due to the transfer of state is transparent to the user.

Upon the first checkpoint operation at the current base station, a control message is sent to the base station that has the last checkpoint, requesting it to purge that checkpoint. Let that base station be on average N'_h hops from the current base station. (We derive N'_h below.) Thus, the cost of purging is $N'_h C_m$. Thus,

$$C_{01} = (r\alpha + 1)C_c + 2C_m + N'_h C_m$$

As stated earlier, during recovery operation, the number of hops traversed to transfer state is at most one. Thus,

$$C_r = (N'_h + \alpha)(C_c + C_m) \text{ where,}$$

$$N'_h = 1(1 - e^{-\mu T_c}) + 0(e^{-\mu T_c}) = (1 - e^{-\mu T_c}) \tag{4}$$

In the above equation, $e^{-\mu T_c}$ is the probability that the last checkpoint took place at the current base station. The total cost C_{tNT} can be determined by replacing the above costs in Equation 2.

4.6 Logging - Pessimistic (LP) Scheme

For this scheme, the state of the process will contain a checkpoint and a log of write events. The message log will contain the write events that have been processed since the last checkpoint. The logging cost will involve only those write events that have to traverse the wireless network to be logged at the base station. Only the user inputs need to traverse the wireless network to be logged. On the other hand write events received from other hosts in the network anyway come via the base station, so they get logged first and then forwarded to the mobile host. Thus no cost is incurred due to logging of write events from other hosts. As stated earlier, ρ is the fraction of write events that are user inputs. Thus, ρr is the number of user inputs between two handoffs. This is also the number of logging operations in a handoff interval. For each logging operation there is a cost for the acknowledgment message sent by the base station over the wireless network. The cost of each acknowledgment message is αC_m.

The handoff cost will now include the cost of transferring the state as well as the message log, and the cost of transferring an acknowledgment. Let, ν denote the average log size during handoff. Then, the average handoff cost will be $(\nu C_l + C_c + C_m)$. Under the assumption of handoffs being a poisson process, $\nu = \frac{k-1}{2}$. (Recall that k is the number of write events per checkpoint.)

Replacing $N_c(T) = \frac{r}{k}$, $N_l(T) = \rho r$, and $C_h = (\nu C_l + C_c + C_m)$ in Equation 1 we get,

$$C_{01} = (\frac{r}{k})\alpha C_c + \rho r \alpha C_l + \rho r \alpha C_m + (\nu C_l + C_c + C_m)$$

During recovery, the checkpoint and the log are present at the current base station. Therefore, the recovery cost is the cost of transmitting a request message from the mobile host to the base station and the cost of transmitting the checkpoint and log over one hop of the wireless network. The expected size of the log at the time of failure is ν'. For poisson failure arrivals, $\nu' = \frac{k-1}{2}$. Therefore,

$$C_r = \alpha(\nu' C_l + C_c + C_m)$$

The total cost C_{tLP} can be determined by replacing the above costs in Equation 2.

4.7 Logging - Lazy (LL) Scheme

The checkpoint and logging operations are same as for the LP scheme described in Section 4.6. When a checkpoint takes place, the old checkpoint and logs at the different base stations are purged off. As determined earlier in Section 4.4, purging cost is $N_h C_m$, and handoff cost is C_m.

$$C_{01} = (\frac{r}{k})\alpha C_c + \rho r \alpha C_l + \rho r \alpha C_m + N_h C_m + C_m$$

As determined earlier, the expected number of write events completed till the time of failure since the last checkpoint is $\nu' = \frac{k-1}{2}$ (Section 4.6). This is distributed over different base stations. The last checkpoint and the logs have to traverse on an average N_h (Equation 3) hops on the wired network to reach the current base station, and an additional wireless hop to reach the mobile host. A cost of $(N_h + \alpha)C_m$ is also incurred due to the request message for the checkpoint and the logs (same as for NL scheme). Therefore,

$$C_r = (N_h + \alpha)(C_c + \nu' C_l + C_m)$$

The total cost C_{tLL} can be determined by replacing the above costs in Equation 2.

4.8 Logging - Trickle (LT) Scheme

The checkpoint and logging operations are same as in LP and LL. The cost of handoff operation is thus the sum of the cost of sending two control messages (same as for NT scheme) and the cost of transferring checkpoint and logs over one hop of wired network. Thus, $C_h = \nu C_l + C_c + 2C_m$. The cost of purging is $N_h' C_m$ (Section 4.5). Thus,

$$C_{01} = (\frac{r}{k})\alpha C_c + \rho r \alpha C_l + \rho r \alpha C_m + \nu C_l + C_c + 2C_m + N_h' C_m \quad \text{and,}$$

$$C_r = (N_h' + \alpha)(\nu' C_l + C_c + C_m)$$

The total cost C_{tLT} can be determined by replacing the above costs in Equation 2.

4.9 Results

We normalize the above equations with respect to C_c. Recall that γ is the relative logging cost and is equal to C_l/C_c. Thus, $C_l = \gamma C_c$. Recall that ϵ is the relative control message cost and is equal to C_m/C_c. We assume that $C_m \ll C_c$ (which is the case in practice). We replace, $C_c = 1$, $C_l = \gamma$, and $C_m = \epsilon$ in the above equations and determine the handoff time, recovery cost and the total cost. The rate of writes β is set to 1.

For our analysis, we assume that $\rho = 0.5$. (Recall that ρ is fraction of write events that are user inputs.) This means that the write events comprise of equal percentage of user inputs and messages from other hosts. For our analysis, we fix the relative control message cost $\epsilon = 10^{-4}$.

4.9.1 Determination of Optimum Checkpoint Interval

An optimum checkpoint interval is required to be determined only for the *logging* schemes. Recall that for a *no logging* scheme a checkpoint takes place upon every write event. However, for a *logging* scheme a checkpoint takes place periodically every T_c units of time. Since, the rate of writes β is equal to 1, the number of write events per checkpoint (k) is equal to T_c. A "good" value for k is to be chosen for the logging schemes. We define a good value of k to be the one that offers the minimum total cost. This value of k (say, $k_{opt_{LY}}$ for a logging scheme that uses scheme Y for handoffs – $Y \in \{P, L, T\}$) is a function of the failure rate λ, relative logging cost γ, wireless network factor α and communication-mobility ratio r. Let us consider the LL scheme as an example. The value of $k_{opt_{LL}}$ for the LL scheme is obtained as a solution of,

$$\frac{\partial C_{tLL}}{\partial k} = 0 \text{ and } \frac{\partial^2 C_{tLL}}{\partial^2 k} < 0$$

Figure 5 illustrates the variation of $k_{opt_{LL}}$ with r and α for failure rate $\lambda = 10^{-2}$ and relative logging cost $\gamma = 0.1$. It can be noticed that $k_{opt_{LL}}$ increases as r and α increase. For a given k, as r increases, the number of checkpoints per handoff increases. This increases the total cost. As α increases, the cost due to a checkpoint increases. Thus, to lower the total cost, k should also increase. Therefore, as r and/or α increases $k_{opt_{LL}}$ also increases.

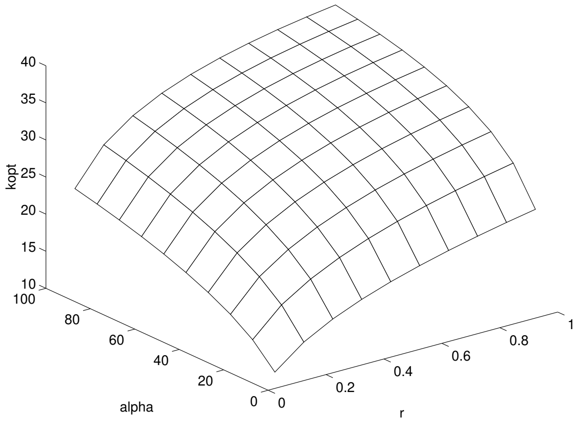

Figure 5: $k_{opt_{LL}}$ vs. r and α : $\lambda = 10^{-2}$, $\gamma = 0.1$

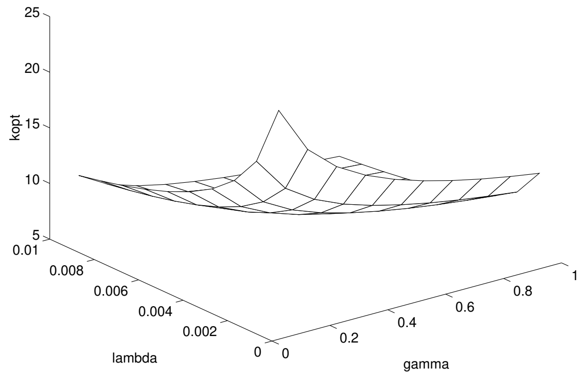

Figure 6: $k_{opt_{LL}}$ vs. γ and λ : $\alpha = 10$, $r = 0.1$

Figure 6 illustrates the variation of the $k_{opt_{LL}}$ with relative logging cost γ and failure rate λ for communication-mobility ratio $r = 0.1$ and wireless network factor $\alpha = 10$. It can be noticed that $k_{opt_{LL}}$ decreases as γ and λ increases. As γ increases, cost of logging operation increases. Thus, checkpoint interval size has to be reduced to decrease total cost. Therefore, $k_{opt_{LL}}$ decreases as γ increases. As λ increases, the probability of failure increases. Thereby, the fraction of recovery cost in the total cost increases. The recovery cost for the logging schemes depends on the average log size during failure. The average log size in turn depends on checkpoint interval size. To decrease recovery cost, we need to reduce checkpoint interval size. Therefore, as λ increases, $k_{opt_{LL}}$ decreases.

Similar behavior was observed for the LP and LT schemes. We used $k = k_{opt_{LY}}$ for the analysis of the logging scheme which uses scheme Y for handoffs, where $Y \in \{L, P, T\}$. We assume that relative logging cost $\gamma = 0.1$. We vary α to represent different classes of wireless networks. We vary λ to represent different failure rates. We vary the value of r to represent different user mobility patterns. We will now illustrate the performance of each of the proposed schemes.

4.9.2 Handoff Time

Recall that the handoff time is the extra time required due to the transfer of state information by the fault tolerance scheme during handoff operation. Let BW be the bandwidth of a link on the wired network. Table 1 illustrates the handoff cost and (handoff time $\times BW$) of the various schemes. The pessimistic handoff schemes incur a very high handoff *time* compared to the lazy and trickle handoff schemes. This is because, in the lazy scheme, there is no state transfer during handoff. And in the trickle scheme, the state transfer is performed separately from the handoff. It can be noticed however, that for a given state saving scheme, the handoff *cost* of the trickle handoff scheme is almost equal to the pessimistic handoff scheme.

Scheme	Handoff Cost	(Handoff Time $\times BW$)
NP	$1+\epsilon$	$1+\epsilon$
NL	ϵ	ϵ
NT	$1+2\epsilon$	ϵ
LP	$1+\nu\gamma+\epsilon$	$1+\nu\gamma+\epsilon$
LL	ϵ	ϵ
LT	$1+2\epsilon+\nu\gamma$	ϵ

Table 1: Handoff Cost and (Handoff Time $\times BW$)

4.9.3 Recovery Cost

In Figure 7, we plot the recovery cost for all the schemes for $\alpha = 10$, and $\lambda = 10^{-2}$. Similar behavior was observed for other values. The recovery cost of the logging schemes is more than the no logging schemes. This is a well-known fact in the traditional fault tolerance schemes. The recovery cost of the NP scheme is independent of communication-mobility ratio r. The NP scheme incurs the lowest cost for all values of communication-mobility ratio r. This is because the last checkpoint state is always present at the current base station. The recovery cost of the NT scheme is a constant for low r ($r < 1$) and slightly more than the NP scheme. This is because the last checkpoint of the host is always available one hop from the current base station. As stated earlier, β is fixed for the analysis. For a fixed β, as μ (i.e., mobility) decreases, r ($= \beta/\mu$) increases, and the probability of the last checkpoint being available at the current base station increases. Therefore, at high values of r ($r > 1$) the costs of NT and NP converge (refer Figure 7).

The recovery cost of the LP and the LT scheme is proportional to the size of the log before failure. The size of the log depends on k. Since, k ($= k_{opt_{LP}}$ or $k_{opt_{LT}}$) increases with r, the recovery cost also increases. Similar to NP and NT schemes, at low values of r ($r < 1$), the recovery cost of the LT scheme is slightly higher than LP scheme. However, at high values of r, the costs of LP and LT schemes become similar (refer Figure 7).

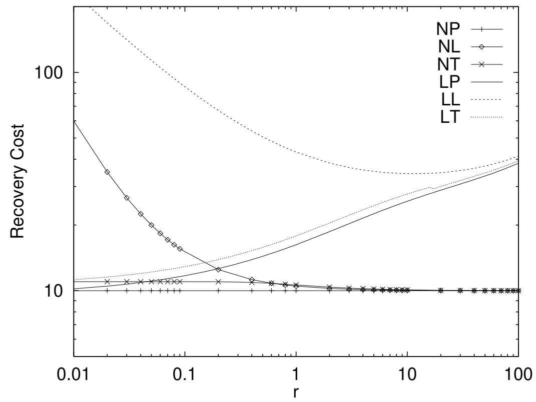

Figure 7: Recovery Cost : $\lambda = 10^{-2}$, $\alpha = 10$

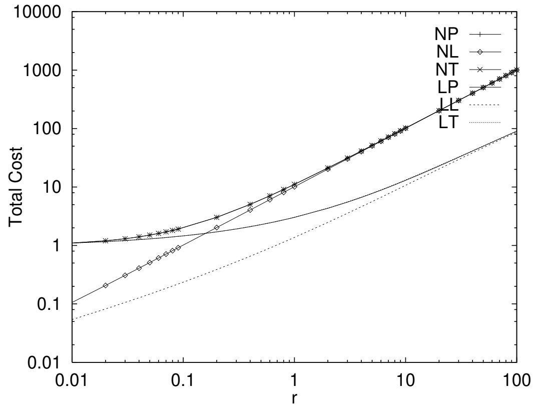

Figure 8: Total Cost : $\lambda = 10^{-2}$, $\alpha = 10$

For low values of r ($r < 1$), it can be noticed that the recovery cost of the lazy handoff (LL and NL) schemes are much larger than the pessimistic and the trickle handoff schemes. This is because the checkpoint state might not be at the current base station. Secondly, the log of write events might be distributed at different base stations. Thus, the cost of recovery will include the cost of transferring the checkpoint state and the log from the various base stations to the current base station and then forwarding them to the mobile host over the wireless link. The LL scheme incurs a very high recovery cost for low r. Lower the value of r, greater is the amount of scatter of recovery information. As r increases, the possibility of a checkpoint operation taking place at the current base station increases. Thus, the recovery cost decreases as r increases. However, as r increases, k ($= k_{opt_{LL}}$) also increases. Thus, after some value of r, the recovery cost starts increasing. On the other hand, the recovery cost of the NL scheme continues to decrease as r increases. At high values of r ($r > 1$), the cost of NL converges to NP and NT. Similarly, the cost of LL scheme become similar to LP and LT (refer Figure 7).

As expected, at high values of r (i.e., low mobility), the recovery cost becomes almost independent of the handoff scheme used – the state saving scheme determining the recovery cost.

4.9.4 Total Cost

Figure 8 illustrates the variation of the total cost of various schemes with r for $\lambda = 10^{-2}$ and $\alpha = 10$. The total cost comprises of the failure free cost and the recovery cost. The total cost of the pessimistic handoff scheme and trickle handoff scheme are almost equal ($NP \approx NT$, and,

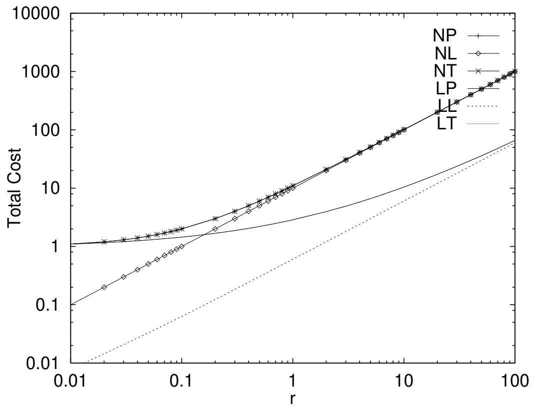

Figure 9: Total Cost : $\lambda = 10^{-5}$, $\alpha = 10$

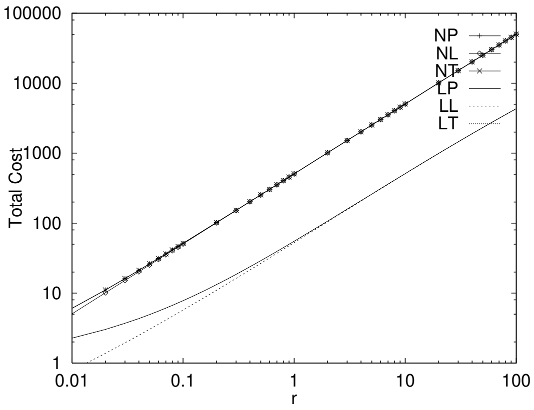

Figure 10: Total Cost : $\lambda = 10^{-2}$, $\alpha = 500$

$LP \approx LT$). The lazy handoff scheme incurs a much lower total cost at low values of r ($r < 1$). At high values of r the total cost of the different handoff schemes converge. However, the difference in the total costs of the logging and no logging schemes remain. The total cost of the no logging scheme is higher than the logging scheme for all values of r. The LL scheme incurs the lowest total cost for all r.

Figure 9 illustrates the variation of the total cost with r for $\lambda = 10^{-5}$. Comparison of Figure 8 and Figure 9 indicates that, for the same α, as λ decreases, the cost difference between the handoff schemes for the *logging* state saving scheme increases. As the probability of failure decreases, the lazy handoff scheme becomes more justified. The total costs of the trickle and the pessimistic handoff schemes are almost always equal, and both are higher than the lazy scheme.

Figure 10 illustrates the variation of the total cost with r for $\alpha = 500$. The total cost increases with α. Comparison of Figure 8 and Figure 10 indicates that, for the same λ, as α increases, the cost difference between the handoff schemes reduces. Thus, the performance of a scheme becomes more dependent on the state saving scheme used than the handoff scheme.

4.9.5 Discussion

The handoff time of the pessimistic handoff schemes are very high and unacceptable for applications that require connection oriented services. During the handoff period , there are no packets sent or received by the mobile host. Thus, if the handoff times are very high, the communication protocols used for these connection oriented services might timeout during handoff [4].

Some applications might require a very quick recovery, and some other application might require a very low total cost to be incurred by the recovery schemes. Some hosts might be running the application in a high failure rate environment, and some in a very low failure rate environment.

As can be observed from the results, there is is no single recovery scheme that performs best (lowest total cost, lowest recovery cost and lowest handoff time) for all application environments.

We will now determine the environments where a particular recovery scheme is best suited. We classify the environment into low failure rate and high failure rate environment. The summary of the results are presented in Figure 11.

In a low failure rate environment, failures occur very infrequently. The primary goal of a recovery scheme in such an environment is to incur low failure free cost. The LL scheme incurs low failure free cost for all values of r. However, for high α values, the difference in the failure free costs of the LL and LT scheme reduces. Since, the recovery time of the LT scheme is much lower than LL scheme for low values of r, it is more preferable to choose LT for high α values.

r	α	λ	Best Scheme
Low	Low	Low	*LL*
		High	*NT*
	High	All	*LT*
High	All	All	*LL*

Figure 11: Summary of Results

In a high failure rate environment, failures occur very frequently. The primary goal of a recovery scheme is to incur low failure free cost and low recovery cost. For low r values, the recovery cost of the LL and NL scheme is very high. Thus, we need to choose between NT or LT. When α is low, NT incurs a low failure free cost (slightly more than LT), and provides a quicker recovery than LT. However, when α is high, LT becomes more preferable. For high r values, LL is preferable over other schemes.

5 Conclusions

Mobile computing is rapidly becoming popular. The new mobile wireless environment presents many challenges due to the mobile nature of the hosts and the limited bandwidth on the wireless network. Presented in this report are recovery schemes for a mobile wireless environment. The recovery schemes are a combination of a state saving strategy and a handoff strategy. Two strategies for state saving, namely, (i) *No Logging* and (ii) *Logging*, and three strategies for handoff, namely, (i) *pessimistic*, (ii) *lazy*, and (iii) *trickle* are discussed.

Our main goal here is to present the limitations of the new mobile computing environment, and its effects on recovery protocols. The trade-off parameters to evaluate the recovery scheme were identified. It was determined that in addition to the failure rate of the host, the performance of a recovery scheme depended on the mobility of the hosts and the wireless bandwidth. We analyzed the performance of the various recovery schemes proposed in this report and determined those mobile environments where a particular recovery scheme is best suited.

Acknowledgements

We would like to thank Fred Meyer for his help and insightful comments.

References

[1] S. Alagar and S. Venkatesan, "Tolerating Mobile Support Station Failures," Computer Science Technical Report, Univ. of Texas at Dallas, November, 1993.

[2] A. Acharya and B. R. Badrinath, "Checkpointing Distributed Applications on Mobile Computers," *Proc. of the 3rd Intl. Conf. on Par. and Dist. Info. Sys.,* pp. 73-80, Sep. 1994.

[3] A. Borg et. al., "A Message System Supporting Fault Tolerance," *9th ACM Symposium on Operating System Principles,* pp. 90-99, Oct., 1983.

[4] G.H.Forman, et.al., "The Challenges of Mobile Computing," *IEEE Computer,* 38-47, Apr.1994.

[5] J. Long, W. K. Fuchs and J. A. Abraham, "Implementing Forward Recovery using Checkpoints in Distributed Systems," *Proc. of Dependable Computing for Critical Applications,* Feb., 1991.

[6] Pankaj Jalote, "Fault Tolerance in Distributed Systems," Prentice Hall, 1994.

[7] P. Krishna, Nitin H. Vaidya and D. K. Pradhan, "Recovery in Distributed Mobile Environments," *IEEE Wkshp on Advances in Parallel and Distributed Sys.,* pp. 83-88, Oct. 1993.

[8] S. Rangarajan, K. Ratnam, and A.T. Dahbura, "A Fault-Tolerant Protocol for Location Directory Maintenance in Mobile Networks," *FTCS,* pp. 164-173, June, 1995.

[9] R. D. Schlichting, F. B. Schneider, "Fail-stop processors : An approach to designing fault-tolerant distributed computing systems," *ACM TOCS,* pp. 222-238, Aug. 1983.

[10] K.S. Trivedi, "Probability and Statistics with Reliability Queueing and Computer Science Applications," Prentice Hall, 1982.

[11] N. H. Vaidya, "On Checkpoint Latency," *Pacific Rim Fault Tolerant Sys.*, December, 1995.

[12] J. W. Young, "A First Order Approximation to the Optimum Checkpoint Interval," *Communications of the ACM,* Vol. 17, pp. 530-531, Sep., 1974.