Recoverable Mobile Environment: Design and Trade-off Analysis*

Dhiraj K. Pradhan P. Krishna Nitin H. Vaidya
Laboratory of Computer and Digital Systems
Department of Computer Science
Texas A&M University
College Station, TX 77843-3112

Abstract

The mobile wireless environment poses challenging problems in designing fault-tolerant systems because of the dynamics of mobility, and limited bandwidth available on wireless links. 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 transparent to the user.

Presented here are schemes for recovery upon a failure of a mobile host. This paper portrays the limitations of the mobile wireless environment, and their impact on recovery protocols. Toward this, adaptation of well-known 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 primarily depends on (i) the wireless bandwidth, (ii) the communication-mobility ratio of the user, and (iii) the failure rate of the mobile host.

1 Introduction

A distributed system with mobile hosts is composed of a static backbone network and a dynamic wireless network [5]. A node that can move while retaining its network connection is referred to as a *mobile host*. A static network is comprised of the fixed hosts and the communication network. Some of the fixed hosts, called *base stations*, are augmented with a wireless interface, providing a gateway for communication between the wireless 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 surrounding region, referred to as a base station's *cell*. A mobile host can reside in the cell of only *one* base station at any time. Because of mobility, an active mobile host moves from cell to cell. Thus, when a mobile host moves from one cell to another, the base station responsible for the mobile host must be changed. This process, known as *handoff*, is transparent to the mobile host. Thus, end-to-end connectivity in the dynamically changing network topology is preserved transparently.

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 [5]. Limitations in battery power make disconnections from the network very frequent. Because of their frequency, disconnections must be treated differently than failures. The difference between disconnection and failure is its *elective* nature. Disconnections can be treated as *planned* failures, which can be anticipated and prepared for [5]. The wireless link is equivalent to an intermittently faulty link, which transmits the correct message during fault-free conditions, and which stops transmitting upon a failure. Disconnections and weak wireless links primarily delay the system response, whereas a host failure affects the system state. Strategies are developed in this paper which tolerate failure of the mobile host. Transient failures which affect the mobile host, as well as permanent failures, are handled. It may be noted that the wireless link failure can be treated as a host failure, as well. When a mobile host fails, it results in a loss of its volatile state. The mobile host is assumed to be *fail-silent*; i.e., the base station is able to detect the failure of the mobile host. One way to implement it is to require that an active mobile host send periodic beacons to the base station.

It will now be discussed why traditional fault-tolerance schemes cannot be applied to a mobile wireless environment. Some of the differences between static and mobile networks are enumerated in Table 1.

Traditional fault-tolerance schemes like checkpointing and message logging [6, 9] require a stable storage for saving the checkpoint and the logs. It has been pointed out [2] that while the disk storage on a static host is stable, the stability of any storage on a mobile host is questionable, for obvious reasons such as dropping of laptops or effect of airport security systems [3]. Thus, a mobile host's disk storage cannot be considered stable and is uniquely vulnerable to catastrophic failures. Moreover, all mobile hosts are not necessarily equipped with disk storage. Thus, we need the stable storage to be located on a static host. An automatic candidate is the 'local base station', where the

*This research is supported, in part, by the Texas Advanced Technology Program under grants C-009741-052 and C-999903-029.

Category	Static Wired Networks	Mobile Wireless Networks
Network char.	Uniform, Non-varying	Non-uniform, Varying
Host's local disk	Stable	Unstable
Stable storage location	Static	Mobile
Key perf. parameter	Failure rate	Failure rate, wireless bandwidth, mobility
Perf. metrics	State-saving cost, Recovery cost	State-saving cost, Recovery cost, Handoff time

Table 1: Difference Between Static Wired and Mobile Wireless Networks: Recovery Perspective

local base station is the base station in charge of the cell in which the mobile host is currently residing. Traditional recovery schemes are not applicable because these mobile hosts move from cell to cell. Thus, a mobile host does not have a fixed base station to communicate with. Also, recovery is complicated because successive checkpoints of a mobile host may be stored at different base stations. This dynamic topological situation warrants formulation of special techniques to recover from failures. Also, some of the failure modes are peculiar to the mobile network not present in a static network.

Traditional fault-tolerant schemes do not consider the disparity in the network characteristics (bandwidth, error) of the static network and the wireless network. Moreover, the network characteristics (bandwidth, error) of the wireless network also vary with the type of network used (infrared, packet relay, satellite, etc.). Over a length of a connection, the mobile host might be employing different types of wireless networks. For example, within a building, infrared will be used; in a campus environment, packet relay will be used; and in a remote region, satellite will be used. Available wireless bandwidth and error conditions will be different in each of these wireless networks. Thus, the appropriate recovery protocol needs to be determined adaptively, based on the characteristics of the underlying wireless network.

Performance of traditional recovery schemes primarily depends on the failure rate of the host [8, 12]. However, in a mobile environment, due to mobility of the hosts and limited bandwidth on the wireless links, parameters other than failure rate of the mobile host play a key role in determining the effectiveness of a recovery scheme. A mobile environment is determined by the mobility, wireless bandwidth and the failure rate. This paper presents the following:

- User transparent recovery with mobile host failure.
- Trade-offs for the recovery schemes proposed.
- Optimal recovery scheme for an environment.

We propose several schemes for recovery from a failure of a mobile host. These proposed schemes have two major components: a *state-saving* scheme and a *handoff* scheme. We propose two schemes for state-saving, namely, (i) *No Logging* (N) and (ii) *Logging* (L), and three schemes for handoff, namely, (i) *Pessimistic* (P), (ii) *Lazy* (L), and (iii) *Trickle* (T). We denote a recovery scheme that employs a combination of a state-saving scheme, X ($X \in \{N, L\}$), and a handoff scheme, Y ($Y \in \{P, L, T\}$), as XY. For example, LL is a recovery scheme that uses a combination of the Logging scheme for state-saving and the Lazy scheme for handoffs.

Each combination provides some level of availability and requires some amount of resources: network bandwidth, memory, and processing power. Through analysis, we show that there can be no single recovery scheme that performs well for all mobile environments. However, we determine the optimal recovery scheme for each environment, as shown in Figure 1.

Mobility	Wireless Bandwidth	Failure Rate	Optimal Scheme
High	Low	Low	*LL*
		High	*NT*
	High	All	*LT*
Low	All	All	*LL*

Figure 1: Optimal Recovery Scheme

This paper is organized as follows. Section 2 overviews related work. Section 3 presents the recovery strategies. Section 4 gives the performance analysis of the recovery strategies, and conclusions are found in Section 5.

2 Related Work

Research in mobile computing primarily has focussed on mobility management, database system issues, network protocols, disconnected operation and distributed algorithms for mobile hosts [5, 7]. Work on fault-tolerance issues is very limited.

Alagar et.al. [1], demonstrate schemes to tolerate base station failures by replicating the information stored at a base station, at several "secondary" base stations. Strategies for selecting the secondary base stations were shown. These schemes can easily be integrated with the recovery schemes presented in this paper, to provide a system that tolerates both base station and mobile host failures.

Rangarajan et.al. [10], 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 old and new location information. The protocol also tolerates the corruption of these logical timestamps.

Acharya et.al. [3], identify the problems with checkpointing mobile distributed applications, presenting an algorithm for recording global checkpoints for distributed applications running on mobile hosts.

In this paper, however, we consider protocols to recover from failure in a mobile host, independent of other hosts in the system. Also, we study the effect of mobility and wirelessness on such recovery protocols.

3 Recovery Strategies

A recovery strategy essentially has two components: a state-saving and a handoff strategy. This Section presents two strategies for saving the state, and three strategies for handoff, to achieve fault-tolerance. Strategies for saving the state are similar to traditional fault-tolerance strategies.

3.1 State-Saving

State-saving strategies presented in this paper are based on 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.

It was indicated earlier [2] that a mobile host's disk storage cannot be considered stable. Thus, our algorithms use the storage available at the base station for the cell in which the mobile host is currently residing, as the stable storage.

Multiple hosts (both static and mobile) will take part in a 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 messages sent by a mobile host will first be sent to its base station, which will forward them to the destination host (static or mobile).

Two strategies to save the process state [6] will be discussed here: (i) *No Logging* and (ii) *Logging*. It is assumed that the mobile host remains in one cell during the length of the application. This is followed by a discussion of three schemes that address the recovery steps needed because of 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 user input. The messages or inputs that modify the state are called *write* events. (If semantics of the message are not known, in the worst case, we might have to assume that the state gets altered upon receipt of every message or user input). 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 resumes operation. Importantly, need for frequent transmission of state on the wireless link is a limiting factor for this scheme.

• *Logging* Approach (denoted as L): This approach is rooted in "pessimistic" logging [4], used in static systems. 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 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 cannot send a response. Only upon receipt of the acknowledgment does the mobile host send 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 until a new checkpoint is backed up at the base station. The base station then purges the log of the old write events, along with the previous checkpoint.

After a failure, when the mobile host restarts, the host sends a message to the base station, which then transfers both the latest backed-up checkpoint of the host, as well as the log of write events, to the mobile host. The mobile host then loads the latest backed-up checkpoint and restarts executing, by replaying the write events from its logs, thus reaching the state before failure. Below, the recovery steps are considered which are needed, arising due to mobility of the hosts.

3.2 Handoff

The mobility warrants a special *handoff* process, described below. The key problem to be addressed is how a recovery can be effected if a mobile host moves to a new cell, as illustrated in the following example.

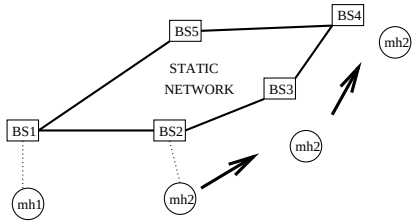

Figure 2: Handoff in the Middle of an Execution

Consider the system in Figure 2. *BSi* denotes i-th base station, and *mhi* denotes i-th mobile host. Here, mobile hosts *mh1* and *mh2* are executing a distributed algorithm. The mobile host *mh*2 has saved both its checkpoint and message log at *BS*2. In the middle of the execution, *mh2* moves to the cell of *BS3*, and then to the cell of *BS4*. Handoff occurs at both the boundaries of *BS2* and *BS3*, and *BS3* and *BS4*. Let a failure of the mobile host *mh2* occur upon reaching the cell of *BS4*. Had *mh2* remained in the cell of *BS2*, the system would have recovered because the 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 identify the base station where the checkpoint is saved. This will warrant additional steps to identify the base station. Therefore, what is proposed is transferring during the handoff process some *information* regarding the state of the mobile host. The following delineates 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 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 checkpoint and/or the log, the new cell's base station sends an acknowledgment to the old base station. The old base station, upon receiving the acknowledgment, purges its copy of the checkpoint and the log, since the mobile host is no longer in its cell.

The chief disadvantage to this approach is that it requires a large volume of data to be transferred during each handoff. Potentially, this can cause long disruptions during handoffs. However it can be avoided if we use the *Lazy* or *Trickle* strategy, as explained.

3.2.2 Lazy Strategy (L)

With *Lazy* strategy, during handoff, there is no transfer of checkpoint and 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 checkpoint 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 is maintained, in addition to the last checkpoint of the mobile host at the base station. Upon a handoff, the new cell's base station keeps a record of the preceding 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 at the base station for each mobile host.

This strategy could lead to a problem if the checkpoint and logs of the mobile host are 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 checkpoint and logs of the mobile host, if present. If a checkpoint is not present, this base station forwards the notification to the preceding base station in the linked list. This process continues, until a base station with an old checkpoint of the mobile host is encountered. All base stations receiving the notification purge any state associated with the particular mobile host.

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

3.2.3 Trickle Strategy (T)

Importantly, 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 entire state information useless.

To avoid this, a *Trickle* strategy is proposed. 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 with *Lazy* strategy.

We make sure that the logs and the checkpoint corresponding to the mobile host are at the "preceding base station" of the current base station[1]. (The preceding 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 always, at most, one hop from the current base station.

To achieve the above, during handoff, a control message is sent to the preceding base station to transfer any checkpoint or logs that had been stored for the particular mobile host. Similar to *Lazy* strategy, the current base station also sends a control message to the new cell's base station identifying the preceding cell location of the mobile host. Thus, the new cell's base station, just retains the identification of the mobile host's preceding cell.

If a checkpoint is taken at the current base station, it sends a notification to the preceding 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 obtains the checkpoint and/or the logs from the preceding base station[2]. The base station then transfers the checkpoint and/or the log to the mobile host. The mobile host then loads the checkpoint and replays the messages from the logs, to reach the state just before failure.

4 Performance Analysis

Basically, six schemes (combinations of state-saving and handoff) are possible. This Section analyzes these schemes, determining which combination is best-suited for a given environment.

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

[2] 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

The following terminology is used, the significance of which will be clearer later in this Section.

- The term *operation* may refer to one of (i) checkpointing, (ii) logging, (iii) handoff, or (iv) recovery.
- **Cost** of an operation 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 $T = 1/\mu$.
- The time interval between two consecutive write events is assumed to be fixed and equal to $1/\beta$. Write events are comprised of user inputs and messages from other hosts. Since we are only interested in the performance penalty due to fault-tolerance of the various schemes proposed, this assumption will not significantly affect the results.
- r: Communication-mobility ratio, defined as the expected number of write events per handoff, 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. The higher the value of α, the costlier is the wireless transmission relative to the wired transmission.
- $N_c(t)$: 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.
- γ: 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_l/C_c).
- C_m: Average cost of transferring a control message over one hop of the wired network. The size of a control message is typically assumed to be much less than the size of an application message.
- ϵ: 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.
- C_r: Average cost of a recovery operation.
- C_t: Average total cost per handoff.

4.2 Modeling and Metrics

The interval between two handoffs is referred to as *handoff interval*. A handoff interval can be represented using a 3-state discrete Markov chain [11, 12], as presented in Figure 3.

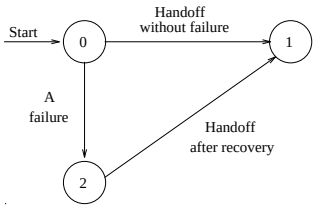

Figure 3: Markov Chain Representation

State 0 is the initial state when the handoff interval begins. During the handoff interval, the host receives messages and/or user inputs (write events). Depending upon 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 significantly affect the results when the average handoff interval is small, compared to the mean time to failure.

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 occurs during the time spent in State 0 before making the transition to State 1. C_{01} is as follows: (Recall that T is the mean handoff interval.)

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

Performance metrics for the proposed schemes are:

• **Handoff Time**: The handoff time is the additional time required to transfer the state information from one base station to other, with the overhead of fault-tolerance. Basically it is 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 restore 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:

$$C_t = C_{01} + P_{02}C_r \quad (2)$$

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 a state-saving scheme, X ($X \in \{N, L\}$), and a handoff scheme, Y ($Y \in \{P, L, T\}$) as C_{tXY}.

Now, we will derive the costs C_{01}, C_r, and the handoff time for each scheme. The total cost C_t for each scheme can be determined by replacing the costs C_{01} and C_r obtained, in Equation 2. Our analysis assumes 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 checkpoint 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. Therefore, the cost of handoff $C_h = C_c + C_m$. Thus:

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

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:

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

4.4 No Logging-Lazy (NL) Scheme

The checkpoint and logging operations are similar to the 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. Thus, the average cost of purging is $N_h C_m$. A handoff operation includes setting a pointer at the current base station, and transferring a control message between the current and the new base stations. 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. Thus:

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

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. 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 checkpoint over the wireless link is incurred: αC_c. The average number of handoff operations completed since the last write event (or checkpoint event) until the time of failure is N_h, where:

$$N_h = \mu T_{cexp} \quad (3)$$

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.

4.5 No Logging-Trickle (NT) Scheme

The checkpoint and logging operations are the 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 the handoff operation is, thus, the sum of the cost of transferring the 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. Therefore, 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 the recovery operation, the number of hops traversed to transfer the state is, at most, one. Thus:

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

$$N_h' = 1(1 - e^{-\mu T_c}) + 0(e^{-\mu T_c}) = (1 - e^{-\mu T_c}) \ , \quad (4)$$

where $e^{-\mu T_c}$ is the probability that the last checkpoint took place at the current base station.

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 come via the base station anyway, 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.) Thus:

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

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:

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

4.7 Logging-Lazy (LL) Scheme

The checkpoint and logging operations are the 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. As also determined earlier in Section 4.4, the purging cost is $N_h C_m$, and the handoff cost is C_m.

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

As determined earlier, the expected number of write events completed until the time of failure since the last checkpoint is $\nu' = \frac{k-1}{2}$. 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,

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

4.8 Logging-Trickle (LT) Scheme

The checkpoint and logging operations are the 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 $N_h' C_m$. Thus:

$$C_{01} = \frac{r\alpha C_c}{k} + \rho r\alpha C_l + \rho r\alpha C_m + \nu C_l + C_c + 2C_m + N_h' C_m$$

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

4.9 Results

The above equations have been normalized 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 a fraction of write events that are user inputs.) This means that the write events comprise an 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 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 needs 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 4 illustrates the variation of $k_{opt_{LL}}$ with r and α for $\lambda = 10^{-2}$ and $\gamma = 0.1$. It can be noticed that

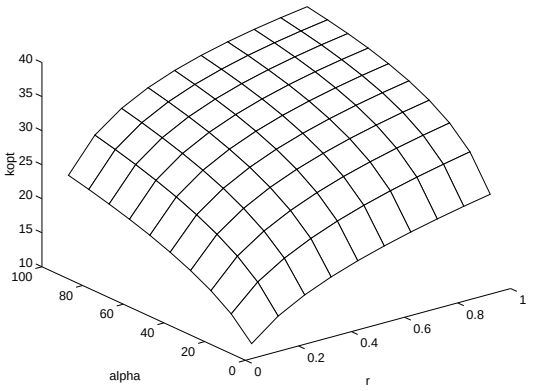

Figure 4: $k_{opt_{LL}}$ vs. r and α: $\lambda = 10^{-2}$, $\gamma = 0.1$

$k_{opt_{LL}}$ 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 illustrates the variation of $k_{opt_{LL}}$, with γ and λ for $r = 0.1$ and $\alpha = 10$. It can be noticed that $k_{opt_{LL}}$ decreases as γ and λ increase. As γ increases, the cost of the 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. Thus, as λ increases, $k_{opt_{LL}}$ decreases.

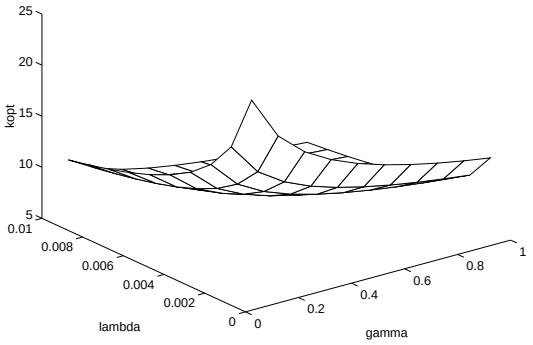

Figure 5: $k_{opt_{LL}}$ vs. γ and λ: $\alpha = 10$, $r = 0.1$

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 additional 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 2 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. 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 2: Handoff Cost and (Handoff Time $\times BW$)

4.9.3 Recovery Cost

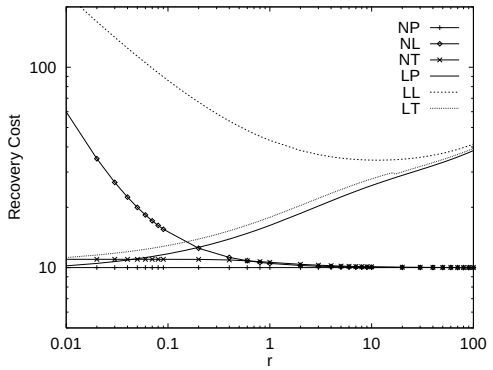

Figure 6: Recovery Cost: $\lambda = 10^{-2}$, $\alpha = 10$

In Figure 6, we plot the recovery cost for all the schemes for $\alpha = 10$, and $\lambda = 10^{-2}$. Similar behavior was observed for other values. As expected, the recovery cost of the Logging schemes is more than the No Logging schemes. The recovery cost of the NP scheme is independent of r. The NP scheme incurs the lowest cost for all values of 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.

The recovery cost of the LP and the LT schemes 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.

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 for 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. The lower the value of r, the greater 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 the LL scheme becomes similar to LP and LT.

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 7 illustrates the variation of total cost of various schemes with r, for $\lambda = 10^{-2}$ and $\alpha = 10$. The total cost is comprised of the failure-free cost and the recovery cost. The total cost of the Pessimistic handoff scheme and the Trickle handoff scheme are almost equal ($NP \approx NT$, and, $LP \approx LT$). The Lazy handoff scheme incurs a 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 remains. The total cost of 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.

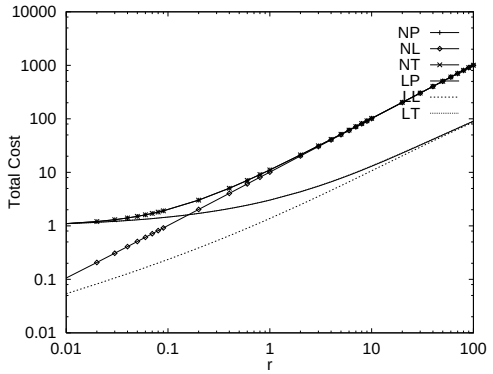

Figure 7: Total Cost: $\lambda = 10^{-2}$, $\alpha = 10$

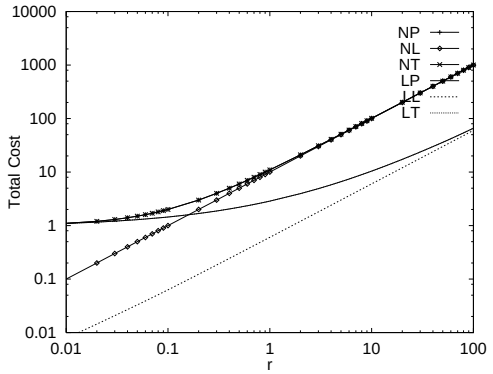

Figure 8: Total Cost: $\lambda = 10^{-5}$, $\alpha = 10$

Figure 8 illustrates the variation of the total cost with r, for $\lambda = 10^{-5}$. Comparison of Figures 7 and 8 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 9 illustrates the variation of the total cost with r, for $\alpha = 500$. The total cost increases with α. Comparison of Figures 7 and 9 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 on the handoff scheme.

4.10 Discussion

Handoff time of Pessimistic handoff schemes is very high, and unacceptable for applications that require connection-oriented services. During a handoff period, there are no packets sent or received by the mobile host. Thus, if handoff time is very high, the communication protocols used for these connection-oriented

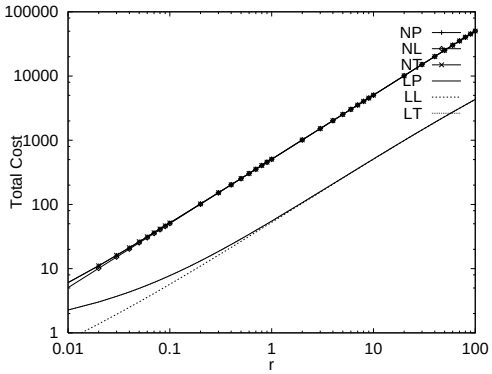

Figure 9: Total Cost: $\lambda = 10^{-2}$, $\alpha = 500$

services might timeout and/or the mobile host might notice long disruption in service during handoffs [5].

Some applications might require a very quick recovery, and some other applications 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 no single recovery scheme that performs best (lowest total cost, lowest recovery cost and lowest handoff time) for all 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 environments.

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 schemes reduces. Since the recovery time (as determined by recovery cost) of the LT scheme is much lower than for the LL scheme for low values of r, it is preferable to choose LT for high α values.

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 schemes 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 preferable. For high r values, LL is preferable over other schemes.

5 Conclusions

Mobile computing's popularity is rapidly increasing. 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 paper 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 paper, and determined those mobile environments where a particular recovery scheme is best-suited.

Currently, we are at work on other problems related to fault-tolerance issues in mobile computing, such as recovery from failure of a base station, fault-tolerant broadcast/multicast protocols, and development of new and efficient distributed recovery schemes.

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," Comp. Sc. Tech. Report, Univ. of Texas, Dallas, Nov., 1993.

[2] R. Alonso and H. Korth, "Database System Issues in Nomadic Computing," *SIGMOD,* June, 1993.

[3] A. Acharya and B. R. Badrinath, "Checkpointing Distributed Applications on Mobile Computers," *IEEE Conf. on PDIS,* Sep. 1994.

[4] A. Borg et. al., "A Message System Supporting Fault Tolerance," *ACM SOSP,* Oct., 1983.

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

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

[7] P. Krishna, "Performance Issues in Wireless Networks," Ph.D. Dissertation, Department of Computer Science, Texas A&M University, 1996.

[8] D. K. Pradhan and N. H. Vaidya, "Roll-Forward Checkpointing Scheme:A Novel Fault-Tolerant Architecture," *IEEE Trans. on Comp.,* vol.43(10), Oct. 1994.

[9] Dhiraj K. Pradhan, "Fault Tolerant Computer System Design," Prentice Hall, 1996.

[10] S. Rangarajan et.al., "A Fault-Tolerant Protocol for Location Directory Maintenance in Mobile Networks," *FTCS,* June, 1995.

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

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