Mobile Computing System Reliability

3.1 Introduction

The mobile computing system is built essentially on top of a wireless network. Such a system consists of distributed applications running on individual mobile nodes that are connected through wireless communication links. Thus the only form of communication between the nodes is message passing. But the wireless link, through which the messages would traverse, suffers from constraints like limited bandwidth, unpredictable delay and environmental noise. The situation is actually even more complicated as most of the link failures in wireless networks are transient and independent in nature. Thus predicting application performance at a given time becomes challenging in such environments. But analyzing performance in terms of availability and reliability is very important for design, deployment and tuning of distributed applications.

The problem of reliability estimation of distributed applications running on wireless networks is addressed in this chapter. The metrics used to estimate reliability are

- Network Coverage
- Two Terminal Reliability [29] and
- All Operating Terminal Reliability [29].

The first one predicts a node’s connectivity to the network while the terminal reliability measures signify performance of distributed applications running at the terminals (mobile nodes in this context). The terminal reliability measures are based on the message passing paradigm, that is, whether a message sent by one terminal can be received by other(s) in consideration.

Recently mobile agents are found to be extremely useful when a continuous connection is impractical and/or bandwidth is scarce such as in wireless computing applications with cellphones and PDAs [95]. In a wireless scenario agents can be utilized by different applications like service discovery, clustering and query processing, efficient routing, intrusion detection etc. (Section 2.4). Agents can also be used to provide fault tolerance of nodes [96] using checkpointing and recovery mechanisms. Here agents are used to locate the MHs and whenever an MH moves further away from its most recent consistent
checkpoint, the agents may coordinate among themselves to move the checkpoint to its (MH) current point of attachment (the MSS with which the MH is presently connected). Thus instead of communicating through passive messages, if active entity like mobile agents are used then effective reliability of the distributed application is expected to improve. So reliability of the agent system and hence that of distributed application running on wireless networks is investigated here.

The next section describes mechanisms for application reliability estimation. This is followed by a thorough discussion on agent reliability for mobile infrastructure networks.

3.2 Reliability Analysis

As mentioned in Section 1.3 the term dependability encompasses attributes like availability (readiness of usage) and reliability (continuity of correct service) [1]. In mobile networks nodes access information through wireless communication links at any time and everywhere (motion and location independence) [24]. Therefore, this environment itself introduces new features and aspects like node mobility, changing network connectivity, power conservation etc. These features affect both availability (readiness for usage) and reliability (continuity of correct service) of the services of the system. For example a node may not deliver its service if

- it moves to a region of no network connectivity or
- bandwidth of the wireless channel is poor or
- the node does not have enough power to transmit.

These significantly reduce the availability of a service provided by the mobile node in a mobile computing system. Due to node mobility and unpredictable environmental noise, the network topology changes with time and connectivity of a node to the network at a certain time cannot be guaranteed here. These facts contribute to the reliability variation of underlying network which becomes a factor that may affect the performance and availability of distributed applications running on mobile wireless networks. Hence reliability and availability of distributed applications must be analyzed for determining the effective strategy of designing them for mobile networks.

To analyze reliability of mobile computing system, traditional reliability measures (used in static distributed systems) are extended to find a scalable approach for estimating system reliability, that is, the probability of successful operation. Due to analytical and computational complexity of developing a closed-form solution for reliability of wire-
3. Mobile Computing System Reliability

less networks, simulation methods, specifically Monte Carlo simulation [28], is used to analyze reliability of such systems.

3.2.1 Mobile Infrastructure Networks

A mobile infrastructure network consists of both Mobile Hosts and static Mobile Support Stations nodes. A set of dynamic and wireless communication links can be established between an MH and an MSS, and a set of high-speed communication link is assumed among the MSSs. An MSS may communicate with a number of MHs but an MH at a time communicates with only one MSS. An MH communicates with the rest of the system via the MSS it is connected to. This is shown in Figure 3.1. The links in the static network may support FIFO message communication [97]. We assume the MSSs and MHs belong to cellular network architecture. Thus the coverage area of one MSS is approximately hexagonal. As an MH moves from one cell (covered by one base station) to another (covered by another base station), wireless channel to the old MSS is disconnected and a wireless channel to the new MSS is allocated. The state of the MH at the time of disconnection is available from the old MSS. Moreover, as long as an MH is connected to an MSS, the channel between them ensures FIFO communication in both directions. Message transmission through these links takes an unpredictable but finite amount of time. Reliable message delivery is assumed during normal operation, that is, there is no message loss or modification.

The MSSs are assumed to be fault-tolerant. The MSS to which an MH is connected

![Figure 3.1: Example of an infrastructured wireless network](image)
to, is referred to as the Home Station (HS) of that MH.

There is no shared memory or common clock among the nodes and communication and synchronization between the nodes is via message passing only. Mobile IP [98] is used as the underlying protocol for message transmission. Only local events take place at MH during the time it is disconnected from the network.

Distributed applications can run on such MHs which are connected to each other through some MSS using wireless links.

If the Cartesian distance between MH$_i$ and MSS$_j$ is less than the radius of the cell covered by MSS$_j$, then MH$_i$ is connected to MSS$_j$ and its link state is said to be connected. Due to mobility the MHs may move to new locations. Thus link state is a function of time. So, link state denotes the network connectivity for the MHs at a particular time instant. But even when the sender and receiver of a message are connected to the network through same/different MSS, a message will successfully reach its intended receiver only after a (propagation) delay bounded by $t_{\text{max}}$. Let the connectivity of MH$_i$ to the network at time $t$ is denoted by $\omega_i(t)$ that can have values 1 (if connected) or 0 (if disconnected). Thus in the worst case, if MH$_i$ sends a message and the receiving MH$_j$ is still connected to the network after $t_{\text{max}}$ time of message transmission, then $\omega_i(t) = 1$ or else $\omega_i(t) = 0$ (when MH$_i$ is disconnected or the receiving MH$_j$ is disconnected within $t_{\text{max}}$). Here the approximate position of the MH after the delay (specified using a mobility model) is speculated with the assumption that in the mean time there is no change in velocity of the MH. Since in reality sudden change in velocity ($d/dv(t) = \infty$) is not possible so we can expect to get approximately the real behavior by maximizing the delay to its upper bound. Consequently it is assumed that the node will be almost in the same position according to the mobility model as in reality. Thus

Distance covered by a node in reality $\approx$ Distance covered by a node according to the mobility model

But links may also fail if the MH itself undergoes a failure. Though not all faults are recoverable, severity caused by a fault may be minimized by adding the widely used fault tolerance mechanism of checkpoint based rollback recovery. In this mechanism the time for taking a checkpoint during normal execution of the application is an overhead. But upon failure, amount of lost computation can be reduced by recovering from the most recent consistent checkpoint [97]. In the present system the time for taking a checkpoint is assumed to be negligible as is the case if nonblocking checkpointing scheme [97] is used and a separate software module or the MSS can take care of this activity of taking checkpoints.
3. Mobile Computing System Reliability

3.2.2 Reliability Estimation

To analyze reliability of a mobile computing system, reliability of individual MHs and that of links between MHs and MSSs need to be considered as MSSs are taken to be fault-tolerant. Reliability \( r_{e_i}(t) \) of a given node is considered to be the probability of successful operation by MH \( i \) at time \( t \). Since the MHs may fail, so, \( 0 \leq r_{e_i}(t) \leq 1 \).

The operating MHs get network connection from their respective nearest MSSs thus forming the topology of the network. In this context failures can be of following three types:

- The source may fail (Figure 3.2(a))
- The destination may fail (Figure 3.2(b))
- Any intermediate node may fail (Figure 3.2(c))

Thus Two Terminal Reliability \( 2TR_m \) in such a scenario can be obtained as the probability of a successful path between a source MH \( i \) to a destination MH \( j \) and determined by [28]

\[
2TR_m = P(\omega_i(t) = 1) \tag{3.1}
\]

Here \( \omega_i(t) \) is the connectivity state of the \( i^{th} \) node at time \( t \) with respect to the source node. As long as the network remains connected via some MSS, any MH can send a message to any other. Thus the scenarios in Figure 3.3(a) and 3.3(b) result in \( 2TR_m = 1 \). But if the path between the sender and receiver breaks due to network

![Figure 3.2: Illustration of node failure: (a) Source node fails; (b) Destination node fails; (c) Intermediate node fails](image)
3. Mobile Computing System Reliability

Figure 3.3: 2TR in different network topology: (a) In a complete graph topology; (b) In a connected graph topology; (c) In a partitioned network topology.

partitioning then $2TR_m$ will be less than 1 (as shown in Figure 3.3(c) where source is MH$_1$ and destination is MH$_4$).

Here it is assumed that a distributed application is running at the source and destination and the application reliability at an instant $t$ depends on the probability of correct operation of the two MHs and the probability of successful message transmission from the source to the destination.

The percentage of MHs connected to the network at time $t$ is referred to as Network Coverage (NC), represented by $\psi(t)$. It can be calculated [28] as

$$\psi(t) = \frac{\sum_{i=1}^{N} \omega_i(t)}{N}$$  \hspace{1cm} (3.2)

Here $N$ denotes the total number of nodes in the network. Finally, another metric All Operating Terminal Reliability (AOTR) [28] is defined which is an extension of All Terminal Reliability (ATR). ATR is the probability that all terminals are connected to the source node whereas AOTR reflects the connectivity of only the operational nodes with the source. Thus in a scenario where nodes do not fail, AOTR becomes synonymous to ATR. But AOTR distinguishes a crashed MH from an MH disconnected due to lack of network connectivity. It may be defined as

$$AOTR = \frac{\prod_{i=1}^{N} check_i(t)}{N}$$ \hspace{1cm} (3.3)

Here $check$ is a variable that is set to 0 if MH$_i$ has failed and is set to $\omega_i(t)$ otherwise. AOTR is relevant because (unlike ATR) it measures the application’s resilience to application instance failures at individual nodes (for example crash failure) [29]. Thus AOTR
3. Mobile Computing System Reliability

indicates the ability to self-heal and maintain cohesion within the distributed application running on the network for the nodes that remain operational at some level.

Node mobility brings two new challenges to traditional reliability calculation:

- **Handoff** - Changing point of attachment (to the network) from one MSS to the other takes appreciable amount of time. So if a mobile node moves at very high speed (user using mobile device sitting in a car on highway) handoff may not be supported that frequently. By the time one handoff is successful, the device has almost crossed the new cell boundary. Thus in that case even if the MH is within the coverage area of some MSS, connection cannot be made. This is taken care of in the proposed reliability estimation technique. A handoff tolerance limit (H) is fixed for all MHs. If handoff frequency crosses this limit, network connectivity cannot be provided.

- **Intermittent Disconnection** - In reality connection to the network is not continuous, rather it is separated by intermittent disconnection intervals. For example user may enter into a tunnel where network connectivity cannot be provided. Such temporary disconnection can be tolerated if the Home Station acts on behalf of the corresponding MH during disconnection. For example, the HS may buffer any message destined for the MH and forward it later. But longer disconnection intervals should not be tolerated as it may signify node failure. Consequently in the reliability estimation, a predefined disconnection tolerance threshold (D) is taken into account. If an MH gets disconnected for less than D time interval, the MH is considered to be connected here, as the HS plays the role of the MH during this time.

The value of both the tolerance limits H and D depend on the network type and mobility pattern of users.

As discussed earlier, it is not possible to develop a closed-form solution for reliability of wireless networks, so, it is also difficult to design a simple but perfect algorithm for reliability estimation. Rather simulation methods, specifically MC simulation, are used to analyze reliability of such systems. These algorithms estimate the reliability and for large number of trials the average reliability estimated converges to its theoretical value. It takes snapshot of the system every $\Delta t$ seconds. The nodes during this time may move according to some mobility model and the link state can also be found out. The metrics (2TR, NC and AOTR) are calculated for the time instant. This process is repeated every $\Delta t$ seconds for the entire simulation time (say T) and average of the metrics are calculated. Then this entire procedure is repeated Q times according to MC simulation.
and the metric values are averaged over $Q$. Thus data is captured in each $\Delta t$th step of $q$th simulation run, that is, for each $(q, \Delta t)$ combination. As $Q$ becomes sufficiently large this metric converges to its theoretical value.

The assumptions made for designing the $MC$ algorithm are as follows.

1. The mobile nodes (that is the MHs) may start from different location but the starting position of the nodes is known, and nodes are connected if they are within a defined transmission range \textit{(radius of the cell covered by the MSS)}. The positions of the MSSs are also known in advance which is quite realistic.

2. All nodes have the same \textit{failure probability} (for simplicity).

3. A link between an MSS and MH either exists or it does not exist. As shown in [28], in practice, links may have a diminished capacity as path loss increases. But here we assume that if the link exists, the capacity is sufficient to pass all traffic taking that route.

4. Node failures are statistically-independent of each other. Nodes are assumed to be either completely operating or completely failed.

5. Whenever the distance between an MH and MSS is within a specified range, then that MH can connect to the MSS. Presently any other environmental factors are not considered here.

The algorithm is summarized in (RwoFT()) Algorithm 1.

In this algorithm the mobility model chosen to simulate node movements is Smooth Random Mobility Model (SRMM)[100]. This model provides temporal dependency such that no MHs can take sharp turns or sudden stops.

As already mentioned nodes may fail due to many reasons. In wireless networks components wear out, battery power drains with time. So, with time failure rate is also expected to rise. To represent such scenario in Algorithm 1, Weibull distribution [28] is used to calculate node reliability $r_i(t)$ as follows

$$r_i(t) = P(n_i(t) = 1) = \exp\left(-\frac{t}{\theta}\right)^\beta$$

(3.4)

Here the operational status of MH$_i$ is represented by $n_i$ where $i = 1, 2, \ldots, N$ where $n_i$ can be either $0$ (if failed) or $1$ (if working). So $n_i(t)=1$ when MH$_i$ is working perfectly. Here $\theta$ and $\beta$ represent scale and shape parameter of Weibull distribution [28] respectively.
3. Mobile Computing System Reliability

**Algorithm 1: RwoFT()**. An algorithm for estimating reliability where MHs are not fault tolerant.

**Input:** Vertices with initial position, initial speed and maximum acceleration of each node

**Output:** NC, 2TR and AOTR

```plaintext
begin
  1  Repeat Q times do
  2    for simulation time T do
  3      for each node i do
  4        Calculate $n_i(t)$ according to equation 3.4
  5        if $re_i(t) > \text{LIMIT}$ then
  6          Node locations are predicted following mobility model (here, SRMM)
  7          MH$_i$ is assigned to its nearest MSS (if any)
  8        for each MH$_i$ within range of an MSS do
  9          MH$_i$ sends a message to a list of MHs randomly selected from the set of working ($n_i(t) = 1$) nodes
 10        if the receiving MH$_i$ remains within the range of an MSS for end-to-end delay [99] time then
 11          if handoff rate of an MH reaches a predefined limit (H) // Tolerates user mobility
 12            $\omega_i(t)$ is set to 0
 13          else $\omega_i(t)$ is set to 1
 14        else if MH remains disconnected for less than a predefined threshold (D) then // Tolerates Disconnection
 15          $\omega_i(t)$ is set to 1
 16        else $\omega_i(t)$ is set to 0
 17        $\psi(t)$ at $t^{th}$ instant is calculated using equation 3.2
 18      end
 19    end
 20  end
 21
 2TR is calculated as $2TR = \frac{1}{Q} \sum_{q=1}^{Q} \frac{1}{T} \sum_{t=0}^{T} \omega(t)$
 22 Overall NC is calculated as $\psi = \frac{1}{Q} \sum_{q=1}^{Q} \frac{1}{T} \sum_{t=0}^{T} \psi(t)$
 23 AOTR is calculated following equation 3.3
end
```
Reliability can improve if fault tolerance mechanisms are in place. Thus the algorithm RwoFT() (Algorithm 1) is modified to include fault tolerant mobile nodes for reliability calculation. During simulation, a process is assumed to be failed at time instant $t_1$ if the value generated by the Weibull distribution exceeds a predefined limit (say LIMIT). The scale and shape parameters are so chosen such that, a process fails with higher probability as time passes. This is because, the probability of malfunctioning of hardware components and surfacing of faults in software components increases with time.

Here the failure of a process is categorized as either a recoverable fault that can be recovered using fault tolerance mechanism, as in [96], or a permanent fault which cannot be recovered during the runtime of the application. In case of process failure, an independent checkpointing mechanism as discussed in [96] is in place that uses software agents to coordinate the activity of message logging and tracks the location of the checkpoint. Here the large memory of the MSSs is utilized to keep message logs and checkpoints. A mobile agent keeps a mobility profile that includes the MSSs where the message logs and/or checkpoints are stored in the last checkpoint interval. Received messages are logged at the MSS that forwards the message to the receiver (last hop). Unacknowledged messages are logged at the sender side.

Since appearance of a fault is a discrete event and mostly the fault is recoverable, it can be represented using Poisson process where values cluster around the mean. Thus a Poisson process decides when a recoverable or permanent fault occurs. Obviously permanent fault is rare. Whenever a recoverable fault appears, the process recovers eventually with the help of checkpoints and message logs. The recovery time depends on the propagation delay and the number of messages communicated in the last checkpoint interval. Actually during recovery, checkpoint and message logs needed for recovery are gathered. This transfer time is calculated as in [96]. The checkpoint transfer time ($CTT$) is calculated as [96]

$$CTT = \frac{CheckpointSize}{LinkSpeed} + PropagationDelay$$  \hspace{1cm} (3.5)

While the checkpoint arrives, message logs also get transferred. So the log transfer time ($LTT$) can similarly be calculated to be [96]

$$LTT = \frac{SizeofMessageLog}{LinkSpeed} + PropagationDelay$$  \hspace{1cm} (3.6)

Since these two transfers, that of checkpoint and message logs occur concurrently and after the checkpoint is loaded, messages from logs are needed to be played in order, the
3. Mobile Computing System Reliability

estimated recovery time can be formulated as the maximum \( (\text{CTT}, \text{LTT}_{\max}) \). Whenever a process recovers from failure it is expected to be active at least for a minimum time interval.

To track the most recent checkpoint another input parameter, namely, *mobility profile* is introduced that is maintained for each MH. In [96], this is maintained by a mobile agent. The mobility profile of each MH contains the different MSS ids where messages communicated in the last checkpointing interval are logged and it also contains the location (MSS id) of the most recent checkpoint. While starting recovery this profile structure shows the MSS ids that should be communicated with (for transferring message logs and the checkpoint).

Thus the modified algorithm for reliability calculation for fault tolerant nodes is summarized \( \text{RwFT()} \) in Algorithm 2.

\( \Delta t \) represents the duration when each node is expected to travel in a linear pattern before selecting a new target speed [100] and/or direction.

The MC method also uses this interval size to determine the points at which the connectivity is determined and neglects the state of the network at times between intervals. Thus only a limited amount of information is lost. But \( \Delta t \) should not be so big that the node moves away from the network and the application running on that node get disconnected in the upper layer due to time out. Hence \( \Delta t \) should not be greater than the disconnection threshold. The disconnection threshold varies depending on the type of service the upper layers demand. For lower bound, we have taken the typical end-to-end delay introduced by one handoff.

3.2.3 Results of Simulation

The simulation is carried out in Java and it can run in any platform. The initial positions of MHs and MSSs are read from a file. To simulate mobility of the MHs Smooth Random Mobility model (SRMM) is used. According to SRMM, the position of each MH at a time instant is determined by Cartesian coordinates \((x_i, y_i)\) for each MH\(_i\). To calculate node location, the initial speed \( v_i \) (at time \( t_0 \)) and maximum acceleration \( a_{\max} \) that a node may attain are input to the system. From the initial speed and maximum acceleration, the maximum speed \( (V_{\max}) \) for each node can be calculated. \( V_{\max} \) for all users is set to 30km/hr though only a few nodes may ever attain this speed depending on \( a_{\max} \) and \( v_i(t_0) \).

Here we are mainly interested in determining connectivity of an MH to the wireless network via some MSS as connectivity to some MSS\(_i\) ensures that the MH can communicate with the rest of the network. Thus mobility of MH becomes visible when an MH
Algorithm 2: RwFT. An algorithm for estimating reliability where fault tolerant MHs are considered.

**Input:** Vertices with initial position, initial speed and maximum acceleration of each node

**Output:** NC, 2TR and AOTR

```
begin
1 Repeat Q times
2 for simulation time T
3   for each node $i$
4     if MH$_i$ is recovering OR there is a recoverable fault at MH$_i$ then
5         Let the node recover.
6     else if a permanent fault appears
7         Fault at MH$_i$ cannot be tolerated.
8     else  // MH$_i$ is working correctly ($re_i(t) > LIMIT$)
9         Same as in Algorithm 1 (steps 6 and 7).
10        if it is a checkpointing instant according to the local clock then
11            A fresh checkpoint is taken and the message logs and mobility profile are cleared.
12       for each MH$_i$ within range of an MSS
13          Same as in Algorithm 1 (steps 9 to 17).
14        if $\omega_i(t) = 1$ then
15            Message is logged in the receiver side.
16            Mobility profile of receiver is updated according to [96].
17            $\psi(t)$ is calculated as in step 17 of Algorithm 1.
18       2TR is calculated as in step 18 of Algorithm 1.
19       Overall NC is calculated as in step 19 of Algorithm 1.
20       AOTR is calculated in step 20 of Algorithm 1.
end
```
3. Mobile Computing System Reliability

Table 3.1: MH Movement Details

<table>
<thead>
<tr>
<th>Time</th>
<th>x</th>
<th>y</th>
<th>ω(t)</th>
<th>Lt(t)</th>
</tr>
</thead>
<tbody>
<tr>
<td>MH1</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>t</td>
<td>1</td>
<td>5</td>
<td>1</td>
<td>MSS1</td>
</tr>
<tr>
<td>t+Δt</td>
<td>1.09</td>
<td>5.05</td>
<td>1</td>
<td>-1</td>
</tr>
<tr>
<td>t+2Δt</td>
<td>1.18</td>
<td>5.05</td>
<td>1</td>
<td>-1</td>
</tr>
<tr>
<td>...</td>
<td></td>
<td></td>
<td></td>
<td>...</td>
</tr>
<tr>
<td>MH2</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>t</td>
<td>0</td>
<td>4</td>
<td>1</td>
<td>MSS1</td>
</tr>
<tr>
<td>t+Δt</td>
<td>0.09</td>
<td>4</td>
<td>1</td>
<td>MSS1</td>
</tr>
<tr>
<td>t+2Δt</td>
<td>0.18</td>
<td>4</td>
<td>1</td>
<td>MSS1</td>
</tr>
<tr>
<td>...</td>
<td></td>
<td></td>
<td></td>
<td>...</td>
</tr>
<tr>
<td>MH3</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>t</td>
<td>2</td>
<td>2.2</td>
<td>1</td>
<td>MSS0</td>
</tr>
<tr>
<td>t+Δt</td>
<td>2.09</td>
<td>2.2</td>
<td>1</td>
<td>MSS1</td>
</tr>
<tr>
<td>t+2Δt</td>
<td>2.18</td>
<td>2.2</td>
<td>1</td>
<td>MSS1</td>
</tr>
<tr>
<td>...</td>
<td></td>
<td></td>
<td></td>
<td>...</td>
</tr>
<tr>
<td>MH4</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>t</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>MSS0</td>
</tr>
<tr>
<td>t+Δt</td>
<td>1.09</td>
<td>0.01</td>
<td>1</td>
<td>MSS0</td>
</tr>
<tr>
<td>t+2Δt</td>
<td>1.18</td>
<td>0.02</td>
<td>1</td>
<td>MSS0</td>
</tr>
<tr>
<td>...</td>
<td></td>
<td></td>
<td></td>
<td>...</td>
</tr>
</tbody>
</table>

changes its point of attachment to the network from one MSS to other. So connectivity of an MH is determined here by the distance between the MH and its nearby MSS (if any). As in [28], the distance \( d_{ij} \) is calculated as

\[
d_{ij} = \sqrt{(x_i - x_j)^2 + (y_i - y_j)^2}
\]

To describe this let us take an example with 4 MHs (\( N=4 \)) situated at points given in Table 3.1. There are two MSSs at (0,0) and (1,2). The cell radius is taken as 3km.

The distance between MH1 and MSS0 is \( \sqrt{(1-0)^2 + (5-0)^2} \) that is more than 3 but distance from MSS1 is \( \sqrt{(1-1)^2 + (5-2)^2} \) equal to 3. So MH1 is connected to MSS1. In this way the network connectivity of the MHs is found. But even when the sender and receiver of a message are connected to the network through same/different MSS, a message will successfully reach its intended receiver only after a delay bounded by \( (t_{max}=) 250 \) ms [99].

As is evident from Table 3.1, MH3 was initially connected to MSS0 but later on
3. Mobile Computing System Reliability

Figure 3.4: (a) Nodes and node mobility; (b) Variation of network connectivity with time increments

at \( t = t + \Delta t \), it has moved to MSS1 causing a handoff. The individual node reliability is described by Weibull shape parameter \( \beta = 1.5 \) and scale parameter \( \theta = 1000 \). The transmission range \( \tau = 3 \text{km} \) and the maximum and minimum velocity with which an MH can move in the network is 30km/hr (as WiMAX can easily support such user mobility) and 0.1km/hr (for pedestrians) respectively. Finally the network coverage and all operating terminal reliability are calculated. The relative position of the nodes is shown in Figure 3.4(a). The figure shows smooth movement of the nodes thus reflecting SRMM.

Figure 3.5: Variation of two terminal reliability with increasing MC simulation steps

As the MHs move, their network connectivity varies as shown in Figure 3.4(b). Network connectivity of all four MHs are shown in this figure. As the movement is smooth, once an MH loses connectivity, there is very little probability of its regaining connectivity instantly.

Algorithms listed in Algorithm 1 (RwoFT()) and Algorithm 2 (RwFT()) can toler-
3. Mobile Computing System Reliability

ate disconnection up to a certain threshold. To set an upper bound for disconnection threshold the timeout interval for TCP traffic (60 seconds approximately) is considered. Thus even if the MSS buffers packets for the disconnected MH, the application stops if the MH does not reconnect within 1 min and send acknowledgement for the buffered packets. So the upper bound for disconnection interval may be approximated to 1 min.

To set the lower bound the handoff delay is considered. Handoff may occur between two WiMAX cells or between WiMAX and WiFi connection. For the second case (considered as this is slower than the first case) the delay due to handoff is close to 550ms. So, the disconnection threshold is kept between 550 ms and 1 min.

Moreover, it is seen that WiMAX networks can support user mobility of 120km/hour. Knowing the fact that typical cell radius in WiMAX is of the order of few kilometers it may be said that 2 handoffs can be tolerated by an MSS in 1 min. So the handoff threshold is taken to be 2 handoffs/min. The variation of two terminal reliability with

<table>
<thead>
<tr>
<th>Parameter Name</th>
<th>Default Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Mobility Model</td>
<td>SRMM</td>
</tr>
<tr>
<td>N</td>
<td>40</td>
</tr>
<tr>
<td>(\tau)</td>
<td>3km</td>
</tr>
<tr>
<td>(T)</td>
<td>20hrs</td>
</tr>
<tr>
<td>(Q)</td>
<td>1000</td>
</tr>
</tbody>
</table>

\(Q\) (number of simulation steps needed by MC simulation) is plotted in Figure 3.5. As is evident, after \(Q = 1000\) the variation is not that much significant. So \(Q\) is taken to be 1000 for the rest of the experiments.

In this setting, network coverage \(\psi(t)\) turns out to be 0.844 and AoTR becomes 0.375 since MH\(_1\) was initially connected to MSS\(_1\) but it lost network connectivity in the subsequent time interval \(t = t + \Delta t\).

To execute any distributed application on these MHs, the MHs need to communicate with each other via message passing. The propagation delay for such a message is mostly dominated by the delay in the last hop, that is, the wireless link. The last hop delay is taken to be 4 seconds [101]. But if the application running at an MH fails and the checkpoint or message log needed for recovery does not reside at the current MSS then a small (as compared to the last hop delay) inter cell communication delay needs to be considered in the calculation of propagation delay in equation 3.5 and 3.6. The link
speed is taken to be 14.4 Mbps which is a reasonable approximation in WiMAX. On an average, message size is taken to be 300KB.

The above mentioned parameter values are taken as default values for the experiments mentioned below. Any change in these parameters is explicitly mentioned. In all subsequent calculations MH$_1$ is considered to be the source node. The other parameters are kept same as in the previous example.

![Figure 3.6: (a)Reliability variation with time increments; (b)Reliability variation with growing network size](image)

In Figure 3.6(a) the sensitivity of AoTR and $\psi(t)$ with time is shown. AoTR is found to be inversely proportional to the square root of time. With smooth node movement, Network Coverage remains almost stable.

Addition of new MHs does not affect AoTR but their positions relative to the MSSs do. Thus adding new MSSs improve AoTR as well as the Network Coverage as reflected in Figure 3.6(b). It is observed that for the initial configuration detailed in Table 3.2, the near optimal value of number of MSSs is 20. If number of MSSs is increased any further, network coverage changes a little.

To measure the performance of the distributed application executing on the MHs, two terminal reliability can be used. It is shown in Figure 3.7(a) that two terminal reliability improves if longer disconnection intervals can be tolerated by the system (logging messages at the MSS for the disconnected MH). But this in turn increases log of received messages kept at the MSSs. Also longer period of disconnection can cause time-outs in the higher layers.

Figure 3.7(b) provides the study result of handoff performance. If the network is designed to support increased or greater speed for moving nodes, and hence able to tolerate increased handoff rate, then the two-terminal reliability of the system improves. But after a certain speed, saturation is reached when performance does not improve.
3. Mobile Computing System Reliability

![Graph](image)

Figure 3.7: (a) System performance with increasing tolerance of network disconnection; (b) System performance with increasing node mobility

![Graph](image)

Figure 3.8: Timely variation of network connectivity with fault tolerant nodes

significantly with change in maximum supported speed $V_{\text{max}}$.

In Figure 3.8, a comparative study of the two techniques described in Section 3.2.2 is shown. The graph clearly shows that introducing fault tolerance results in better network coverage and the performance gradually improves with time. As time increases more and more nodes tend to fail and if such faults are tolerated performance improves. So this is the reason behind the gradual improvement in performance.

3.3 Reliability Analysis of Agent based Mobile System

In distributed systems instead of message passing, mobile software agents can be sent to coordinate distributed activities. This option seems to be particularly useful when continuous connection is impractical and/or bandwidth is scarce as in wireless networks. While a mobile agent is enroute, the MH spawning the agent need not remain connected.
Upon completing its journey the agent can submit the results to the HS and kill itself without waiting for the owner MH to come up and get connected to the network. But behaviour of an agent depends on the distributed application for which the agent is spawned. However, reliability and availability issues are to be addressed before mobile agent based applications can be deployed commercially in wireless networks.

In a wireless scenario agents can be utilized by different applications like service discovery, query processing, intrusion detection etc. Agents can also be used to provide fault tolerance to nodes [96]. But most of these works do not consider reliability of the agents for the assigned task. Nodes in a network may spawn a number of agents. Agents may form groups also and each group may be considered to be independent of the other. In such cases the task assigned to each group does not depend upon the task assigned to other groups and hence accomplishment of the tasks is also independent of each other. In this work MAS is considered to be consisting of independent groups of mobile agents. The task assignment and accomplishment of each group of mobile agents is not influenced by the presence or movement or computation of any other agents of other groups. However, agents of the same group tolerate same amount of delay for migration and they tend to have similar migration pattern. This is in sync with the concept as the application that has spawned the agents may need to collect data from some specific nodes. For simplicity, it is assumed that each node can create at most one agent group. The amount of delay tolerated by an agent is called its delay limit.

The reliability (R) of MAS is defined as the conditional probability (p) that the system is operational during a period of time [102] subject to the conditions of the underlying network. Consequently the system is said to be fully operational if

- the mobile agents which are enroute are operational (no software fault) and

- the other mobile agents finished their task within delay limit specified by their owner MH

MAS is said to be fully down if all mobile agents spawned are either non-operational or lost (could not retract back in time). Moreover, the system can be partially operational if some of its mobile agents are operational [77]. Partially operational means some agents either exhibit software faults or they could not finish the task they were assigned within stipulated time. Here a model is proposed for finding reliability of such agent based system (in wireless networks) that can be found to be partially operational most of the time. To describe the model we assume the wireless network as described in Section 3.2.1. The MHs move according to Smooth Random Mobility Model [100]. But in order to incorporate environmental factors and hence make the environment more realistic,
received signal power is calculated according to two ray propagation [103] of radio signals. A link is considered to be failed if the receiver receives a signal through the link having inadequate power (less than a predefined threshold). Even if the adjoining nodes do not change their position, this received signal power varies with time due to environmental changes like frequency selective fading [104], heavy rainfall, short term fading [104] etc. This features transient nature of the faults [1]. The details of received power calculation following two ray propagation model is skipped here and is discussed in next chapter in Section 4.2.2.

The model of reliability estimation is described in two parts-

1. Modelling mobile agents in wireless networks and

2. Reliability estimation of the above model

The terms used in this section are summarized in Table 3.3.

<table>
<thead>
<tr>
<th>Terms</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>N</td>
<td>Number of Nodes(MHs)</td>
</tr>
<tr>
<td>NoM</td>
<td>Number of MSSs</td>
</tr>
<tr>
<td>M</td>
<td>Number of mobile agents that constitutes MAS and are deployed in the network</td>
</tr>
<tr>
<td>G</td>
<td>Number of Agent Groups</td>
</tr>
<tr>
<td>X</td>
<td>Number of nodes that are more important for an agent to visit</td>
</tr>
<tr>
<td>$p^k_i$</td>
<td>Priority of MH$_i$ for agent k</td>
</tr>
<tr>
<td>$\lambda_i(t)$</td>
<td>Task route reliability of $i^{th}$ agent in a step of simulation</td>
</tr>
</tbody>
</table>

### 3.3.1 Modeling Mobile Agent Based System

Let us consider that there are $G$ ($\leq N$) groups of mobile agents each consisting at least one agent. Thus there are $G$ owner nodes in this network. $M$ agents ($M \geq G$) from $G$ groups start their journey from these $G$ nodes. Each owner provides the list of nodes that will be required by the group to visit, to accomplish the assigned task. It is also not necessary that each group of agents will get the same list of nodes to visit. This is because the owner of the group may distribute the list of nodes to visit among the agents of its group. Thus each agent gets a list of nodes ordered by priority whose information is needed by the owner. During simulation Normal distribution is used to generate this
priority list of nodes for the agents where the mean of the distribution is the same for each group and standard deviation (X) determines the number of higher priority nodes to be visited. Normal distribution is used here as it is analytically traceable. Since a normally distributed variable has a symmetric distribution about its mean, the priority list generated for the agents of one group will not be same but similar. Agents from a group show more similarity in migration pattern when smaller number of high priority nodes (X) is assigned to them. But as more and more nodes get higher priorities, it is infeasible to ask all agents to visit all higher priority nodes. So, the job is divided among them causing little similarity in migration pattern of an agent group. Thus agents of the same group share the same mean (of Normal distribution) resulting in similar priority lists. But a node may be given different priority by different owners. Also an agent (a) may not always choose a yet unvisited node having highest priority. Rather it may migrate to a lower priority neighboring node (j) (attached to same or nearby cell) than a higher priority distant node (k) as follows

\[
\text{next} = j \text{ if } (d_{\text{curr},k} - d_{\text{curr},j}) > \text{offset} \\
= k \text{ otherwise}
\]

Here \(d_{\text{curr},j}\) denotes the distance between an agent’s current host site (curr) and next probable destination host (j). \(\text{offset}\) represents the extra distance of the high priority node as compared to a low priority nearby node. This strategy helps an agent to quickly cover a part of the connected network that may not remain connected after some time. Thus overall performance of the agents may improve.

Each mobile agent may have different delay limit (assigned by the owner) [3]. So if there is a path between the current position and next destination of a mobile agent (with end-to-end delay much less than the remaining time of the agent) and both nodes are operational, only then the agent will visit its next destination taking that path. Otherwise depending on how much time the agent has to finish its task (before a time-out occurs at its owner), the agent may choose to wait as in [3]. If the agent fails to migrate to that destination despite several attempts, then it chooses a new yet unvisited (possibly low priority) destination from its list to migrate to. Retries are important for handling transient fault at/near the chosen destination. For example if an agent fails to migrate in the first attempt because of a transient fault, the fault might shortly be recovered and the agent most likely makes a successful migration in subsequent attempts.

It should be noted here that an agent tries to migrate from \(\text{MH}_i\) to \(\text{MH}_j\) means that the agent sends its replica to \(\text{MH}_j\). If the replica works successfully at \(\text{MH}_j\), then a reply
would come and the copy at MH$_i$ would kill itself.

### 3.3.2 Modeling Agent Reliability

In this scenario the reliability of MAS with respect to the network status and its conditions (for example connectivity of the links, path loss probability etc.) is studied. Each agent is expected to visit the nodes from its priority list to accomplish its task. Each group of agents starts its journey from a given node that acts as its owner. It is assumed that a node can own only a single group of agents. In fact, for better performance, the MSS connected to an MH may also spawn agents on behalf of its MH as in [96].

As observed, reliability of MAS ($R_{MAS}$) heavily depends on how much reliably the underlying network performs ($R_{NET}$). We denote reliability of the system ($R_s$) to be a conditional probability as:

$$R_s = R_{MAS} | R_{NET}$$  \hspace{1cm} (3.8)

Reliability of the network ($R_{NET}$) can be treated as an accumulative factor of $(1-P_{node})$ and $P_{link}$. To calculate $R_{NET}$ several metrics are described in the previous Section 3.2.2 like NC [105] which indicates the ratio of nodes that are connected to the network (among total number of nodes).

The way working conditions at nodes and their connectivity ($R_{NET}$) affects $R_{MAS}$ is shown below. If an agent $i$ visits $j$ nodes with different priorities ($w_j$) then its reliability ($\lambda_i$) will be a product of two factors - weighted average of the visited nodes and individual software reliability (denoted by $r_i(t)$), as follows.

$$\lambda_i(t) = r_i(t) \times \frac{\sum_j w_j l_j}{\sum_j w_j}$$  \hspace{1cm} (3.9)

Here $l_j = 1$ if agent $i$ visits $j$th node from its list successfully. Thus if an agent $i$ has $0 < \lambda_i(t) < 1$ then the agent is partially operational. The value of this metric varies with underlying network conditions. Now, the probability that the MAS is operational for a given network scenario i.e, the system reliability $R_S$ at an instant $t$ can be calculated as the mean of reliability of all its components, that is, the agents in this system.

$$R_s(t) = \frac{1}{G} \sum_{j=1}^{G} \frac{\sum_{i=1}^{\left|\text{group}_j\right|} \lambda_i(t)}{\left|\text{group}_j\right|}$$  \hspace{1cm} (3.10)

Here $\left|\text{group}_j\right|$ represents the number of agents in group$_j$. Thus overall reliability can
be calculated as

\[ R_s^q = \frac{1}{T} \sum_T R_s(t) \]  

(3.11)

Total time for which the simulation is executed is taken as \( T \). Monte Carlo simulation is used here that divides time into some discrete \( \Delta t \) interval and takes snapshots of the system at such intervals. This simulation for \( T \) time units is repeated \( Q \) times so that \( R_s \) eventually converges to its theoretical value as follows.

\[ R_s = \frac{1}{Q} \sum_Q R_s^q \]  

(3.12)

The corresponding proposed algorithm WirelessAgentReliabilityEstimation() listed in Algorithm 3 calculates \( R_s \).

Since in a typical scenario the nodes are not static during the entire tour of the mobile agents, so after every single move the entire network configuration (hence the effect of node mobility) is recalculated.

Here it is assumed that the agent can always retract back to its owner whenever it chooses. This assumption is quite justified as either the MSSs (fault tolerant) spawn agents [96] or the state of a disconnected MH is often available at a particular MSS (its Home [96]).

It may be seen in practice that in a network some nodes have rich information and the agents tend to move to those nodes over the others. That is why, the nodes are prioritized by providing a priority list rather than allow agents randomly select the next destination in step 15 of WirelessAgentReliabilityEstimation() algorithm (Algorithm 3).

### 3.3.3 Results of Simulation

The initial positions of MHs and MSSs are read from a file. The initial positions of agents are assumed to be known. All agents of the same group start from the same node, designated to be the owner. We take the help of an example to show data generated by our simulation program and then the detailed analysis of simulation results are shown. The simulation time is taken to be 3 hours. 6 MHs are considered to be roaming around a network and connected via 3 MSSs. 4 agents (1, 2, 3 and 4) from 2 groups are spawned by nodes MH\(_1\) and MH\(_2\) respectively. Thus MH\(_1\) deploys agents 1 and 2 and the rest are spawned by MH\(_2\). Initially as shown in Figure 3.9 all the MHs were in one region, connected to the wireless network via MSS\(_1\). But MH\(_4\) moves fast and gets connected to MSS\(_2\) in the next time instant. Other nodes change position but stay within the coverage area of MSS\(_1\). Agents are shown by callouts where a dotted

**Input:** M (number of mobile agents in the system), G (number of agent groups)
The initial state of the network (node position, location, speed of the nodes)

**Output:** $R_s$

```
begin
1 Repeat Q times do
2 Initialize $\lambda_j \forall j$ and a source for each mobile agent.
3 List of nodes along with their initial positions is given.
4 The priority list for each agent group is formed according to Normal distribution as described
5 for simulation time T
6 for each node i
7 Node locations are predicted following mobility model
8 Node connectivity is determined according to their proximity to an MSS and two ray propagation model [103].
9 Some nodes may also fail because of software/hardware failure according to Poisson distribution.
10 for all agents (M) in the system
11 Individual software reliability of the agents $r_i(t)$
12 if migration failed in the previous time instant then
13 Retry to move to the same destination.
14 if an agent fails to migrate despite several attempts
15 Go to next step to choose a new destination. // depends on the maximum delay tolerated by the owner
16 An agent picks up a yet unvisited node as its next destination according to equation 3.7.
17 if that destination is connected to the network
18 The agent moves to the new destination depending on the instantaneous background noise level in the network.
19 if agent migration succeeds
20 $\lambda_j$ is set.
21 Calculate agent reliability following equation 3.9.
22 Reset the value of $\lambda_j \forall j$.
23 Calculate MAS reliability for time T using equation 3.11.
24 Calculate system reliability following equation 3.12.
end
```
MH_1 and MH_2 give their agents a list of nodes to visit with different priorities according to Normal distribution as described in Section 3.3.1. Top nodes in the priority list have higher priorities (p_h) and rest have lower priorities (p_l) according to the following set of relations:

\[ p_h = \begin{cases} 
\frac{1}{N'} \left( \frac{N'}{X} - 0.1 \right) & \text{if } N' \leq 2X \text{ initially} \\
\frac{2}{N'} & \text{otherwise}
\end{cases} \] (3.13)

In this equation whenever \( N' \leq 2X \), \( 2/N' \) comes down to \( \geq \frac{1}{X} \) meaning at most \( X \) nodes are assigned equal priorities while the rest have no priority. In order to prevent that \( (0.1/N') \) is subtracted in the first case. \( p_l \) is generated as follows:

\[ p_l = \frac{1 - p_h \times X}{N' - X} \] (3.14)

Here an agent is expected to visit \( N'(\leq N) \) nodes in total including \( X \) number of MHs with higher priorities over the rest. Thus \( N' \) in the above relations represent the length of the priority list. The sum of the priorities is always 1 as shown below:

\[ X \times p_h + (N' - X) \times p_l = 1 \] (3.15)
3. Mobile Computing System Reliability

For example if $N' = 4$ and $X = 3$ then according to equations 3.13 through 3.15, 3 higher priority nodes will be assigned priority of $\left(\frac{1}{4} \times \left(\frac{4}{3} - 0.1\right)\right) = 0.3083$ while the lower priority node is assigned a priority of $\left(\frac{1-0.3083 \times 3}{4-3}\right) = 0.0751$. These set of equations work reasonably well as long as $\frac{N'}{X} > 1.1$. When $N$ and $X$ become approximately equal (in reality this is quite unlikely) 0.1 in equation 3.13 may be replaced by 0.01 or an even smaller value. $N'$ can be different for different agents. The above equations are also valid even if $N'$ is less than the total number of nodes ($N$) in the network (that is, an agent need not visit the entire network). But we have considered $N'$ to be equal to $N$. Thus agents of the same group share the same mean (of Normal distribution) resulting in similar priority lists.

Suppose the application running at MH$_1$ needs prior information from MH$_4$ and MH$_5$ (Figure 3.9). Thus MH$_4$ and MH$_5$ are given higher priorities ($\frac{2}{N' = 5} = 0.4$) for agent 1 following equation 3.13. Similarly MH$_2$ and MH$_5$ may be given higher priority for agent 2. This explains the priority list of the agents shown in Table 3.4. The list is generated using Normal distribution with group owner id as mean and number of high priority nodes as standard deviation. So agents from same owner exhibit similar migration pattern. Negative values are discarded. Since the MHs remain at almost the same region, the agents at first tend to visit the MHs having higher priority values (following equation 3.7). Thus agent 1 moves to MH$_4$. The simulation is carried out for 3 hours and the overall reliability is found out to be 1.00.

<table>
<thead>
<tr>
<th>Agent ID</th>
<th>Priority List</th>
<th>Priority</th>
</tr>
</thead>
<tbody>
<tr>
<td>Agent 1</td>
<td>4,5</td>
<td>0.4</td>
</tr>
<tr>
<td></td>
<td>2,6,3</td>
<td>0.6666667</td>
</tr>
<tr>
<td>Agent 2</td>
<td>5,2</td>
<td>0.4</td>
</tr>
<tr>
<td></td>
<td>6,4,3</td>
<td>0.6666667</td>
</tr>
<tr>
<td>Agent 3</td>
<td>1,3</td>
<td>0.4</td>
</tr>
<tr>
<td></td>
<td>6,4,5</td>
<td>0.6666667</td>
</tr>
<tr>
<td>Agent 4</td>
<td>3,1</td>
<td>0.4</td>
</tr>
<tr>
<td></td>
<td>6,4,5</td>
<td>0.6666667</td>
</tr>
</tbody>
</table>

We did a series of experiments to validate our protocol. The default values of the parameters used in these experiments are listed in Table 3.5. Any change in these values is explicitly mentioned. It is seen through experiments that MAS reliability does not grow appreciably with increasing $M$ rather it is almost independent of MAS size if other parameters remain same. Agent reliability is evaluated with increasing task load.
Table 3.5: Default Values of Simulation Parameters

<table>
<thead>
<tr>
<th>Parameter Name</th>
<th>Default Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Mobility Model</td>
<td>SRMM</td>
</tr>
<tr>
<td>M</td>
<td>30</td>
</tr>
<tr>
<td>G</td>
<td>5</td>
</tr>
<tr>
<td>N</td>
<td>30</td>
</tr>
<tr>
<td>NoM</td>
<td>20</td>
</tr>
<tr>
<td>X</td>
<td>15</td>
</tr>
<tr>
<td>Minimum required signal power</td>
<td>18dBm</td>
</tr>
<tr>
<td>Delay Limit</td>
<td>$N \times \text{avg propagation delay}$</td>
</tr>
<tr>
<td>T</td>
<td>6hrs</td>
</tr>
<tr>
<td>Q</td>
<td>100</td>
</tr>
</tbody>
</table>

on agents by increasing the number of higher priority nodes ($X$) to be visited. Figure 3.10(a) shows the variation of system reliability with varying ratio of priority nodes for different values of network coverage. Here ratio of priority nodes is calculated as

$$\text{Ratio of Priority Nodes} = \frac{X}{N}$$  \hspace{1cm} (3.16)

![Reliability Variation with Nodes having Increasingly Higher Priorities](image1)

![Reliability Variation with Increasing Network size](image2)

Figure 3.10: (a) Variation of agent reliability with increasing number of high priority nodes to be visited; (b) Reliability variation as network gets bigger

When $X$ is small, it is hard to find out the high priority nodes especially when these are distant from the agent’s current position or have undergone a hard handoff [96]. Hence initially reliability improves with increasing number of high priority nodes. But as $X$ increases, according to equation 3.15 the difference between absolute values of high and low priorities decrease. Hence as more and more nodes are assigned higher priorities,
3. Mobile Computing System Reliability

the agent’s job gets tougher resulting in a downturn as shown in Figure 3.10(a). The effect is even more with poor network coverage (NC=0.18). The figure shows clearly that agents perform best when the ratio of priority nodes is 0.5. Thus if an application requires more than 0.5N nodes to be visited by an agent, new agents should be spawned for better performance. It is found that this is true for bigger networks too.

In the experiments the initial positions of the MHs are given in such a manner that network diameter increases with greater number of nodes. As a result the probability of network partitioning increases in bigger networks as more nodes may get disconnected from the network. This decreases the network coverage (percentage of nodes connected to the agent owner, in this context). Also increase in network diameter makes the job of an agent harder. Thus reliability degrades but as Figure 3.10(b) shows, the agents still perform quite well (reliability > 0.8) even in networks with as many as 50 nodes. Although for small networks agent reliability is almost independent of M, as network gets bigger (>25 nodes), more agents (hence bigger MAS) proves to be helpful. But as the network continues to grow further (network diameter increases) causing poor network coverage, reliability of agent based system drops even with moderate number of agents.

In the experiments the initial positions of the MHs are given in such a manner that network diameter increases with greater number of nodes. As a result the probability of network partitioning increases in bigger networks as more nodes may get disconnected from the network. This decreases the network coverage (percentage of nodes connected to the agent owner, in this context). Also increase in network diameter makes the job of an agent harder. Thus reliability degrades but as Figure 3.10(b) shows, the agents still perform quite well (reliability > 0.8) even in networks with as many as 50 nodes. Although for small networks agent reliability is almost independent of M, as network gets bigger (>25 nodes), more agents (hence bigger MAS) proves to be helpful. But as the network continues to grow further (network diameter increases) causing poor network coverage, reliability of agent based system drops even with moderate number of agents.

The effect of network coverage on agent performance is evident in Figure 3.11(a). Here coverage is decreased by either increasing network diameter (by adding nodes at distant places) or decreasing number of MSSs. But in both cases the effect is same, that is shown in Figure 3.11(a). While agents move from one node to another, they dynamically decide about the routes. The agents are also designed to tolerate transient faults and they are found to work with quite high reliability (>0.5) even when coverage is as low as 0.269 (Figure 3.11(a)). Figure 3.11(a) indicates that if MAS is designed in a
3. Mobile Computing System Reliability

way mentioned in Section 3.3.1 then agents can perform reliably even when coverage is poor (disconnection is frequent). Moreover for coverage=0.308 onwards agent reliability reaches almost a stable state. Thus even with a few MSS (less expensive infrastructure) and hence low coverage, applications utilizing mobile agents are found to work quite reliably with moderate number of nodes (30 nodes as shown in Table 3.5).

The infrastructure, hence number of MSS plays a key role in wireless networks towards maintaining connectivity. As more MSSs are installed more and more nodes get connected, coverage gets better that improves MAS reliability in turn. But for number of MSSs=14 where N=30, agent reliability (=0.946) is found to almost stabilize (Figure 3.11(b)). Thus with <0.5N number of MSSs, agents are found to reach a steady state. This is in conjunction with the previous result as 14 MSS can cover as low as 36.7% of the networks. But in such network agents on an average accomplish (as high as) 94.6% of their task assigned in 6 hrs (Table 3.5).

![Timely Variation of Reliability](image1)

![Reliability Variation with Increasing Delay Tolerance](image2)

**Figure 3.12:** (a) Variation of reliability with time; (b) Variation of reliability with increasing limit for delay tolerance

However the time of each simulation run does not seem to affect reliability in the long run. As shown in Figure 3.12(a) despite initial perturbation MAS reliability eventually reaches stable state with time. For smaller number of MSSs, MHs quickly move out of the range of MSSs thereby degrading network coverage. This explains the initial sharp fall in the curve corresponding to MSS=10. But eventually network coverage stabilizes. This helps MAS to attain a steady state with time. Thus with smaller number of MSSs, it takes longer to reach a stable state.

Finally the effect of delay tolerance on agent performance is shown in Figure 3.12(b). The x-axis in Figure 3.12(b) shows the maximum delay limit (in terms of hop count that an agent can make) among 5 agent groups (Table 3.5). The difference in delay limit between group\(_i\) and group\(_{i+1}\) is time to tolerate 1 hop. Thus agents from group\(_i\) can wait
for an extra hop than agents from group $i+1$. Thus delay requirement is most stringent for agents of group $G$ representing real time traffic. But group $G-1$ can tolerate delay for an extra hop than that of group $G$ and so on. Thus tasks assigned to agent groups with lesser ids tend to be best effort type as they can tolerate more delay. The figure 3.12(b) also indicates that agents can perform reliably even in poorly connected network with more number of nodes. The graph shows that as delay limit for an agent increases the agents can beat transient faults at wireless links more efficiently. But eventually agent reliability reaches an almost steady state (so for better performance the delay limit need not be increased infinitely) because poor network coverage can never be overcome totally. Thus it can be observed that for $N=30$ and/or where agents are expected to visit 30 nodes, the maximum delay limit for an agent group is 50 hops. But in bigger ($N=50$) networks as the job of agents gets tougher (it needs to visit more nodes), the agents should be designed to tolerate more delay in order to provide a steady performance. But, interestingly the resulting agent reliability in steady state becomes comparable to the previous case where $N=30$. This proves that distributed applications spawning agents can be made scalable in wireless networks if the delay tolerance limit for the agents is chosen carefully.

### 3.4 Conclusion

A scalable approach to estimate reliability of distributed applications running on infrastructure wireless networks is presented in this chapter. To avoid analytical complexity Monte Carlo simulation is used for this purpose.

In this work the metrics used for calculating reliability of distributed systems are two terminal reliability and all operating terminal reliability. In the reliability calculation, node failure is categorized into two types - recoverable (common) and irrecoverable (rare). It is seen that introduction of fault tolerance proves beneficial in the long run.

In order to improve performance even further message passing mechanism is replaced with that of agents. An MH acting as an owner creates a group of agents having similar delay requirements (according to the requirement of the application) and asks each of them to visit similar node list for some purpose. The agents share their collected information with their owner if they come back in time (within the delay limit). The results indicate that the design of MAS (Section 3.3.2) achieves appreciable reliability even with poor network coverage when number of MSSs <0.5 times number of MHs. Thus the agents are shown to overcome frequent network disconnection. Moreover MAS performs best when half of the nodes that an agent needs to visit are given higher
priorities over the others.

Here MAS is assumed to work on wireless network with static MSSs. But in a purely ad hoc scenario the task of the agents get harder as no dedicated nodes (such as MSS) can be found for coordinating network related task. This is explored in subsequent chapters.