In the *Proc. of the 25th International Symposium on Fault-Tolerant Computing*, June, 1995, Pasadena, CA

A Fault-Tolerant Protocol for Location Directory Maintenance in Mobile Networks

Sampath Rangarajan[1] Karunaharan Ratnam[1] Anton T. Dahbura[2]

ABSTRACT In this paper, we present a fault-tolerant protocol for maintaining location directories in mobile networks. The protocol tolerates base station failures and also allows for consistent location information to be maintained about mobile hosts that switch off and arbitrarily reappear in some other part of the network. Further, the protocol tolerates the corruption of a logical time stamp that is part of any protocol where new location information has to be distinguished from old location information when a location directory is updated. We formally show that the protocol maintains consistent location information and does not overwrite new location information with old location information. The protocol can be hierarchically organized to reduce the message overhead incurred by location directory updates.

1 Introduction

The advent of wireless, or *mobile* networks, consisting of fixed local regions, which support communication among mobile entities has introduced a new and challenging set of problems in distributed systems design. Most notable among these is the need for the fixed portion of the system to efficiently track the whereabouts of the mobile entities [2][3], especially so that information can be routed to the mobile entities in a timely fashion [9][11]. The cell-based structure of the network is an artifact of the need to reuse available bandwidth, which is in general is quite limited.

In the mobile network, base stations (BS), each of which covers a geographical area called the *access zone*, keep track of the *mobile hosts* (MH) that enter that zone [5][14]. These base stations are part of a wired network. The base stations and the mobile hosts communicate through a wireless connection, and access zones of the base stations could overlap. It is the responsibility of the base station which covers an access zone to route outgoing/incoming packets from/to hosts located in that zone at that time, considering that mobile hosts can constantly be moving from one zone to another. For that reason, the base stations need to have recent information about the location of the mobile host at the other end. When a mobile host moves from one access zone to another, either the old base station or the new base station (or both) should disseminate this information to the other base stations, updating the others about the location of this host. This will enable base stations to maintain *location directories* (LD) of all the mobile hosts on the network [8][10][15].

Strategies for maintaining and manipulating data-structures that store location information were presented in [1][6]. The focus of these theoretical studies is on the trade-off between updating and retrieving information from location directories, and not on the problem of designing practically implementable distributed protocols for location directory maintenance. The common goal is to minimize the message complexity incurred during the *update* operation and the latency incurred during the *find* operation. In [4][7] practical IP-based protocols were presented for mobile networking. In the protocol given in [4], information about the whereabouts of a particular mobile host is kept in a single location in the network. When another entity in the network (base station or mobile host) wishes to establish a connection with

---

[1] Department of Electrical and Computer Engineering, Dana Research Building, Northeastern University, Boston, MA 02115.

[2] Motorola Cambridge Research Center, One Kendall Square, Building 200, Cambridge, MA 02139.

a given mobile host, it either must do so through the entity which knows its location or must ask the entity to send it the routing information before establishing the connection, thereby adding to the message latency. In the protocol presented in [7], there is no explicit update operation; instead, base stations query other elements in the network about the whereabouts of a mobile host only when a new connection is to be established. This may lead to excessive latency during the find process.

Our main contribution in this paper is the development of a fault-tolerant distributed protocol for location directory maintenance. In the following, we discuss the need for fault-tolerance in these protocols.

Due to the communication latency which is inherent in distributed systems, any location directory management protocol that maintains changing location information requires mechanisms that enable old location information to be distinguished from new location information. Such mechanisms are required regardless of whether replicated copies of the location information is kept in the base stations on the wired network or only one copy of the location information is kept in a central location on the wired network. For instance, assume that a MH moves from the access zone of base station A to the access zone of base station B. Base station B will update (one or more) copies of the LD with this information by sending a message through the wired network to the site(s) that maintains this directory. Now, assume that the MH moves from the access zone of B to the access zone of another base station C; C will send a message to update the LD to reflect the fact that this MH is within its access zone. If the message from B is received at the site (maintaining the LD) after the message from C is received, then the LD will be updated with the old location information unless time stamps are kept to distinguish old information from new information. Of course, physical time stamps cannot be used as the clocks at the different base stations may not be synchronized. Instead, logical time stamps can be used which are simply a monotonically increasing sequence of numbers with which update messages are tagged. In this case, a more recent update message should have a larger sequence number compared to an older update message.

Given that two different BSs send contiguous (in time) update messages, tagging update messages corresponding to a certain MH with a monotonically increasing sequence number can be accomplished only if a BS is aware of the largest sequence number that has been used so far to tag an update message corresponding to that MH. This is accomplished by letting the MH carry this sequence number with it which can then be passed onto the BS that becomes its monitor next. Such sequencing is essential to any protocol where there is the possibility of old location information overwriting more recent location information.

There are two problems associated with such sequencing. Firstly, if the sequence number carried by the MH becomes corrupted for any reason, consistent sequencing may not be maintained. For example, if the sequence number carried by the MH and communicated to a new monitor base station is smaller than what it should be, then the location information that is tagged by this sequence number will be construed to be old and will be discarded. Secondly, when MHs go into the off mode, they may lose memory and hence, the sequence number that they carry. In this case, when a MH becomes active again it may not be able to provide the new monitoring base station with the correct sequence number.

The protocols that have been proposed to date for location directory maintenance have not accounted for these types of failures, while in the protocol presented in this paper, we tolerate the corruption of sequence numbers (which we call location counters) and we also allow for MHs to wake up and join the mobile network after being switched off. Further, the protocol tolerates the failure of base stations on the wired network as long as the wired network remains connected. Thus, our protocol is fault-tolerant in the following aspects:

1. Any location directory management protocol that maintains changing location information needs sequence numbers that enable old location information to be distinguished from new location information. Our protocol tolerates the corruption of these sequencing mechanisms.

2. When a MH switches off and on again, the sequencing information carried by the MH may be lost. Our protocol tolerates this loss of sequencing information.

3. Base stations can fail, and if they do, the basic mechanisms that are used in the protocol enable the MHs that were monitored by the failed base station to be monitored by other base stations. For this to happen, there must exist another base station that has an overlapping access zone with the failed base station. That is, the MH should be within the access zone of more than one base station. If the MH is not within the access zone of any fault-free base station, then it will not be monitored until it moves into the access zone of some base station.

The location directory can be maintained in a hierarchical fashion, so that in most cases the changing information about the location of the mobile hosts need be disseminated only to the base stations within part of the lowest level of the hierarchy.

The paper is organized as follows. Section 2 describes the system model and defines the terminology used. Section 3 first provides a high-level description of the protocol and then presents details of the protocol, including examples that illustrate its execution. The correctness of the protocol is discussed in Section 4. Conclusions are presented in Section 5.

2 System Model

A mobile network consists of base stations (BS) connected by a wired network and mobile hosts (MH) that move freely around. The wired network is assumed to be a point-to-point network of arbitrary topology. Mobile hosts communicate with base stations through wireless channels which are assumed to be FIFO. Each MH has its own identity (address) regardless of its geographical location within a mobile network. Base stations are responsible for keeping track of MHs within their local access zones (referred to as a cell). A subset of base stations serve as Reporting Stations (RS). Reporting stations communicate with MHs in a wireless sense. They function as gateways between the wired and the wireless sides of the network. In the rest of the paper, we assume that all base stations serve as reporting stations.

Information about the location of all (or a subset of) the mobile hosts is maintained at each base station. This directory of location information is called the Location Directory (LD). A LD could be hierarchical in nature, with Regional Directories (RD) providing information about MHs in a locality. Generally, the information provided enables the location of a MH to within a cell.

In *Active mode*, the mobile host is in normal operation and is actively sending messages to or receiving messages from other mobile hosts on the network. In *Stand-by* mode, the mobile host is not actively sending or receiving messages. It uses only minimum power to maintain status information such as its i.d., the i.d. of the base station that monitors its present location and the location counter value (to be explained later). The mobile host also monitors cell changes. In *Off* mode, the mobile host has its power switched off. A mobile host that has been powered off will not necessarily retain any memory when it is powered on again. In this paper, we do not differentiate between the active and the stand-by modes of operation.

3 Protocol Description

3.1 Overview of the Protocol

In the protocol presented in this paper, base stations that monitor the mobile hosts periodically emit "beacon" signals within their cells. A MH finds out that it is in the cell of a BS when it is able to receive a beacon signal from the BS. The MH could then send a request to the BS requesting to be monitored by the BS. The request is in the form of a *monitor-me-join* (mmj) message. Along with the *mmj* message, information about the identity of the MH and the location counter (LC) value (to be explained later) is also sent. Similarly, when a MH wakes up, it sends a request to a BS requesting to be monitored. This request is in the form of a *monitor-me* (mm) message. Thus, the BS can distinguish requests from MHs that have just woke up from those that have moved into its access zone.

In case the MH enters an area where the cells of more than one BS overlap, the MH may wish to change its monitor if it finds that another BS has a stronger beacon signal. If this monitor's beacon signal becomes weaker later, at that time it could send a request to another BS (if another BS with an overlapping cell is available) requesting to be monitored, and so on.

When a BS receives an *mmj* request from a MH, it sends an *ackmmj* message back to the MH indicating that it is willing to be the gateway for this MH. From then on, whenever a message is sent from/to this MH, this message needs to be directed through this base station. The MH should receive the *ackmmj* message within a certain period of time; if not, it times out and sends an *mmj* request to another overlapping BS if one exists. This procedure is repeated until an *ackmmj* is received. Once an *ackmmj* is received from a BS, the MH is then able to communicate with other MHs on the network through this BS. In the case of overlapping access zones, once the *ackmmj* message has been received and a connection has been established with a new BS, the MH sends a *break* signal to the old BS to severe that connection. Together with the break signal, it also sends the identity of the new BS and an updated location counter value.

When a BS sends an *ackmmj* back to the MH, it also updates its own LD and disseminates information about the new location of the MH to the other nodes in the wired network. While the most recent updated packet is propagating through the network, some of the nodes still maintain old information. Because of this, requests for connection to a MH could be received at a BS that is not monitoring this MH any more. In this case, this request is forwarded to the BS that is currently monitoring the MH as per the location information available at this time at this BS (that received the connection request). It is possible that the location information at the BS (that received the connection request) is old and information about the MH's new location (and location counter value) has not reached this BS yet. In this case, if a request for the MH arrives at this BS and gets forwarded to (say) BS_i, as the MH has moved from BS_i, this request is forwarded by BS_i to the BS that is monitoring the MH as per the information that has reached BS_i so far. This way, through a chain of forwarding steps, the request eventually reaches the BS that is currently monitoring this MH.

3.1.1 Information dissemination and location counter corruption

Each MH has a location counter associated with it. The location counter LC acts as a logical time stamp. Whenever a MH sends an *mmj* message to a BS, the LC value is also sent along with it. The BS increments the LC value by one, sends an *ackmmj* message back to the MH together with the new LC value, updates its own LD, and disseminates information about the new location of the MH together with the LC value to the neighboring BSs in the wired network. The base stations that receive this information determine if this information is new information (*newinfo*), old information (*oldinfo*), or same information (*sameinfo*), depending on whether the LC value they received for this MH is larger, smaller, or equal to the LC value they have locally stored at that time for this MH. If the information is new, it is propagated to all the neighbors and the local LC is updated. If the information is old, it (the LC value for this MH and its location) is updated and sent back to (only) the sender. If the information is the same, it is not propagated any further. Thus, the location counter serves to distinguish between new, old, and same information.

Given this functionality for the location counter, it is important to ensure that the counter values contained locally at a base station and the ones carried in the information messages should always monotonically increase. This property is required to enable more recent information to be associated with a larger LC value than that associated with older information. Informally, we would like to guarantee that newer information does not get overwritten by older information. When MHs move from one cell to another, they notify the new BS of their LC value. If this notified LC value is correct, then the above property can be shown to hold. Our protocol is tailored to recover from situations where this notified location counter value is incorrect. That is, the location counter value has been corrupted. Similarly, a MH could switch off and then reappear in another part of the network. As discussed earlier, we assume that when a MH switches off, it loses its memory and subsequently its location counter value. When this MH wakes up and sends *mm* to a BS, it does not have the ability to send its correct location counter value. As we will show later, this case is equivalent to the case where the location counter value is corrupted. Hence, our protocol will be shown to accommodate the joining of such MHs that switch off and then reappear elsewhere in the network in a consistent manner.

Our protocol tolerates location counter corruption by making the incorrect location counter value converge to a consistent location counter value and stabilize itself. This is accomplished as follows. Assume that a MH switches base stations (say) from BS_i to BS_j. The LD at BS_j will contain one of the following about this MH:

1. BS_j retains old information that this MH is being monitored by some BS_h. This could happen if BS_j has not yet received information that the MH moved into BS_i's cell.

2. BS_j retains the most recent information that it is being monitored by BS_i.

Consider the situation where the LC value carried by the MH gets corrupted and becomes incorrect. This incorrect LC value is sent by the MH (as part of the mmj message) to BS_j. In that case, if BS_j updates its location directory and disseminates new information on the network about the MH's arrival with an incremented incorrect LC value, it is possible that this information is deemed *old* information by other base stations on the network and ignored. It is also possible that some other base stations receive this as *same* information in case the LC value received is the same as the LC value stored for this MH in the local LD. Even though the location information is different, because the LC value is the same, this information would be ignored unless this is perceived as new information. In our protocol, we take an optimistic approach and let BS_j disseminate this information together with an incremented incorrect LC value. Here, the following situations can arise.

1. The disseminated LC value is larger than the LC value that any other BS has stored in its LD for this MH.
2. The disseminated LC value is smaller than the LC value that at least one BS has stored in its LD for this MH.
3. The disseminated LC value ties the largest of the LC values that can be found stored in some LD on the network for this MH.

If the disseminated LC value is larger than the LC value that any other BS has for this MH, then this information will be perceived as new information and the different LDs will be correctly updated. If the disseminated LC value is smaller than one other LC value on the network, eventually this larger LC will reach BS_j and at this time BS_j resolves the inconsistency by first checking to see if the MH is still in its cell (by sending a *check* message) and if so, incrementing the LC value and disseminating it back on the network. If the MH is not in its cell any more, BS_j does not have to do anything as the inconsistency will be resolved by the BS who monitors the MH at that time. In case the LC value ties the largest of the LC values that is stored in some LD, our protocol guarantees that this situation is made known to BS_j so that it can resolve this inconsistency as in the previous case.

An similar situation arises when a MH wakes up after being switched off and sends a *monitor-me* (mm) message to some base station BS_j. In this case, BS_j will disseminate *newinfo* about the location of this MH by incrementing the LC value that it stores in its LD for this MH. If there is an inconsistency, then it will be resolved exactly as described in the last paragraph. By the time the MH wakes up after being switched off, if the most recent LC value (the LC value just before it was switched off) has reached all the BSs on the network, then the incremented LC value that BS_j disseminates will be larger than the LC value contained in the LD of any other BS and hence no resolution will be necessary.

In a later section, detailed examples that illustrate these aspects of the protocol are presented.

3.1.2 Base station failures

Base station failures are handled in our protocol without any extra overhead. When a base station fails, the MHs that are being monitored by this BS will no longer receive the periodic *beacon* signal from this BS. In this case, the MHs will time out and send *mmj* messages to other BSs, if any such BSs with overlapping cells exist. If a MH is not within the cell of any BS, no monitoring connections can be established.

In the current version of the protocol we do not provide techniques for intergrating repaired base stations back into the network. In [12], we presented a distributed diagnosis algorithm that enables the monitoring and dissemination of fault status information of nodes (workstations) on a network. In this algorithm, nodes *test*, or *monitor*, one another periodically such that each fault-free node is tested by exactly one other node. When a node detects that a fault-free node it is monitoring has become faulty or that a faulty neighbor has been repaired, it propagates this new information to its fault-free neighbors, which propagate the information to their neighbors, and so on. In this way, the system overhead due to monitoring is minimized during periods when the status of the nodes in the system does not change, and new information is disseminated as quickly as possible throughout the network.

When a faulty node is repaired (becomes fault-free), it sends a *testme-join* (tmj) message to one of its neighbors, informing that node that it is joining the system and asking that node to test it. If it times-out on this request, it makes the request to another neighbor and so on until it finds a neighbor to test it. When a fault-free node gets information that its tester has failed, it realizes that it has become an *orphan*. In this case, it sends a *testme* message to one of its neighbors asking that node to test it.

This distributed fault-diagnosis algorithm could be used to diagnose base station failures and disseminate this information to other base stations. In this case, when base stations are repaired they can join the wired network and obtain consistent location information about the MHs on the network from the BS that becomes its tester on the wired network.

3.1.3 Data Structures

The data structures required are as follows.

Location directory:
A location directory (LD) is maintained at each base station in the wired network. A LD maintains information about the location of each MH in the network. In the LD at (say) BS_i, an entry is as follows: $(MH_{id}|BS_{id}|LC)$. Let us refer to the entry for the jth mobile host in the LD of BS_i by $LD_i[j]$ and to the individual fields in the entry by $LD_i[j].field$. For example, $LD_i[j].LC$ refers to the LC field. MH_{id} refers to the i.d. of the MH (note that this may not have to be explicitly kept if the i.d. can be used as an index into the array. Then, the index specifies the i.d. of the MH). BS_{id} gives the location of the MH; that is, the i.d. of the base station that is monitoring this MH as per information that is available at BS_i. If a request to connect to a MH comes into BS_i at this time (because this MH was being monitored by BS_i at an earlier time), this location information is used as a forward pointer and the request is forwarded to BS_{id} by BS_i. LC refers to the location counter value for the MH as per the information that is available at BS_i. It is possible that the above information is old and information about the MH's new location (and location counter value) has not reached BS_i yet. In this case, if a request for the MH arrives at BS_i and is forwarded to BS_{id}, as the MH has moved from BS_{id}, this request is forwarded by BS_{id} to the BS that is monitoring the MH as per the information that has reached BS_{id} so far. This way, through a chain of forwarding steps, the request eventually reaches the MH.

Messages within the wired-network:
Within the wired-network, information about the changing location of different MHs is disseminated as explained earlier. The data structure for the message that carries this information is quite similar to location directory except that no forwarding pointer information needs to be carried. This message contains entries of the form $(MH_{id}|BS_{id}|LC_c)$. There could be multiple number of entries of this form in a message. The number of entries depends on the number of MHs about which new location information needs to be propagate

Messages within the wireless network:
Within the wireless network, BSs exchange messages with MHs. An *mmj* message from a MH to a BS has the structure $(MH_{id}|LC)$. LC is the location counter (for this MH) that is maintained locally at the MH. The *mm* message only contains (MH_{id}).

3.2 Details of the Protocol

In this section, we give a specification of the location directory maintenance protocol. The specification is provided in terms of finite-state machine descriptions of the behavior of a MH. The finite-state machine descriptions are shown in Figure 1.

In these figures, ? is used to denote the message that is received, and !, to denote the message that is sent in response. A ?/! is preceded by the name of the base station (BS) that the message is received from/sent to. If multiple messages are sent in response to the reception of a message, then these are listed in the causal order in which they are sent. The messages that are received and those that are sent as responses are separated by a /. Events that are triggered locally are denoted without anything preceding the ? or !; there could be messages that are sent out in response to these locally triggered events. For example, $i?ackmmj/!connect$ refers to the situation where the MH has received a acknowledgment to its request to be monitored by base station i and in response it records that it is monitored by base station i through the internal event $connect$. A to is used to denote the time-out event that is triggered internal to a node. For example, $to(ackmmj)/j!mmj$ means that the MC has timed-out on the acknowledgment to its request to be monitored by some base station and in response requests another base station j to monitor it.

3.2.1 Messages

The messages that are sent/received between a BS and a MH are as follows:

beacon refers to periodic signals from the BSs in the network. A MH may receive beacon signal(s) from one or more BSs. Even after the MH has established a connection with a BS, it will continue to monitor the *beacon* signal from that BS. If the *beacon* signal is not received within a specified time by the MH then it will time-out, denoted by *to(beacon).*

mmj message is sent by the MH to a BS to initiate a connection between the MH and the BS.

mm is similar to the *mmj* message, but it indicates that the MH's power has been turned off and then on again.

ackmmj is an acknowledgment message that is sent by a BS in response to an *mmj* message. If a MH does not receive the *ackmmj* within a certain time-out period, it times out. This is denoted by the internal event *to(ackmmj)*. Similarly, *ackmm* is an acknowledgment message that is sent by a BS in response to a *mm* message and *to(ackmm)* is an internal event.

break refers to the message sent by the MH to break its connection with a BS. Besides breaking the connection, the *break* message also contains a forwarding pointer that serves to redirect to the new BS any messages that are received at this BS for the MH.

check refers to the message sent by a BS to check if a MH is still in its access zone. This is done when an inconsistency in location information has to be resolved by the BS as explained earlier.

ackcheck is an acknowledgment message that is sent by a MH in response to a *check* message from a BS.

listen and *connect* are internal events to a MH. *listen* refers to the fact that the MH is "listening" to the *beacon* signal from one or more BS(s). *connect* refers to the establishment of a connection between the MH and the BS.

Between the different BSs on the wired-network, information messages that propagate location information (*newinfo*, *sameinfo* and *oldinfo*) are exchanged as explained earlier.

3.2.2 States of a mobile host

A mobile host(MH) (Figure 1) can be in one of the following five states.

State 1: (*Disconnected*) The MH is not within the access zone of any BS. (i.e., it is not being monitored by any BS.)

State 2: (*Wait*) The state of MH will change from 1 to 2 when it receives a *beacon* signal from a BS. It will then send an *mmj* message to the BS, and wait for the *ackmmj* message from the BS.

State 3: (*Connected*) Having received the *ackmmj* message, the MH will be able to connect to other mobile hosts through this BS. (i.e., this BS becomes a gateway of the MH.)

State 4: (*Reconnecting*) When the MH decides to send an *mmj* message to another BS while there is still a connection with the previous BS, it *will not* break the connection with the previous BS unless it has received the *acktmj* message from the new BS. After establishing a new connection, it will then break the old connection with the previous BS and the MH will go back to state 3.

State 5: (*Awake*) The power at a MH could go off at any time. When the MH is powered on again, the MH will be in the *Awake* state. The behavior of the MH in this state will be similar to its behavior in State 1. It will listen to *beacon* signal(s) and try to form a new connection with a base station. However, it will send a $monitor - me$ (*mm*) message instead of an *mmj* message to the base station to let the base station know that it wants to join the network after its power has been turned off and on again.

3.2.3 Mobile host state transitions

Consider the transitions in the finite-state machine diagram shown in Figure 1.

1. Transitions out of state 1 *(Disconnected)*.

   Transition to state 2*:* BS_i*?beacon/*BS_i*!mmj*: The MH has received a *beacon* signal from one of the base stations (BS_i), and it does not have a connection with any BS. It then sends i an *mmj* message and waits for the *ackmmj* message back from BS_i.

   Transition back to state 1*: ?listen/null*: The MH continues to listen for beacon signals but cannot hear any.

2. Transitions out of state 2 *(Wait)*.

   Transitions to state 3*:* BS_i*?ackmmj/!connect* or BS_i*?ackmm/!connect*: The MH receives a *ackmmj* or *ackmm* message from a base station (BS_i). Therefore, the MH has formed a connection with BS_i. *connect* is an internal event.

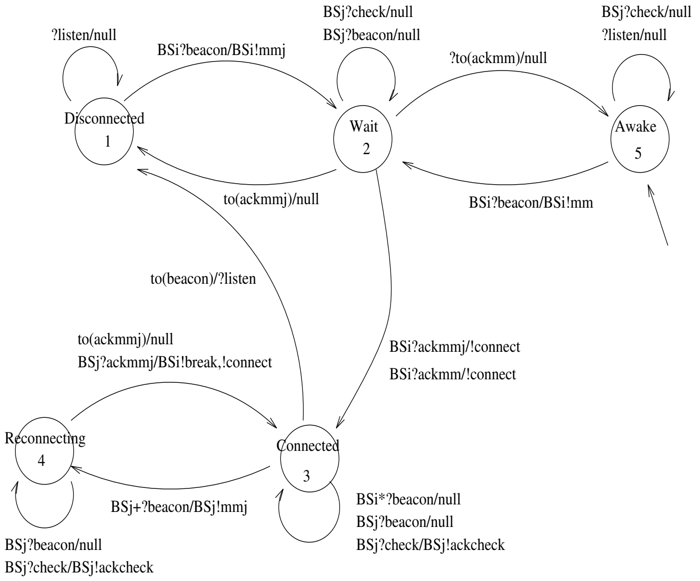
?listen/null
BSi?beacon/BSi!mmj
BSj?check/null
BSj?beacon/null
?to(ackmm)/null
BSj?check/null
?listen/null
Disconnected
1
Wait
2
Awake
5
to(ackmmj)/null
BSi?beacon/BSi!mm
to(beacon)/?listen
to(ackmmj)/null
BSj?ackmmj/BSi!break,!connect
BSi?ackmmj/!connect
BSi?ackmm/!connect
Reconnecting
4
Connected
3
BSj+?beacon/BSj!mmj
BSi*?beacon/null
BSj?beacon/null
BSj?check/BSj!ackcheck
BSj?beacon/null
BSj?check/BSj!ackcheck

Figure 1: Mobile host state transitions

Transition to state 1*: to(ackmmj)/null*: The MH times out on the acknowledgment to an mmj message that it sent to a base station. It then goes back to the Disconnected state and waits for another beacon signal.

Transition to state 5: to(ackmm)/null: The MH times out on the acknowledgment to a mm message that it sent to a base station. It then goes back to the Awake state and waits for another beacon signal.

Transitions back to state 2*:* BS_j*?beacon/null*: The MH has received a *beacon* signal from BS_j, and it is waiting for a connection with another BS. It then simply ignores the *beacon* signal from BS_j. BS_j*?check/null*: The MH receives a *check* message from a BS but now it is waiting to join another BS after it just woke up or after being disconnected. Therefore, it does not respond to this check.

3. Transitions out of state 3 *(Connected)*.

   Transition to state 4*:* BS_{j+}*?beacon/j!mmj*: The MH has received a *beacon* signal from one of the BS (BS_{j+}). Even though it realizes that it has a connection with another BS, it may decide to form a stronger connection with BS_{j+}. It then will send an *mmj* message to BS_{j+}.

   Transition to state 1*: to(beacon)/!listen*: The MH has timed-out on all beacon signals. That means, it does not hear any beacon signal. In this case, it knows that it has been disconnected from the network and continues to listen for some beacon signal.

   Transitions back to state 3*:* BS_{i*}*?beacon/null*: The MH receives a beacon signal from the BS that it is already connected to. BS_{i*} refers to this BS. The MH will ignore this signal. However, the MH will time out if the *beacon* signal from BS_{i*} is not received within a certain time-out period. BS_j*?beacon/null*: The MH receives a beacon signal from any BS other than the one that it is already connected to. The MH will ignore this signal. BS_i*?check/*BS_i*!ackcheck*: The MH receives a *check* message from the base station that is monitoring it and acknowledges this check message.

4. Transitions out of state 4 *(Reconnecting)*.

   Transitions to state 3*:* BS_j*?ackmmj/i!break; !connect*: The MH has received a *ackmmj* message from BS_j in response to an *mmj*. It will then send a *break* message to its present monitor BS_i to break the connection. At the same time, it forms a connection with BS_j. The *connect* is an internal event. *to(ackmmj)/null*: The MH times out on the acknowledgment to its request to be

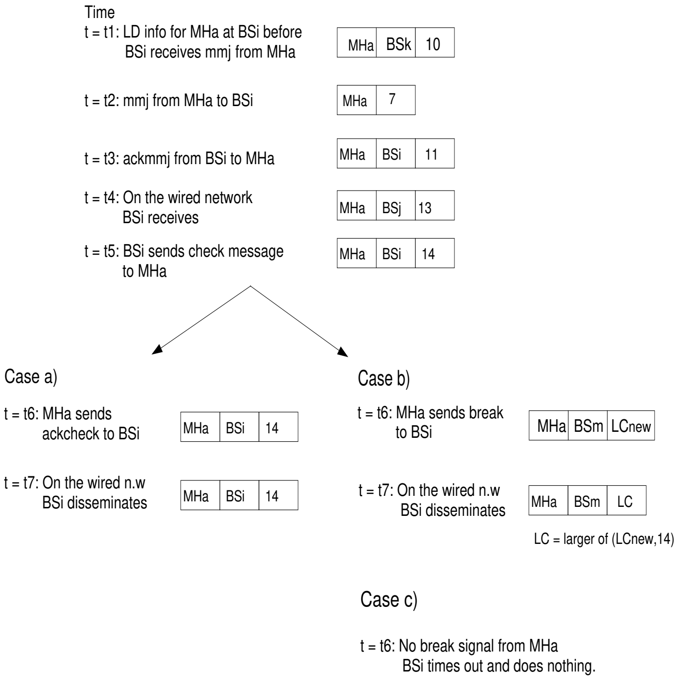
Time
t = t1: LD info for MHa at BSi before BSi receives mmj from MHa
MHa | BSk | 10
t = t2: mmj from MHa to BSi
MHa | 7
t = t3: ackmmj from BSi to MHa
MHa | BSi | 11
t = t4: On the wired network BSi receives
MHa | BSj | 13
t = t5: BSi sends check message to MHa
MHa | BSi | 14
Case a)
t = t6: MHa sends ackcheck to BSi
MHa | BSi | 14
t = t7: On the wired n.w BSi disseminates
MHa | BSi | 14
Case b)
t = t6: MHa sends break to BSi
MHa | BSm | LCnew
t = t7: On the wired n.w BSi disseminates
MHa | BSm | LC
LC = larger of (LCnew,14)
Case c)
t = t6: No break signal from MHa BSi times out and does nothing.

Figure 2: Sequence of messages exchanged in $scenario$ 1

monitored by BS_{j+}, the BS with a stronger beacon signal. In this case, it goes back to state 3 and may try to connect to another BS with a stronger beacon signal later.

Transitions back to state 4*:* *j?beacon/null*: The MH ignores any beacon signal that it receives when in state 4. BS_i*?check/*BS_i*!ackcheck*: The MH receives a *check* message from the base station that is monitoring it and acknowledges this check message.

5. Transitions out of state 5 *(Awake)*.

   Transition to state 2*:* BS_i*?beacon/*BS_i*!mm*: The MH receives a *beacon* signal from a base station (BS_i). It has been switched on after being switched off and hence does not have a connection to any BS. Therefore, it sends BS_i a *mm* message and waits for the *ackmm* message back from BS_i.

   Transitions back to state 5*:* *?listen/null*: The MH has just been turned on after being turned off, listens for beacon signals but cannot hear any. BS_j*?check/null*: The MH receives a *check* message from a BS but now it is waiting to join another BS after being switched off and on again. So, it does not respond to this check.

3.3 Protocol Execution Examples

The location counter value of a mobile host could be incorrect (does not follow the correct sequence) due to the following reasons. **(a).** The LC value stored at the MH is corrupted due to a hardware failure. **(b).** MH has just awakened from the disconnected state and does not retain memory of the LC value it stored before it was switched off.

Both of the above situations are very similar as far as the fault tolerance aspects of the protocol are concerned. In both cases, the MH can send to the BS that it wishes to join a LC value which is smaller, equal, or larger than the LC value that the BS has stored in its LD for that MH.

Let us consider two scenarios that illustrate how the protocol tolerates this type of failure.

3.3.1 Scenario 1

Figure 2 shows the sequence of messages exchanged for this scenario. In this scenario, at time t_1, the location directory at base station (BS_i) contains information that mobile host (MH_a) is monitored by base station BS_k and that the LC value is 10. At time t_2 assume that MH_a receives a stronger beacon signal from BS_i, decides to join BS_i, and sends a corrupted LC value in its *mmj* message to BS_i. If the LC value received by BS_i from MH_a is smaller than what BS_i has in its LD for MH_a, then BS_i will increment the LC value in its LD and send it to MH_a as part of the *ackmmj* message (at time t_3). Also, BS_i will disseminate the new location information of MH_a along with the new LC value to the rest of the base stations. Once MH_a receives the *ackmmj* message, it will update the local LC value with the LC value that was received as part of the *ackmmj* message. However, the new LC value that BS_i has stored in its LD for MH_a may be smaller than what some other base station(s) (say BS_j) has stored in its LD. This will happen, for example, if the information disseminated by BS_j, to which MH_a had previously been connected, is yet to reach BS_i. At time t_4, BS_i receives BS_j's LD information for MH_a that contains a LC value larger than what BS_i had sent out. In this case, there are three possible scenarios that could occur.

(a). MH_a is still within BS_i's cell and is connected to it. Once BS_i receives the LC value from BS_j, it will increment that LC value and send a *check* message (at time t_5) along with the new LC value to MH_a to determine whether MH_a is still connected to it. Since MH_a has neither moved out of BS_i's cell nor established connection with any other base station, MH_a sends an *ackcheck* message to BS_i (time t_6). Upon receiving the *ackcheck* message from MH_a (time t_7), BS_i will update its location directory with the new LC value (i.e., 14) and disseminate the location information of MH_a on the wired network. This process will be repeated until one of the following events happens. a) there are no base stations with LC value for MH_a strictly larger than what BS_i has stored in its LD, or b) MH_a has moved out of BS_i's region or established a connection with some other base station. In the first case, all base stations will eventually receive the location information of MH_a from BS_i. The latter case is analyzed next.

(b). MH_a has moved to another cell, and had sent a break signal (time t_6) to BS_i before moving out. Let us assume that MH_a has moved into the cell of BS_m and has established a connection with BS_m. The new LC value LC_{new}, which came along with the *break* signal, depends on what LC value BS_m had stored in its location directory for MH_a. If LC_{new} is larger than the LC value at BS_i, then BS_i updates its LD with $(MH_a|LC_{new}|BS_m)$. If LC_{new} is smaller, then BS_i eventually updates its LD with the LC value that arrives on the wired network from BS_m. In the latter case, it is BS_m's responsibility to resolve the correct LC value for MH_a and disseminate this information. In both cases, BS_i disseminates the LD information it has stored in its LD for MH_a on the wired network.

(c). MH_a has moved into the cell of some other base station and has established connection with that base station without sending a break signal to BS_i. In this case, once BS_i receives the LC value from BS_j, it will check to see whether MH_a is still in its region. Since MH_a has moved out, there will be no *ackcheck* message from MH_a. Eventually, BS_i times-out and does nothing. However, BS_i propagates the information it received on the wired network to other base stations to make sure that the new base station, to which MH_a has moved to, is aware of the fact that there is a base station with a larger LC value for MH_a.

3.3.2 Scenario 2

Consider another scenario where MH_a establishes connection with base station BS_i and its new LC value is updated to be 11. At time t_4 (see Figure 3) BS_i receives location information about MH_a disseminated by some other base station which indicates that MH_a is connected to BS_j with LC value 11. Eventually, BS_j will receive information originated by BS_i which indicates that MH_a is connected to BS_i with LC value 11. Now, both BS_i and BS_j have the same LC value for MH_a, but different location information. Both base stations will increment this tied LC value and send a *check* message (time t_5) to MH_a. All the base stations (including BS_i and BS_j) that received this tied LD information for MH_a will update their LD for MH_a with $(MH_a|BS_i|11|BS_j|11)$ to indicate that there are two base stations with the same LC value that claim to have a connection with MH_a. This tie situation can only be resolved by the base station that MH_a is currently connected to. Again, there are three possible scenarios that could occur.

(a). MH_a is still within BS_i's cell and has not established a connection with any other base station. Since, MH_a is still connected to BS_i, it sends a *ackcheck* message (at time t_6) to BS_i. Upon receiving the *ackcheck* message, (at time t_7) BS_i will remove BS_j's name from the LD entry for MH_a and disseminate the new location information on the wired network.

Since, MH_a is not currently connected to BS_j, BS_j will time out on its *check* message. (Note: MH_a may have moved back in, but in that case, MH_a would send an *mmj* message first). Now, the location directory information for MH_a at BS_j will have both BS_i and BS_j in the base station

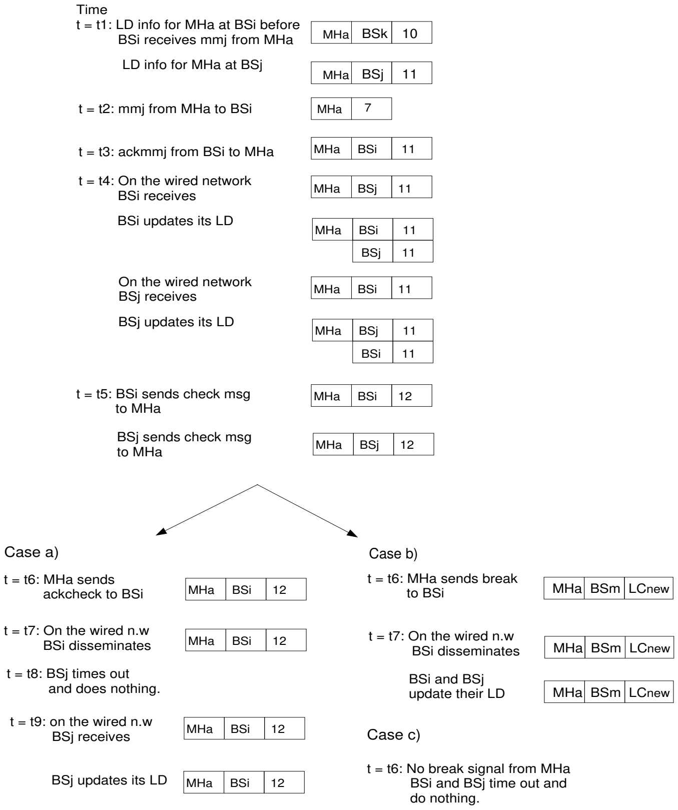
Time
t = t1: LD info for MHa at BSi before BSi receives mmj from MHa
MHa | BSk | 10
LD info for MHa at BSj
MHa | BSj | 11
t = t2: mmj from MHa to BSi
MHa | 7
t = t3: ackmmj from BSi to MHa
MHa | BSi | 11
t = t4: On the wired network BSi receives
MHa | BSj | 11
BSi updates its LD
MHa | BSi | 11
BSj | 11
On the wired network BSj receives
MHa | BSi | 11
BSj updates its LD
MHa | BSj | 11
BSi | 11
t = t5: BSi sends check msg to MHa
MHa | BSi | 12
BSj sends check msg to MHa
MHa | BSj | 12
Case a)
t = t6: MHa sends ackcheck to BSi
MHa | BSi | 12
t = t7: On the wired n.w BSi disseminates
MHa | BSi | 12
t = t8: BSj times out and does nothing.
t = t9: on the wired n.w BSj receives
MHa | BSi | 12
BSj updates its LD
MHa | BSi | 12
Case b)
t = t6: MHa sends break to BSi
MHa | BSm | LCnew
t = t7: On the wired n.w BSi disseminates
MHa | BSm | LCnew
BSi and BSj update their LD
MHa | BSm | LCnew
Case c)
t = t6: No break signal from MHa BSi and BSj time out and do nothing.

Figure 3: Sequence of messages exchanged in $scenario\,2$

field. Since, BS_j cannot resolve the tie situation, it will propagate the location directory information that arrived on the wired network (i.e., $(MH_a|BS_i|11)$) to other base stations while leaving the entry $(MH_a|BS_i|11|BS_j|11)$ in its LD.

All other base stations except BS_i and BS_j, will just propagate the information that came on the wired network since they are not currently connected to MH_a. But, they will update their own location directory with $(MH_a|BS_i|11|BS_j|11)$. If they receive the same information again (i.e., message saying MH_a is with either BS_i or BS_j with the same LC value) they will ignore the information. Once BS_i resolves the tie, all other base stations (including BS_j) will remove BS_j from their LD entry for MH_a, and update the LC value.

(b). MH_a has moved out of BS_i's cell and had sent a break signal before leaving. Since the LC value of MH_a before moving out of BS_i's cell is equal to the LC value that BS_i has received on the wired network, the new LC value that came along with the *break* signal is strictly larger than what BS_i received. Hence, BS_i updates its LD entry for MH_a with $(MH_a|BS_m|LCnew)$. Then BS_i disseminates the new location of MH_a on the wired network. When BS_j receives this new information, it will also update its location directory.

(c). MH_a leaves BS_i without sending a break signal. In this case, both BS_i and BS_j time out on their *check* messages to MH_a. But, both of them propagate the LD information that came on the wired network. They will update the base station field for MH_a with $(MH_a|BS_i|11|BS_j|11)$. Once they receive the LD information from the new base station that MH_a is connected to, BS_i and BS_j will update their LDs. All other base stations do likewise. The new LC value is strictly larger than what BS_i and BS_j had stored earlier in their LD entry for MH_a.

4 Correctness

The correctness of the protocol can be shown at two different levels. At the coarse level, we consider the system of mobile hosts after the *last movement* has occurred and then show that after this last movement, *eventually* all the fault-free base stations will have the same location counter value, and will have the same and correct location information for all the active mobile hosts (as mentioned earlier, we do not make a distinction between the active and stand-by modes of operation of a mobile host). We need to show that this property holds in spite of location counter values getting corrupted and also that this property holds for all the mobile hosts including the ones that might have switched off and on again. The occurrence of *last movement* refers to the last time any mobile host travels from one cell to another and changes its monitor. Using this approach we can show that the desired properties for the protocol holds after sufficient time has elapsed since this last movement. This *eventual correctness* approach is useful in showing that after all movement of all the mobile hosts has stopped, eventually, location information that is in transit will reach all the base stations and consistent location information about all the mobile hosts will be known to all the base stations. In fact, the idea of *last movement* can be individually applied to each of the MHs separately to show that eventually, all the base stations will have the correct location information for each of the MHs on the network.

Of course, at a finer level, we need to show that the base stations never make *backward progress* in learning information about the location of the mobile hosts. For this, we need to show that base stations will never overwrite newer information in their LDs with older information that they may receive through the information messages on the wired network. Secondly, we need to show that the base stations always make *forward progress* in learning newer information about the location of the mobile clients. In order to show this, we can appeal once again to the notion of *eventual correctness*, as eventual correctness implies forward progress.

Further, to show that there is no livelock due to which wasteful messages could be sent, we need to show that after the *last movement*, eventually no messages that are part of the protocol specification are exchanged on the wired -network and only the periodic *beacon* signals are sent over the wireless channels.

In the following, we state the above properties in terms of Lemmas and Theorems. Due to space considerations, we do not provide the proofs. Further details can be found in [13].

Let τ be the time at which the last movement in the system has occurred. Let B be the set of fault-free base stations on the wired-network at some time $t > \tau$. Let N specify the total number of MHs that are active at some time $t > \tau$ on the network and let the location of a mobile host MH_i after the last movement be denoted by LM_i.

The following three lemmas and theorem show that after the last movement, eventually all the fault-free base stations will have the same location counter value, and will have the correct location information for each of the active mobile hosts in the network.

Lemma 1: At some time $t > \tau$, all the fault-free base stations will have the *same* location counter values for each of the active mobile hosts. That is, $\forall\, BS_i, BS_k \in B, \forall\, j, 1 \leq j \leq N, LD_i[j].LC = LD_k[j].LC$.

Lemma 2: At some time $t > \tau$, all the fault-free base stations will have the *same* location information for each of the active mobile hosts. That is, $\forall\, BS_i, BS_k \in B, \forall\, j, 1 \leq j \leq N, LD_i[j].BS_{id} = LD_k[j].BS_{id}$.

Lemma 3: At some time $t > \tau$, any fault-free base station BS_l which becomes the monitor for any active mobile host MH_i after this mobile host's last movement, will have the *correct* location information for MH_i in its LD. That is, $LD_l[i].BS_{id} = LM_i = BS_l$.

Lemmas 1,2 and 3 lead to the following theorem:

Theorem 1: At some time $t > \tau$, all the fault-free base stations will have the same location counter values and *correct* location information for each of the active mobile hosts. That is, $\forall\, BS_i \in B, \forall\, j, 1 \leq j \leq N, LD_i[j].BS_{id} = LM_j$

At a finer level, we show that base stations do not make backward progress in learning location information. That is they do not overwrite newer location information for a MH with older location information.

Lemma 4: Consider a mobile host MH_i which is at a fault-free base station BS_j at time t_1 and at a fault-free base station BS_k at time t_2 where $t_2 > t_1$. Assume that for all time $t > t_2$, the mobile host MH_i never visits BS_j again. Consider the location directory at another fault-free base station BS_l. At time $t_3 > t_1$ let $LD_l[i].BS_{id} = BS_j$ and at time $t_4 > t_3$, let $LD_l[i].BS_{id} = BS_k$. Then, $\forall\, i$ and $\forall\, l$, there does not exist a time $t > t_4$ such that at time t, $LD_l[i].BS_{id} = BS_j$.

Lemma 4 and Thoerem 1 provide the following theorem.

Theorem 2: The base stations never make backward progress and always make forward progress in learning about the changing location information of all the mobile hosts.

It is also shown that once all the fault-free base stations have the most up-to-date location information about all the active mobile hosts, no further messages related to the location directory maintenance protocol are disseminated other than the periodic beacon signals.

Theorem 3: For all $t \geq t'$ where t' is some time $> \tau$, no protocol messages are exchanged on the wired network and only *beacon* signals are sent on the wireless channels.

5 Conclusions

We have presented a fault-tolerant protocol for maintaining location directory information in mobile networks. The protocol tolerates base station failures and also allows for mobile hosts that switch off and arbitrarily reappear in some other part of the network. The protocol also tolerates the corruption of a logical time stamp that is part of any protocol where new location information has to be distinguished from old location information when a location directory is updated. It is shown that the protocol maintains consistent location information and does not overwrite new location information with old location information.

6 Bibliography

[1] B. Awerbuch and D. Peleg, "Concurrent on-line tracking of mobile users", *Proc. ACM SIGCOMM Symposium on Communications, Architectures and Protocols*, October 1991.

[2] B.R. Badrinath et. al,, "Locating strategies for personal communication networks", *Proc. of the IEEE Globecom Workshop on networking of Personal Communication*, December 1993.

[3] A. Bar-Noy and I. Kessler, "Tracking mobile users in wireless networks", *IEEE Transactions on Information Theory*, 1993.

[4] P. Bhagwat and C. Perkins, "A mobile networking system based on Internet Protocol (IP)", in *Usenix Symposium on Mobile and Location-Independent Computing*, August 1993.

[5] D.J. Goodman, "Cellular packet communications", *IEEE Transactions on Communications*, vol. 38, pp. 1272–1280, August 1990.

[6] S.J. Mullender and B.M. Vitanyi, "Distributed Match-Making", *Algorithmica*, vol. 3, pp. 367–391, 1988.

[7] J. Ioannidis, D. Duchamp and G.Q. Maguire, "IP-based protocols for mobile internetworking, *Proceedings of ACM SIGCOMM*, pp. 235–245, 1991.

[8] J. Ioannidis et. al, "The design and implementation of a mobile internetworking", *Proc. of Winter USENIX*, January 1993.

[9] K. Keeton et. al, "Providing connection-oriented network services to mobile hosts", *Proc. of the USENIX Symposium on Mobile and Location-Independent Computing*, Cambridge, MA, August 1993.

[10] P. Krishna et. al, "Location Management in Distributed Mobile Environments", *Proc. of the Third International Conference on Parallel and Distributed Information Systems*, September 1994.

[11] C. Perkins, "Providing continuous network access to mobile hosts using TCP/IP", *Joint European Networking Conference*, May 1993.

[12] S. Rangarajan et. al, "A Distributed diagnosis algorithm for arbitrary network topologies", *IEEE Transactions on Computers*, Special Issue on Fault-Tolerant Computing, vol. 44, no. 2, pp. 312–334, February 1995.

[13] S. Rangarajan et. al, "A Fault-Tolerant Protocol for Location Directory Maintenance in Mobile Networks", Technical Report, Department of ECE, Northeastern University, December 1994.

[14] R. Steele, "Deploying personal communication networks", *IEEE Communications Magazine*, vol. x, pp. 12–15, September 1990.

[15] S.F. Wu and C. Perkins, "Caching location data in mobile networking", *IEEE Workshop on Advances in Parallel and Distributed Systems*, October 1993.