Report Number: WUCS-98-02

1998-01-01

Algorithms for Message Delivery in a Micromobility Environment

Amy L. Murphy, Gruia-Catalin Roman, and George Varghese

As computing components get smaller and people become accustomed to having computational power at their disposal at any time, mobile computing is developing as an important research area. One of the fundamental problems in mobility is maintaining connectivity through message passing as the user moves through the network. This is usually accomplished in one of two ways: search or tracking. In search, an algorithm hunts the mobile unit through the network each time a message is to be delivered, while in tracking, a specific home keeps up to date information about the current location of the mobile unit. Our... **Read complete abstract on page 2.**

Algorithms for Message Delivery in a Micromobility Environment

Amy L. Murphy, Gruia-Catalin Roman, and George Varghese

Complete Abstract:

As computing components get smaller and people become accustomed to having computational power at their disposal at any time, mobile computing is developing as an important research area. One of the fundamental problems in mobility is maintaining connectivity through message passing as the user moves through the network. This is usually accomplished in one of two ways: search or tracking. In search, an algorithm hunts the mobile unit through the network each time a message is to be delivered, while in tracking, a specific home keeps up to date information about the current location of the mobile unit. Our paper proposes two message delivery algorithms based on these two paradigms of mobility. In general, our approach is to adopt existing algorithms from distributed computing to solve the problem of message delivery in the mobile setting, allowing us to leverage off existing knowledge about these algorithms and extensive research from distributed computing. The transformation from distributed to mobile computing is accomplished by treating the mobile units as messages that roam across the fixed network structure. First we show how snapshot algorithms can be adapted to perform message delivery through search, and then how the model of diffusing computations can be altered to track a mobile unit.

Algorithms for Message Delivery in a Micromobility Environment

Amy L. Murphy
Gruia-Catalin Roman
George Varghese

WUCS-98-02

February 24, 1998

Department of Computer Science
Washington University
Campus Box 1045
One Brookings Drive
Saint Louis, MO 63130-4899

Algorithms for Message Delivery in a Micromobility Environment

Amy L. Murphy
Gruia-Catalin Roman
and
George Varghese
Washington University, St. Louis
alm, roman, varghese@cs.wustl.edu

As computing components get smaller and people become accustomed to having computational power at their disposal at any time, mobile computing is developing as an important research area. One of the fundamental problems in mobility is maintaining connectivity through message passing as the user moves through the network. This is usually accomplished in one of two ways: search or tracking. In search, an algorithm hunts the mobile unit through the network each time a message is to be delivered, while in tracking, a specific home keeps up to date information about the current location of the mobile unit. Our paper proposes two message delivery algorithms based on these two paradigms of mobility. In general, our approach is to adopt existing algorithms from distributed computing to solve the problem of message delivery in the mobile setting, allowing us to leverage off existing knowledge about these algorithms and extensive research from distributed computing. The transformation from distributed to mobile computing is accomplished by treating the mobile units as messages that roam across the fixed network structure. First we show how snapshot algorithms can be adapted to perform message delivery through search, and then how the model of diffusing computations can be altered to track a mobile unit.

Categories and Subject Descriptors: []:

General Terms: Message Delivery, Mobility

1. INTRODUCTION

Mobile computing reflects a prevailing societal and technological trend towards ubiquitous access to computational and communication resources. Wireless technology and the decreasing size of computer components allow users to travel within the office building, from office to home, and around the country with the computer at their side. Both location-transparent and context-dependent services are desired. Decoupled computing is becoming the norm [Kistler and Satyanarayanan 1992]. Disconnection is no longer a network fault but a common event intentionally caused by the user in order to conserve power or a consequence of movement. Tethered connectivity is making way to opportunistic transient connections via radio or infrared transmitters.

As this new world of computing is taking form, many fundamental assumptions about the structure and the behavior of computer networks are being challenged and redefined. Mobile computing is emerging as a novel paradigm with its own characteristic problems, models, and algorithms. This results in at least two kinds of research questions. First, how are particular tasks (e.g., maintaining file consistency) solved in a mobile setting. Second, what is the precise relation (e.g., similarities and differences) between mobile computing and traditional distributed computing.

Our paper attempts to make a contribution to both kinds of questions. On the algorithmic side, we describe new algorithms for sending messages to mobile units. On the modeling side, we describe a simple approach to modeling mobile units that has considerable similarity to the standard distributed computing model. This in turn allows us to transfer useful results from classical distributed computing to the new mobile setting.

Distributed versus Mobile Computing. The typical model of distributed computing treats a network as a graph in which vertices represent processing nodes and edges denote communication channels. Faults may render parts of the network inoperational either temporarily or permanently. Despite faults, the overall structure is considered to be static. One way to introduce mobility in this context is to treat the nodes as mobile support centers coordinating multiple radio base stations. Mobile units are allowed only to connect and disconnect from mobile support centers through communication with the base stations. The result is a fixed core of static nodes and a fluid fringe consisting of mobile units. The obvious connection to traditional distributed computing and an extensive investment in current network technologies helped this model become dominant in mobile computing today [Badrinath et al. 1994; Johnson 1996]. Consequently, much effort is being expended on providing Internet connectivity to mobile units. Nevertheless, other models are being investigated as well. Ad-hoc network is a term that came to denote transitory associations of mobile units which do not depend upon any fixed support infrastructure. One can visualize an ad-hoc network as a continuously changing graph. Connection and disconnection is controlled by the distance among units and by their willingness to collaborate in the formation of a cohesive, albeit transitory, community. The density of mobile units may provide an additional discriminating feature of these kinds of networks. Participants at a conference and disaster relief workers may find it necessary to interact with each other in this manner when the static support infrastructure is not available. One can imagine many other practical applications as well.

While mobility demands a novel perspective on distributed computing, it is equally imperative to investigate any essential features of mobility that are already present in our current view of distributed computing. In fact, the term "mobility" has been used already to refer to processes that migrate across the network [Gray et al. 1996; Ranganathan et al. 1997]. We suggest yet another way of thinking about mobility in the context of the traditional fixed graph structure. The basic idea is to treat mobile units as roving messages which preserve their identity as they travel across the network. Many practical applications may be suitable for this kind of modeling. A cellular phone, for instance, travels from one cell to the next. While operating inside one cell, the phone may be viewed as residing at a node inside the support network; similarly, the handoff protocol (triggered by the detection of signal degradation) may be modeled as the traversal of a channel between two nodes representing the individual cells. Voice transmissions among two phones are also modeled as messages.

Our interest in this model rests with *its ability to facilitate the application of established distributed algorithms to problems in mobile computing.* To illustrate this point, this paper shows how algorithms designed to compute distributed global snapshots provide reasonable solutions for the problem of multicasting messages among mobile units and how algorithms used to detect termination of diffusing computations provide reasonable solutions for tracking mobile unit movement.

Prior Work on Message Delivery. While isolated mobile units are useful, most users require the mobile unit to communicate with others in terms of voice (e.g., cellular phones) and data (e.g., email) delivery. Thus a fundamental problem in mobile computing is the delivery of a message from a source to a mobile unit. This allows the mobile unit to continue communicating and be seamlessly part of an ongoing distributed computation.

Differences in spatial and time granularity of movement have lead to two approaches to delivery. The literature has referred to these styles as micromobility and macromobility. In macromobility, the user moves slowly among large networks. This is analogous to a user taking their laptop computer from a local network at work to a local network at home. It is assumed that this movement is relatively infrequent, and often involves complete disconnection from network resources for a length of time. Micromobility models the movement of a user within a single network, or a tightly clustered group of networks. This is similar to the mobile telephone model in which a user participating in a phone conversation moves among support centers without losing the connection. In micromobility, the user is allowed to move any number of times, but the rate of movement must be slow enough to allow for the handoff to complete. Otherwise, the telephone conversation will be lost.

While the *unicast* problem of delivering a message to a *single* recipient is important, in recent years the *multicast* problem of delivering a message to *multiple* recipients has also become crucial. Multicast support through the MBONE has become a standard part of the Internet [Eriksson 1994] and is finding wide use for conferencing (e.g., tools like VIC and VAT [McCanne and Jacobson 1995; Jacobson and McCanne]) and video distribution. In the context of Mobile IP and macromobility, two possible approaches are available. The mobile unit can register to receive multicast packets through its home agent, in which case, the home agent must encapsulate every multicast message and unicast it to the foreign location of the mobile unit. Alternately, the mobile unit can join the multicast group at a local multicast router on the subnet it is visiting. This assumes the local subnet supports multicast. Because many multicast protocols rely on the address of the sender to properly forward messages along a multicast tree, corresponding changes must be made in order for the mobile unit to send to a multicast group address

Standard solutions to unicast message delivery (e.g., cellular phones [Ioannidis and Maguire 1993] and Mobile IP [Perkins 1996]) rely on *tracking* a mobile unit as it moves around a fixed wired network. For example, in cellular systems as a phone P moves from cell C to adjoining cell $C\prime$, the phone detects a stronger signal from $C\prime$ and requests a handoff from C to $C\prime$. After the handoff, $C\prime$ is now responsible for forwarding voice packets to the phone, and the cellular system keeps track of this association between $C\prime$ and P. Similarly in Mobile IP, this is accomplished by the mobile unit registering its new location with a home agent, and having the home agent forward any messages for that mobile unit to the new location. Although Mobile IP is well accepted among the proponents of the Internet, partially due to its incremental deployment, other approaches propose changing the routers to adapt to the movement of the mobile units and effectively intercept packets en route to the home agent and direct them toward the mobile unit itself. Because these approaches involve fundamental changes to the routers, this is more popular as a solution within the corporations developing the technology.

One disadvantage to tracking surfaces if the mobile unit moves rapidly. The tracking system must constantly send updates even if the mobile unit does not receive any messages during this period. The frequency and overhead of tracking information dissemination scales poorly with the speed of movement. However, in a system with moderate amounts of, or slow movement, tracking involves less overhead in message delivery than broadcast search mechanisms.

In search solutions, the location of the mobile unit is not kept anywhere in the system. Therefore, in order to get a message to the mobile unit, the sender must either broadcast a search request to locate the mobile unit, then forward the message along the same path that the search message took to that location, or the sender can simply broadcast a copy of the message. The first mechanism has been suggested for ad hoc mobility where there is no infrastructure to route packets along. This approach takes advantage of the natural broadcasting nature of mobile communication such as radio to execute a broadcasting protocol to any neighboring mobile units within range. This kind of route discovery is useful in moderately changing environments where a route to a mobile unit is viable long enough for both route discovery and message delivery. Clearly, searching the entire Internet for a mobile unit appears ludicrous. However, the Internet is divided into a number of hierarchical domains and subnets and the intent is to perform a form of broadcast search within a small local domain. For example, a cable office may wish to multicast messages to a number of cable servicemen that are roaming in the metro St. Louis area.

Both search and tracking have merits based on expected modes of mobility, therefore in this paper we address each in turn, adapting algorithms from distributed computing to perform message delivery. For search, we adapt the snapshot algorithm, specifically one presented by Chandy and Lamport. For tracking, we use the diffusing computation model presented by Dijkstra and Scholten to maintain a route to the mobile unit. By doing this, we bring together the two concerns of the paper: applying techniques from distributed algorithms to mobile computing, and the problem of message delivery. Section 2 defines the problem we intend to solve, namely message delivery in a mobile setting. In Section 3 we explore the use of snapshot algorithms as a search mechanism for message delivery, present the motivation, algorithm properties, and possible extensions. We also consider adaptations that make the approach viable in a model similar to the cellular telephone system. Section 4 provides similar details for a tracking strategy based on diffusing computations. Finally, Section 5 brings these results together and concludes the paper.

2. PROBLEM DEFINITION: MESSAGE DELIVERY

The problem we are interested in is the delivery of messages among pairs of mobile units. In our abstract model, both the mobile units and the messages are represented as messages. A mobile unit can send and receive messages only when it is present at some node in the fixed network, a situation that models the existence of an established connection between a mobile unit and a support center. Whenever a mobile unit is on a channel, it may be viewed as being temporarily disconnected from the network and, therefore, unable to communicate. Since we no longer differentiate between communication lines and physical movement, it is reasonable to ask what happens when messages and mobile units are found on the same channel. We make the assumption that all channels preserve message ordering, i.e., they are FIFO channels. This appears to require that mobile units travel through space and reconnect to the next support center faster than messages can be transmitted across the network. The FIFO behavior, however, can be realized by integrating the handoff protocol with message passing. The details of this will be expanded in a later section. The only other assumption we make about the behavior of mobile units is that a node can deliver any messages before the mobile unit moves on, i.e., movement is slower than communication. Finally, we assume that the network is fully connected, i.e., there is always a way to deliver the message to its destination agent.

The message delivery problem can now be formulated as follows: Given a fully connected network with FIFO channels and guaranteed message delivery, a message located at one of its nodes, and a mobile unit for which the message is destined, develop a distributed algorithm that guarantees single delivery of the message, and leaves no trace of the message, at either a node or a mobile unit, within a bounded time after delivery. Minimizing storage requirements across the network is another concern.

Because mobile units do not communicate directly with one another, the network must provide a mechanism to transmit the messages. The original message is assumed to be in the local memory of some processing node, presumably left there by some mobile unit which is the source of the message. Since a mobile unit is not required to visit all nodes to gather its messages, the message cannot remain isolated at the node on which it is dropped off, but instead must be distributed through the network. The specifics of this distribution mechanism are left to the algorithm.

3. BROADCAST SEARCH

First we explore broadcast search mechanisms for message delivery to rapidly moving mobile units. Broadcast search is not quite trivial because the mobile unit may move from node to node, like the Artful Dodger, one step ahead of the broadcast. The problem can be solved if nodes keep copies of the message for an indefinite period. However, Internet routers do not have the storage or the intention of storing application messages indefinitely. Messages must be garbage collected quickly if the scheme is to have any chance of being practical. Our solution has the attractive property of guaranteeing delivery exactly once (in practice, with unreliable links, delivery would be at most once) while allowing rapid garbage collection in time proportional to a round trip delay on a single link.

3.1 Motivation

The problem we are interested in is the delivery of messages to a mobile unit in a network of fixed mobile support centers (MSCs) and radio base stations (RBSs). For simplicity, we will assume each MSC controls only one RBS, and all MSCs whose radio base stations are neighbors have a fixed communication channel between them.

In Figure 1 in the text each cell represents a single (MSC, RBS) pair, and the MSCs of all neighboring cells are connected by a fixed network. A mobile unit can send and receive messages from only one RBS at a time, and only when it is in the cell associated with that RBS. The simplifying assumption that all neighboring base stations be physically connected can be easily removed by adding symbolic channels between the physically adjacent calls. The details of implementing this vary with the solution, and will be addressed in detail for our approach in a later section.

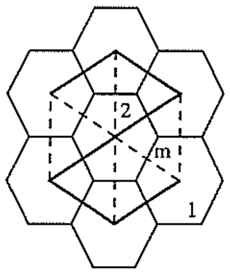

Fig. 1. Cell based broadcast.

A common broadcasting scheme in this model is to construct a spanning tree over the MSCs and send the message along this tree. In Figure 1, a spanning tree is indicated by the solid lines. For this example, suppose a mobile unit is located at cell 1, near the border of cell 2. Suppose cell 2 initiates the message to be delivered to the mobile unit. Following the proposed spanning tree broadcast, MSC_2 has RBS_2 broadcast the message locally, then forwards the message on the two outgoing links of the spanning tree. After sending the message successfully, MSC_2 deletes its copy of the message. The MSCs down stream behave in a similar manner, broadcasting locally, forwarding the message to their children, and finally deleting their copy of the message. If the mobile unit does not move out of cell 1, it will receive a copy of the message when it is broadcast by RBS_1. However, because the mobile unit is on the border between cell 1 and cell 2, it is possible for a handoff to be initiated and for the mobile unit to lose contact with RBS_1 and pick up communication with RBS_2. If this handoff occurs after RBS_2 deletes its copy of the message and before RBS_1 broadcasts its copy of the message, the mobile unit will not receive the message even though it was connected to the network during the entire broadcast lifetime of the message. Although in reality messages travel through the network much faster than a mobile unit can travel through space, because a basic handoff requires very little time to complete, and the length of the path along the spanning tree for the two cells m is moving between could be longer to traverse than for the handoff to complete, it is reasonable for a simple broadcast mechanism such as this to fail in this kind of circumstance.

3.2 Chandy-Lamport Snapshot

Therefore, we propose an alternative broadcast algorithm which guarantees the mobile unit in the above circumstance will receive a copy of the message. Our solution is based on the classical notion of a snapshot from distributed computing, in particular, our algorithm is based on the original Chandy-Lamport snapshot algorithm [Chandy and Lamport 1985].

Before moving on to our solution to message delivery, we must digress into some of the details of the general snapshot algorithm. A snapshot algorithm is typically executed over a graph consisting of processors and communication channels connecting these processors. The goal of the snapshot is to record a consistent state which is possible in an execution of the running computation. Nodes communicate via message passing along the channels. The state of the system consists of the values of the registers and memory at the processors and any messages in transit between the nodes. For example, a simple snapshot algorithm can freeze computation at all processors, record the state of the processors and the location of all messages, then restart the computation. This is impractical in most distributed settings, but it gives the intuitive notion of the results of a snapshot algorithm.

Although snapshot algorithms were developed to detect stable properties such as termination or deadlock by creating and analyzing a consistent view of the distributed state, minor adjustments allow them to perform message delivery in the dynamic, mobile environment. A traditional global snapshot is a collection of the local snapshots for the individual processing nodes consisting of the local state plus all messages in transit to that node. In general, distributed snapshot algorithms proceed across a network by passing the knowledge that the snapshot is occurring to neighboring nodes via control messages. The state of the processors and message locations are recorded. Every message appears exactly once in the global snapshot either on a channel or at a node.

To translate this into the mobile setting, the nodes in a distributed computation directly correspond to the mobile support centers and radio base stations of mobility. The messages recorded by the global snapshot correspond to the mobile units. Finally, the messages being delivered to the mobile units are carried in the snapshot control messages. Therefore, in the mobile setting, a local snapshot contains information about the mobile units in current communication with the base station, as well as the mobile units in transit to that node. Every mobile unit appears exactly once either at a node or on a channel. Because of the way various snapshot algorithms work, and the mobility inherent in the system, by the time the snapshot is collected and analyzed to locate the destination mobile unit, any message sent directly to that location may miss the mobile unit. For this reason, the computation of the snapshot itself must be modified to perform message delivery. Specifically, we note that because the mobile unit is emulating a message in the distributed setting and each message is recorded exactly once in a snapshot, each mobile unit in the mobility model must appear only once in the snapshot. By replacing message recording by actual delivery, the message will be delivered exactly once to the destination mobile unit by the base station which records the mobile unit in its local snapshot. More simply stated, recording a mobile unit is equivalent to message delivery.

3.3 Snapshot Delivery Algorithm

Throughout this section, we use the Chandy-Lamport algorithm for global snapshots. In the environment for this algorithm, FIFO channels are assumed between all processing nodes. For the remainder of this section we will assume single directional, FIFO channels along which both messages and mobile units move. In Section 3.3.3, however, we will relax this model by proposing reasonable additions to the handoff protocol to allow reality to model this FIFO channel. Further, mobile movement is characterized by a mobile unit moving into a channel. Also for the purposes of explanation we will eliminate the RBS from the model, but again, in Section 3.3.3 we will bring it back with slight modifications.

Because we are modeling message delivery, we assume the snapshot is initiated at the original location of the message, a single MSC. In general, snapshot algorithms proceed across a network by passing the knowledge that the snapshot is occurring to neighboring MSCs. We append to this knowledge the actual message. When the message arrives at an MSC, it is stored and the local snapshot begins. As the snapshot proceeds, the MSC watches for the destination mobile unit to arrive. If the local snapshot completes without recording the destination mobile unit as part of its state, the MSC deletes its copy of the message, assured that the destination mobile unit will appear in some other MSC's local snapshot. However, if the destination MSC is part of the snapshot, either in communication or on an incoming channel, this MSC is responsible for delivery. Once the message has been delivered, it can be deleted from the MSC. In this manner, as the local snapshots complete, the message copies are removed from the MSCs, meeting the requirement that no copies remain in the system.

We capture the essence of our solution in a pseudo code program shown in Figure 2. This notation shows an action based environment. Each action is one atomic step. The actions can be executed anytime the **when** conditions are true. We assume weak fairness of action execution meaning that in an infinite execution, each action is executed infinitely often. Because we do not formalize the notion of the channel between the MSCs, it is implicit that when a message or mobile unit moves from one MSC to another, it is put on a FIFO channel between the two MSCs and is processed from the head of the queue.

We assume that initially the system only contains the location of each mobile unit and the message to be delivered is at only one location. Therefore, the channels will have no messages on them and all memory is cleared. We introduce one auxiliary state variable into the system besides the message and mobile unit, and this concerns the flushed status of a channel. Basically flushed is used to indicate that the local snapshot of the channel is complete, however the specific definition and how this is determined will become evident as the algorithm is explained

A sample system state can be seen in Figure 3. In this figure, the blue circles indicate nodes that have not seen a

```
1  when message arrives at A from B
 a       mark channel (B,A) as flushed
 b       if message copy not at A
            send message on all outgoing channels
            keep local copy of message
            if mobile unit at A
               deliver message

2  when mobile unit arrives at A from B
         if channel (B,A) is not flushed and message copy at A
            deliver message

3  when anytime
         if mobile unit at node A and channel (A,B) exists
            mobile unit moves onto (A,B)

4  when anytime
         if all incoming channels flushed
            delete message copy
            unflush all channels
```

Fig. 2. Snapshot Delivery Code

copy of the message, red nodes are currently storing the message, and light blue nodes have deleted the message from local storage. The black circles indicate messages in transit. Node A is the origin of the message. Let us consider next the actions performed by the algorithm. When the message arrives at an MSC, Figure 3 node E, by definition, the channel it arrived on, (B, E) becomes flushed. If the MSC was already holding a copy of the message when the new message arrived, Figure 3 node A, no further action is taken (Figure 2 action 1). This is one of the cases where redelivery to a stationary mobile unit must be avoided. However, if the MSC was not previously holding the message, the arrival of the message triggers a delivery to the mobile unit when the latter is in communication with the MSC, and a dispersion of the message on all outgoing channels (action 1b). Through this fanning out of the message and the connectivity assumptions, every MSC eventually processes one message from every incoming channel. When a mobile unit arrives at an MSC, the MSC must determine whether or not to attempt to deliver the message. Trivially, if the MSC does not have a copy of the message, the decision not to deliver is made (action 2). However, if the MSC is marked and the mobile unit arrives on a flushed channel, it is known that the mobile unit has already received a copy of the message, and we must not redeliver it. This will be proven in the next section. If the MSC is marked and the mobile unit arrives on an unflushed channel, delivery is attempted.

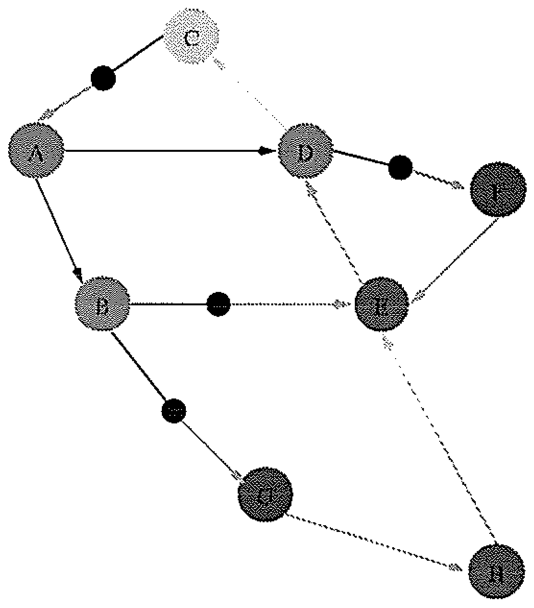

Fig. 3. Phases of delivery

The processing of a mobile unit from the head of a channel (action 2) models mobile unit movement from a channel to an MSC. We must also allow movement from an MSC to a channel. We add one statement to the program allowing a mobile unit which is located at an MSC to move onto any of its outgoing channels (action3). Because of the weak fairness assumption, the statement is eventually selected and the mobile unit moves on, thus modeling its random movement. In most real situations, the movement will be controlled by a user, however this could be coded into the program as a predefined path for the mobile unit to follow. Again, for generality, we focus on random movement.

The final statement of the program addresses the requirement that no trace of the message remain in the system when the algorithm terminates (action 4). To do this, the message must be deleted from every MSC, and the channels must return to their original, unflushed, state. We use the local snapshot property of the Chandy-Lamport algorithm to detect this. When the local snapshot terminates, all incoming channels are flushed, and the final statement is enabled for execution, and when it is selected, the cleanup is completed. In Figure 3, node C has already cleaned up its state.

3.3.1 *Properties.* In the previous section we presented a global snapshot algorithm modified to perform message delivery in a mobile system. Because the approach is based on a well understood algorithm from distributed computing, we can adapt the proven properties from the distributed computing environment into the mobile environment. Three main properties are proven for the Chandy-Lamport distributed snapshot: there is no residual storage in the system at some point after the algorithm begins execution, every message is recorded once, and no message is recorded more than once. We translate these properties directly into the mobile environment stating that (1) eventually there is no residual storage in the system at some point after the delivery process begins, (2) the message is delivered to the intended recipient, and (3) the message is delivered only once. In this section, we present a proof outline for these properties.

To show that eventually all information concerning the message is removed from the system, we must show that the program will eventually reach a state where there are no messages at any nodes, there are no messages in any channels, and the channel is unflushed. This can be shown by observing that by the connectivity of the graph and the dispersion rules of the algorithm, every node eventually receives a copy of the message on each of its channels. In Figure 3, this is analogous to a black message arriving at every node and turning it red. Once this occurs, each of the channels will be flushed, and the cleanup action is enabled, clearing the flushed variable and removing the copies of the message at each node. In the figure, all nodes and channels will turn light blue. There will not be any messages in channels because once a message has passed through a channel, it is not possible for another message to be sent down that channel. Therefore, we have shown that eventually the system is clear.

Next we must show that the message is eventually delivered. First we note (from the previous paragraph) that eventually all nodes receive the message. When this occurs, if the mobile unit has been delivered to, then the proof is complete. Otherwise, if the mobile unit has not been found then it must be the case that the mobile unit is located on an unflushed channel and the mobile unit must be in front of the message on this channel. In Figure 3, an unnotified mobile unit must be located at a blue node, or on a green channel, but in this case, we are describing a graph in which all nodes are already read, therefore the mobile unit must be located on a black channel. If the mobile unit was behind the message, on a black channel segment, it would have already received the message. Therefore, because the mobile unit arrives on an unflushed channel, and by the assumption that all nodes have received a copy of the message, the node it is heading toward must have a copy of the message. Delivery will occur when the mobile unit moves onto the node.

Having shown delivery, it remains to be shown that the message is delivered only one time. To do this, it is sufficient to show that after delivery occurs, a mobile unit cannot be in the position to be delivered to. These two delivery cases are (1) at a node without a copy of the message when the message arrives, and (2) arriving at a node with a copy of the message on an unflushed channel. To show each of these, we characterize a region of the graph called *will-be-notified* and define it as the set of all nodes which have not received a copy of the message, the channels which have not had a copy of the message pass through them and do not currently have a message on them, and the channel segments between a message and a node. In Figure 3, this region cooresponds to the blue nodes and green channels. With this definition, the above two conditions reduce to showing that once delivery has occurred, the mobile unit is never in *will-be-notified.* To show this is true, we use the intuitive notion of a path as a sequence of nodes and channels between a mobile unit and the *will-be-notified* region. It can be shown that on such a path, there must always be a message. Therefore, in order for the mobile unit to move back into the *will-be-notified* region, it must overtake this message, and because of the FIFO assumption of the channels and the rules for dispersion of the message, this is not possible. Therefore, the message can only be delivered once.

3.3.2 *Reality Check.* Here we reexamine assumptions we made to show that they are reasonable. The issues we discuss here are non-FIFO channels, multiple RBSs per MSC, base station connectivity, reliable delivery on links, the involvement level of MSCs, and storage requirements.

Non-FIFO Channels: One major objection to using the Chandy-Lamport algorithm is its reliance on FIFO behavior of channels. More specifically, in Section 3.2, we modeled both the mobile units and the messages as traveling on the same channel. This seems like an unreasonable assumption given that mobile units move much more slowly through space than messages move through a fixed network. To further explore this problem, we must adopt a more specific model of reality and how the FIFO assumption is broken. Next we will propose a simple fix and show why the channel models FIFO again. Finally, we will relate this back to the handoff protocol.

One of the American standards for analog cellular communication is AMPS. In AMPS, cellular telephones tune to only one frequency at a time. This shapes the handoff protocol. We adapt this protocol for use with mobile units and the snapshot algorithm. When the signal from one mobile unit begins to degrade, the MSC communicating with it on frequency f_1 sends a message to all bordering cells requesting a signal strength of f_1. After the responses are collected, MSC_1 selects the strongest signal as coming from the cell the mobile unit is moving into, MSC_2. At this point, a handoff begins. Figure 4a shows the stages of the handoff of mobile unit m moving from MSC_1 to MSC_2. First, MSC_1 sends a *handoff request* to MSC_2. MSC_2 responds with an available frequency f. Next, MSC_1 sends a *switch* message to the mobile unit with the new frequency. The mobile unit switches to the new frequency and sends a *hello* to MSC_2. Finally, MSC_2 sends a *handoff complete* message to MSC_1.

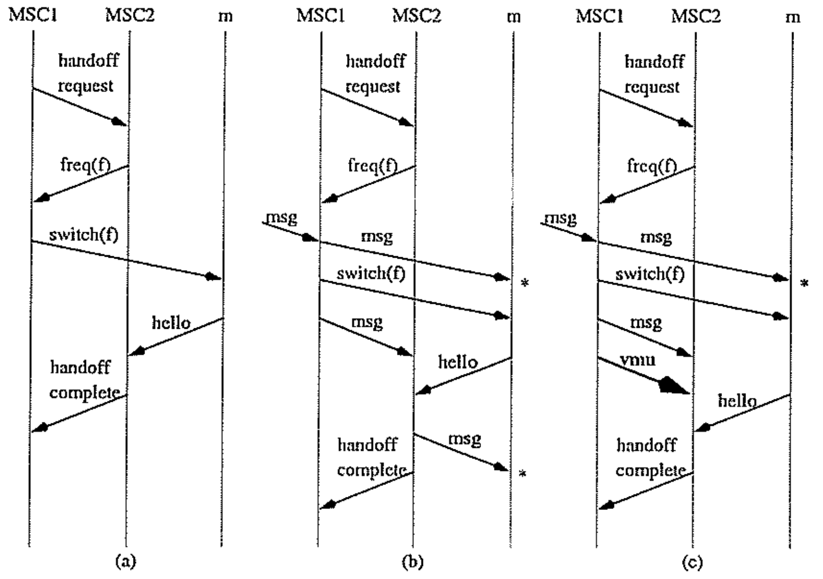

Fig. 4. a) AMPS-like handoff. b) Double delivery example (* indicates message delivery) c) Virtual mobile unit, FIFO

By using this approach, we know when a mobile unit is moving between cells, and which cells it is moving between. We also note that the mobile unit is not involved in the handoff until the last moment when it changes the frequency it is tuned to. Although this is slightly different with digital technologies and soft handoffs, we will leave that discussion for another paper.

The primary problem we face with this handoff is the possibility of double delivery of a message to a mobile unit. This situation arises because it is possible for a message to overtake the mobile unit while it is involved in a handoff. An example can be seen in Figure 4b.

When MSC_1 receives a message it broadcasts it to all mobile units in its cell, including m. After this broadcast, the switch message is sent to m and the message is forwarded to MSC_2. When MSC_2 receives the message and sees the *hello* from m, it does not know if the message was sent to m before or after the switch. Therefore, MSC_2 must send the message to m. In the example presented, this causes a double delivery.

The problem arises because instead of having one channel on which both messages and mobile units travel, we have, in effect, two channels which are not synchronized. One way to synchronize them is to simulate the switching of the mobile unit on the physical link by sending a packet representing a virtual mobile unit (*vmu*). The arrival of this *vmu* on the channel indicates to MSC_2 whether the mobile unit switched before or after the broadcasting of the message at MSC_1. If MSC_2 receives the *vmu* before the *hello* from the actual mobile unit, MSC_2 delays the broadcast of the message, if any are pending, until the arrival of m. If the actual mobile unit arrives before the mobile unit, all communication with m must be delayed until the *vmu* arrives. This is analogous to saying that while the *vmu* is in the channel, it has not arrived at MSC_2. Therefore, the channel has been made FIFO by simply sending one additional packet on the physical link between the MSCs. Figure 4c shows how the *vmu* affects the delivery. Just as the switch message was sent after the broadcast of the message, the *vmu* is sent after the message is put onto the channel.

One possible disadvantage to this solution is that if the mobile unit arrives before the *vmu*, there is a time when the mobile unit is in communication range of the system, however it is prevented from communicating because the system views it as being on a channel. Although this is probably unlikely because as noted previously, messages move along channels faster than the switch messages that cause the mobile unit to change location, there is a solution. It is possible to treat the entire handoff as being an atomic transaction. This is equivalent to putting the *vmu* into the handoff request. In order to make the transaction complete, MSC_1 must not broadcast any messages between the handoff request and the handoff complete. This too has its disadvantages, namely the delay in processing of messages at the MSCs. Therefore, although the first approach adds messages to the system and requires additional processing at MSC_2, it does not prevent broadcasting at either MSC.

At this point, we should note that this adaptation of the AMPS model of handoff does not allow the mobile unit to jump past the message in the channel. Therefore, it is not possible for the mobile unit to avoid the message. However, in figure 4b we did show how the message can be delivered twice. This problem of duplicate delivery can be eliminated by either the introduction of the virtual mobile unit or by using sequence numbers on each message and requiring the mobile unit to check this value before processing. Although this is a common approach in practice, it does require extra storage at the mobile units while the techniques presented above confine the additional processing and storage to the MSCs.

Multiple RBSs per MSC: Until this point we have only allowed one radio broadcast station for each support center, however, in today's cellular telephone system, MSCs manage sets of RBSs. For example, in Figure 1, it is feasible for one MSC to manage all cells shown. Because the algorithm we presented is intended to be run over the fixed network formed by the MSCs, the handoff mechanisms apply only to the movement of a mobile unit from a RBS supported by one MSC to another RBS supported by a different MSC. The question remains about how to broadcast the message within the cells supported by a single MSC and maintain the constraints of single delivery and guarantee the single delivery. Because the MSC acts as a coordinator of the mobile units present at each of the RBSs, it is feasible to run a similar snapshot among the RBSs, allowing it to terminate before any handoffs to other MSCs are permitted. This simple solution shows how our algorithm can be implemented in a layered fashion.

Base station connectivity: Another possible concern with the model we present is the necessity for physical connections between all MSCs whose cells border one another. Because of the high cost for such connectivity, it is possible that the physical wires may not exist. To allow the snapshot to function in such a setting, we add virtual channels between adjacent cells and include such channels in the snapshot as a channel along which control messages must be sent and received. In the implementation, however, we must be careful to ensure the FIFO nature of this virtual channel. It would also be possible to add a virtual channel between two non-adjacent cells if a mobile unit was likely to move between them in a disconnected manner. This would only work if the same type of handoff was used for this type of mobility, which is not likely the case because disconnection is typically a longer-lived situation, but the possibility is worth mentioning.

Reliable delivery on links: The snapshot delivery algorithm assumes that link delivery is reliable. Most of the Internet uses unreliable links like Ethernets, frame relay, and ATM. The probability of error on such links may be small but packets are indeed dropped. A possible solution is to add acks for multicast messages as is done, for example, in the intelligent flooding algorithm used in Links State Routing in OSI [Zimmerman 1980] and OSPF [Moy 1994]. Another solution is to only provide best-effort service. Since lost messages can lead to deadlock we need to delete a message after a timeout even if it is still expected along a channel.

Involvement level of MSCs: In every snapshot, every MSC must be involved to guarantee delivery and termination. In a paper on running distributed computations in a mobile setting [Badrinath et al. 1996], the authors warn against requiring involvement of all mobile units in a computation, especially due to the voluntary disconnection often associated with mobile computing. Such disconnection is often done to conserve power, or in some cases, to allow disconnected operation. In either case, the mobile unit is not available for participation in the distributed algorithm. These arguments are important for creating distributed algorithms for mobile computing environments, however our goal is not to create a global snapshot of the mobile units, but instead to employ the snapshot technique to a different end, namely message delivery. Additionally, the control messages of the snapshot are not being run over the mobile units, but rather over the fixed mobile support centers. In order to guarantee delivery, the mobile unit must be present in the system, however, this is a reasonable assumption because there is no means to reach a disconnected mobile unit. At this point, it is interesting to note that if the mobile unit is not present in the system during a delivery attempt, the algorithm will terminate normally, removing all trace of the message from the system, but without delivery.

Storage requirements: In snapshot delivery, we assume that the MSCs hold a copy of the message for delivery to the mobile agents for a bounded period of time limited to the duration of the local snapshot. In a system with bi-directional channels, because the local snapshot terminates when the message arrives on all incoming channels, a local snapshot can be as short as a single round trip delay between the MSCs. One can argue that it is not the place of the MSCs to be maintaining copies of messages when their primary purpose is routing. However, in this case, because no routing information is being kept about the mobile units, the system will be required to keep additional state in order to provide delivery guarantees. Therefore, keeping a copy for a short duration is a reasonable assumption.

3.3.3 *Extensions.* The snapshot delivery algorithm is extendible to perform delivery of multiple messages simultaneously, delivery to rapidly moving mobile units, route discovery, and multicast. Each of these will be examined in turn.

Multiple message deliveries: To deliver multiple messages simultaneously using snapshots, we run several copies of the algorithm in parallel. This is analogous to having each MSC index and store the incoming messages and maintain separate channel status for each message in the system. This information is kept until the MSC locally determines it can be cleared. In the worst case, every node must have storage available for every potential message in the system, as well as maintain channel status with respect to each message. Although this appears excessive, we maintain that the nature of the snapshot algorithm in a real setting will not require maximum capacity. In other words, because the MSCs are able to locally determine when to delete the messages, the nature of the network will determine how long a message is stored at the MSC.

Rapidly moving mobile units: Another advantage to this algorithm is the ability to operate in rapidly changing environments with the same delivery guarantees. In Mobile IP, mobile units must remain in one place long enough to send a message with their new address to their home agent for forwarding purposes, and remain at that foreign agent long enough for the forwarded messages to arrive. With forwarding enhancements added to the foreign agents in Mobile IP, the issue is minimized because the former location of a mobile unit becomes a kind of packet forwarder. However, even with forwarding, if the agent moves too rapidly and the system is unable to stabilize, forwarded packets will chase the mobile unit around the system without ever being delivered. Because snapshots do not maintain a notion of home or route, movements are immediately accounted for by the delivery scheme.

Route discovery: In more moderately changing environments, route discovery can increase efficiency and decrease overhead. In these situations, the snapshot delivery algorithm can be used to perform route discovery. When the discovery message from source S located at MSC_S arrives at the destination mobile unit D located at MSC_D, the mobile unit responds with a packet directed toward MSC_S with MSC_D and a particular RBS in its data to identify its location to the source. This assumes that the source is also moving relatively slowly, however a route response message could be sent to S from D in a similar manner as the route discovery. All packets sent to D after the discovery should contain MSC_D and RBS, and routing along the fixed network is now possible.

Multicast: As stated previously, another concern in the mobility community is multicast. While Mobile IP addresses the issues of macromobility, and some work has been done on reliable multicast for micromobility [Acharya and Badrinath 1993] when the set of recipients is known, our algorithm easily extends to perform multicast to all mobile units in the system during the execution of the snapshot without knowing the list of recipients. Without changing the snapshot algorithm, it can be shown that delivery of a message is attempted to all mobile units in the system before the algorithm terminates. If the message contains a broadcast or multicast destination address, delivery can be carried out whenever a connection with the mobile unit accepting those addresses is established. Interestingly, even in broadcast or multicast, the restriction of single delivery of a message holds. Although this description is concise, the importance of it should not be lost in its simplicity.

Our modified snapshot algorithm has worst-case overhead of one message per link in each direction to multicast. By contrast, the algorithm used in IP DVMRP [Deering and Cheriton 1990] effectively computes a tree. Its overhead is the number of links in the tree plus the number of links that have endnodes that participate in this multicast.

4. ROUTE MAINTENANCE

Another approach to message delivery in a mobile setting involves tracking the location of the mobile unit. In Mobile IP, this is accomplished by having the mobile unit register with its home agent every time it moves. The home agent is supplied with a new IP address for the mobile unit, and depends upon normal IP routing to send the packets to the new address. Mobile IP concentrates all information about the new location at the home agent and relies on updates from the mobile unit to maintain connectivity.

Extensions to Mobile IP introduce route optimizations that allow the foreign agents to become active in the forwarding process when a mobile unit moves. Forwarding has also been explored in a more general sense at the routing layer with the development of active routers [reference]. The routes to the mobile units are updated at the routers as well as at the home agent. If a packet arrives at a router which knows the direction to the current foreign address of the mobile unit, the packet does not have to return all the way to the home agent before being forwarded. Instead, the router forwards the packet immediately. Issues to be addressed in this approach include the consistency of the routes kept by the routers and the frequency of updates made by the mobile units to the routers in order to minimize cache inconsistencies.

Both Mobile IP and active routers attempt to maintain up to date information about the mobile unit, in effect performing a route maintenance operation. This approach has several advantages over the broadcast search mechanisms presented in the previous section. One such advantage is in moderately changing environments where a mobile unit is stationary for an extended period of time and is exchanging packets with another relatively stationary mobile unit. Broadcasting to the entire network is expensive and overburdens it with unnecessary traffic. Discovering a route and using it for all packet transfers can minimize overhead in such a system.

While Mobile IP maintains a route from a fixed home agent to the mobile unit and active routers allow two mobile units to establish a route between them, our approach uses a home agent and maintains a subset of nodes representing a subset of the history of movement of the mobile unit. As before, we adopt an algorithm from distributed computing, specifically diffusing computations developed by Dijkstra and Scholten [Dijkstra and Scholten 1980] for termination detection.

4.1 Dijkstra-Scholten

As before, the model of distributed computing we consider is a set of processing nodes communicating via message passing. The term diffusing computation describes a set of algorithms in which a single processor serves as the root of the computation. As messages are sent from the root to other nodes, these nodes are added to a tree representing nodes which are participating in the computation. As messages are sent, more nodes are added to the tree. When a leaf node completes computation, it signals to its parent that is is no longer part of the tree. When all children of a node have been removed, this node can also be removed. When all children of the root have signaled their termination and the root has completed its processing, the overall computation has terminated.

In general, we wish to exploit the graph management approach of diffusing computations. As with the snapshot approach to message delivery, our objective is to provide a mapping from diffusing computations to mobile unit tracking. This is accomplished in a similar manner as before by modeling the messages of a distributed computation as mobile units. Intuitively, as the mobile unit moves among nodes, the tree expands to include the new location of the mobile unit. As the mobile unit returns from leaf nodes to nodes already in the graph, the leaf nodes are no longer necessary in order to keep a path from the root to the mobile unit. Therefore, the tree is allowed to shrink to remove these unnecessary segments of the history. Because the tree tracks the movement of the mobile unit by attaching its current location to the tree, message delivery is a simple superposition of a broadcast onto the history tree.

We will approach the details of this algorithm by presenting a progression of tracking algorithms which use diffusing computations as a basis. Initially, we will explore only the graph growth and message delivery portions of the algorithm. We will then allow nodes which are not aware of message delivery to be removed from the graph, and finally, all nodes not on the path from the root to the mobile unit will be allowed to be removed from the graph, whether or not they are aware of the message delivery. Following this presentation, we will describe another algorithm inspired by diffusing computations which uses a slightly different graph management algorithm.

4.1.1 *Ever Expanding Tree.* As a starting point, we introduce only the tree growing portion of the diffusing computation, essentially creating a tree rooted at the home agent representing all the nodes the mobile unit has visited. When a mobile unit moves to a node for the first time, the node is attached to the tree as a child of the node the mobile unit arrived from. Once a node is in the graph, it is never removed. The delivery scheme imposed on top of the graph is simply to leave a copy at every node in the tree starting from the root and propagating to all children links until every node has a copy of the message. Because the mobile unit is always in the tree and the message is propagated to every node in the tree, eventually the mobile unit and the message will be at the same node and delivery will occur.

The specific actions necessary to this algorithm can be seen in Figure 5. We make the same atomicity assumptions as before, allowing each action to be one atomic step and using weak fairness in action selection. Initially, we assume that the node is at the root and there is no message in the system. At some arbitrary time, a message is introduced at the root of the tree, initiating delivery. Although the nature of the approach is simple, one of the first questions that arises in implementation due to the asynchronous nature of communication is how to decide whether the node the mobile unit is en route to should be added to the graph or is connected through another portion of the graph. This could easily be solved by propagating global information about the tree structure to all nodes, however a local solution is more scalable. Another way to state the problem to say that we must maintain the invariant that the mobile unit is always in the tree, but not violate the nature of the tree by having multiple paths to a single node from the root. To accomplish this, we introduce the notion of a potential child and a real child. When a mobile unit moves from node A to node B, B is added as a potential child of A. The potential child links are not actually part of the tree, but are necessary in the message delivery portion of our algorithm. When the mobile unit arrives at B (Figure 52), a local decision is taken whether or not the node is already in the graph based on the existence of a parent pointer at B. B then sends a control message to A indicating whether it should be added to the real child list or simply deleted from the potential child list.

The question naturally arises why the potential child status is even necessary. From the previous discussion, it appears that the control messages from B would be enough to decide on the connectivity of the graph. Without the need for message delivery, this is true, however our goal is to guarantee message delivery in the presence of mobility. Therefore, we need the message to propagate to all nodes in the graph. In a static situation, this is trivial, however in the presence of movement, we must take into account the nodes the mobile unit is moving onto. Therefore, during delivery, nodes will forward the message to both real and potential children, therefore including all nodes in the message delivery process. In order to eliminate the possibility of a node receiving the message twice, we add the restriction that a node accepts messages only from its parent. Therefore, by the time the mobile unit has been processed (due to the atomicity assumptions), the node knows whether or not any subsequent message are coming from the parent or are duplicates. Therefore, we are able to avoid duplicate processing of the message at each node. This is an important issue because we want to be able to clean the message up after termination. If a node was allowed to receive the message message multiple times, we would not be able to ensure that the message was eventually removed from the whole system.

To prove that message delivery is guaranteed, we introduce the concept of a complete level in the tree. A complete level is only meaningful during message delivery and is defined to be the lowest level of the tree above which all nodes have a copy of the message, and at which all nodes have a copy of the message. For example, Figure 6, the finished level is two because all nodes at level one have the message, and although the message has propagated to level three, there is a node at level three which does not have a message copy. Intuitively, once all nodes at a level have a copy of the message, the number of nodes at this level cannot increase unless message delivery has occurred. To show this, we observe that in order for the number of nodes at a level to increase, the mobile unit must be traveling from a node in the previous level to a node not in the graph. However, if the mobile unit is at a node in the previous level, by the definition of the lowest complete level, this level must be also be complete, the mobile unit must have received a copy of the message.

It can further be shown that the number of complete levels must increase until delivery has occurred. To show this, we observe that on the level immediately below the last completed level, there is a fixed number of nodes without the message. By similar reasoning as before, this number cannot increase without delivery occurring. Therefore, we show that between the completed level and each node in the next level without the message copy, there must be a copy of the message on the channel. This is true because the message is propagated along the channels to all real as well as potential children (5c). Because of the finite number of nodes in the system and the fact that the nodes cannot be added at the last completed level or the next level below, the message must be delivered eventually, even if every node in the system becomes part of the tree.

Throughout the above argument, we have used the region created by the diffusing computation algorithm to limit the broadcast of the message to a tree representing a history of the movement of the mobile unit, and have shown that a message originating at the root is guaranteed to be delivered to the mobile unit. We can also show that after delivery, an acknowledgment is sent to every node and all copies of the message are deleted. Intuitively, the connectivity of the graph provides the means for propagation of the cleanup ack. Because the ack is propagated to both parent and child, it will eventually reach all nodes in the graph(2a, 5b, 6).

Although this algorithm provides a clean approach to guaranteed delivery, its graph maintenance algorithm is cumbersome because once a node is added to the tree, it cannot be removed. In the next section, we extend the algorithm to allow nodes to be removed from the tree while they are not participating in any message delivery.

```
1   when mobile unit moves from A onto channel (A,B)
          A records B as a potential child

2   when mobile unit arrives from channel(A,B) onto B
 a        if message at B
            deliver message
            B sends ack to parent and child
 b        if B has parent (B is in the graph)
            send notchild(B) to A
 c        else (B is not in the graph)
            B sends ischild(b) to A
            B sets parent to A

3   when ischild(B) arrives at A
          B is moved from potential child list to real child list

4   when notchild(B) arrives at A
          B is removed from potential child list

5   when message arrives at B from A
 a        if A is parent(B)
            store message
 b          if mobile at B
              deliver message
              delete message
              send ack to A
 c          else
              send message to all children and potential children

6   when ack arrives at A from B
          if message at A
            delete message
            send ack to all children (real and potential) and parent, except B
```

Fig. 5. Graph never shrinks

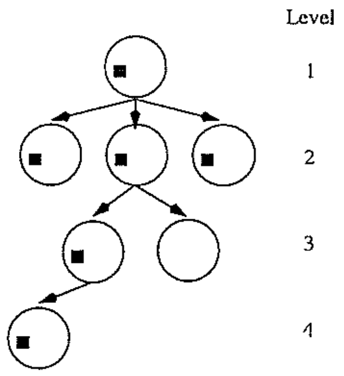

Fig. 6. Last level of completion equal to 2

4.1.2 *Tree shrinks when not delivering.* Our next refinement uses the same graph growth and message delivery as the previous algorithm, but we add two new statements to allow nodes to be removed from the tree while the message is not being delivered. With this extension, the tree can shrink to a more reasonable size, allowing nodes that are no longer in the path from the root to the mobile unit to be removed and therefore not included in future message deliveries. The same state information is kept at each node, but we add a new message type: *remove*. A *remove* message is issued from a leaf node when it does not have a copy of the message and the mobile unit is not present (see Figure 7). When the remove arrives at the parent, the child is removed from the real child list. With this enhancement, we must again show that message delivery is guaranteed and that the message is deleted from each node after delivery.

To show guaranteed delivery, we can adopt the argument of completed levels presented in the previous section. The

```
7  when anytime
         if A has no children (potential or real)
            AND A has no message copy
            AND mobile not at A
               send remove(A) to parent
               set A's parent to null

8  when remove(A) arrives at B from A
         delete A from realchildlist at B
```

Fig. 7. Statement replacement and addition to allow node deletion during inactivity

main difference that we must acknowledge is the fact that the number of nodes without the message at the level below the last completed level can decrease in two ways: either by the message arriving, or by the node issuing a *remove* request. We examine three possible conditions for the channel state when a *remove* message is sent from a leaf in Figure 8.

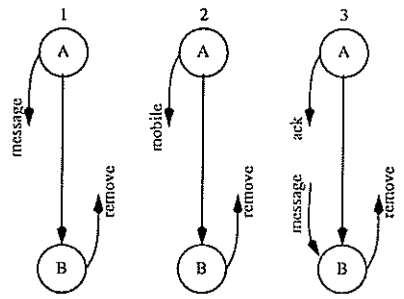

Fig. 8. Possible message/remove interactions

In the first situation, the parent, A, receives the message and sends it to the child, B, at the same time that B send a *remove* to B. It is important to note that when the *remove* is issued, the child no loner keeps any information about the old state. Therefore when the message arrives at B, statement 5 will be invoked, but the originator of the message is no longer recorded as the parent and the message will be dropped. When the *remove* arrives at A, the former parent, it cannot (and need not) distinguish whether the message arrived before the *remove* was sent or after. Therefore the *remove* is successful, we have no dangling messages, and we are not concerned about the message missing the mobile unit because it is not located at the node which was just removed. If the mobile unit again moves from A to B, we can be assured the delivery has occurred because A had a copy of the message.

The next case involves the mobile unit moving toward a node which has just sent a *remove* message. When the mobile arrives at B, B no longer has a parent, so an *ischild*(B) is sent to A. When the mobile left A, B was put into the potential child list (even though it was already in the real child list). When the *remove* arrives, B is taken out of the real child list but the potential child list is not touched. Therefore, B is re-added as the child of A. Delivery is not a concern because the message will follow the potential child list and will catch up with the mobile unit.

The final case is similar to the first, except we add the notion that the parent has sent both the message and the ack before the *remove* is sent from the child. When the message arrives at B after the *remove* is sent, it is dropped because A is no longer the parent of B. Next, the ack is dropped because the message being acked is no longer present at B. A has no way of knowing the that the message and ack were dropped because no state relevant to that information is being kept.

The next question to address is how message delivery is guaranteed. Delivery occurs in two ways, either the mobile unit is present at a node when the message arrives (5), or the mobile unit arrives at a node and the message is there already (2). There are two possible interesting states to look at, either delivery has occurred or delivery has not occurred. We are only interested in the state where delivery has not yet occurred. If delivery has not occurred, then either the mobile unit is at a node without the message or the mobile unit is on a channel. At this point we note that once a node has a copy of the message, it cannot be removed from the tree until the ack arrives and the message is deleted, i.e., delivery has occurred somewhere in the graph. We also note that by the connectivity of the graph, a node will either receive a copy of the message from its parent or be removed from the graph.

In proving delivery, we next define the depth of the tree, m, to be the equal to the last complete level as defined previously. By the definition of this level, there are no messages in transit between level m and level $m-1$. Therefore, no new nodes will be added to the graph at this level or above unless delivery occurs. For this to happen, the mobile unit would be moving from a node in this set to one outside, but because in such a case the mobile would have to be located at a node in the set and all of the nodes have a copy of the message, delivery must have occurred.

To show that delivery must occur, we show that either m increases or delivery occurs. For the level to increase, there must be at least one node on the next level without a copy of the message. On the channel from the parent to this node, there must be a message. When the message arrives, either the node receives it and we are closer to increasing the level (by decreasing the number of nodes at this next level without message copies), or a *remove* has been sent and the message is dropped. When the *remove* arrives, the child is deleted and we are closer to increasing the level (by decreasing the number of nodes at the next level). Because the level can only increase and there is a maximum level equal to the number of nodes in the system, delivery must occur. It should also be noted that acks are not generated until delivery, so messages cannot be deleted prematurely. We can further show that by the connectivity of the graph and rules for ack propagation, acks do propagate to all nodes with message copies.

4.1.3 *Graph shrinks during delivery.* The next logical refinement is to allow the graph to shrink even when the message is present at a node. By further allowing the nodes that are not on the path from the root to the mobile unit to be removed from the message distribution tree, we further improve the efficiency of the message delivery algorithm by reducing the number of nodes involved. To realize this refinement, we alter the conditional in statement 7 by removing the restriction that the node generating the *remove* must not have a copy of the message, see Figure 9. We must also be careful to delete the message at the node so that we are able to guarantee full cleanup of all message copies.

```
7  when anytime
       if A has no (potential or real) children AND mobile unit not at A
         send remove(A) to parent
         set A's parent to null
       if message at A
         delete message
```

Fig. 9. Statement replacement to allow node deletion during delivery

To prove message delivery with this extension, we use the same construction of complete levels as before. The primary difference is that a complete level can shrink in size as nodes are removed from the graph. Completed levels still cannot grow unless delivery occurs, and although nodes can be removed from a completed level, they can only rejoin at a lower level. The levels increase by the same rules as before. The upper bound on the number of levels is still the number of nodes in the systems, and therefore by the connectivity of the graph and the propagation rules, the message will be delivered.

4.1.4 *Discussion and extensions.* With this final version of the diffusing computation tree construction rules in place, we now have an efficient algorithm that actively maintains information about the region of the graph where the mobile unit has been recently. With higher bandwidth and computation given over to maintenance of the graph, the graph itself can be made smaller and more efficient. Because of the way the graph is constructed, it is possible for the route from the root to the current location of the mobile unit along the tree to be less than optimal. Therefore, a logical extension is to run an optimization protocol to reduce the length of the path from the root to the mobile unit while maintaining connectivity. This type of algorithm should be explored to provide even more efficient use of the network. The tradeoff with this approach is between the increased bandwidth required to run the optimization, and the increased benefit by reducing the number of nodes involved in message delivery.

While we have only discussed this approach for one mobile unit it is possible to extend this directly to include information for a group of mobile units. With such an extension, sending a multicast message to the group would require modifying the cleanup portion of the algorithm to ensure at least once delivery of the message to all recipients. If the current approach were employed, it would be possible for delivery to occur at one node and initiate the cleanup, and for another mobile unit to move from a region in which the message had not yet been propagated into the region that had been cleaned up. One possible practical approach would be to not run an explicit cleanup protocol, but instead allow messages to be deleted after a timeout equal to the diameter of the graph. This would allow sufficient time for all the nodes in the graph to receive a copy of the message and for delivery to occur, but would also ensure that cleanup would eventually take place.

4.2 Backbone and Tails

Having developed the standard diffusing computation graph maintenance algorithm, we now introduce a different graph maintenance algorithm inspired by the previous investigation. Our goal in this case is to reduce the complexity of the graph, therefore we note that only the path between the root and mobile unit is actually necessary for delivery. In the previous approach, although the rest of the graph can be cleaned up by the remove rules, it is still possible for a message to be propagated to these nodes unnecessarily. To avoid this, our new approach only propagates messages down the path from the root to the mobile unit. To accomplish this, we require the mobile unit to carry additional information concerning the graph structure. The specific information will be described later.

4.2.1 *Graph maintenance.* In general, this approach creates a graph that has a specific appearance which we refer to as a backbone and tails. The backbone is the sequence of nodes and channel segments from the root to the current location of the mobile unit. If the mobile unit is on a channel, the segment between the last node in the backbone and the mobile unit is included in the backbone. The remainder of the graph form what we refer to as the tails, or sequences of nodes and channels that point toward the backbone. In the previous approaches, the graph segments not in the path from the root to the mobile unit terminated in leaf nodes with no children. In this case, the top of the tail segments are not connected to the graph, but the last node of the tail points to a node in the backbone. In Figure 10b, the nodes $\{D, E, F\}$ form one tail and node G forms another tail. The nodes from the root to node C form the backbone.

The graph expansion for backbone and tails is the same as in the previous algorithms. When a mobile unit arrives at a node that is not part of the tree, the node is added as a child of the node the mobile unit arrived from (Figure 11, statement 2). If the node the mobile unit arrives at is already in the graph, no new nodes are added, but the links are changed. Our primary goal is to create tails while preserving the backbone. Figure 10a and b shows the effect of a mobile unit moving from node F to C, where C is already on the backbone. The movement of the mobile unit makes C the child of F. To create the tail, the link from C to its child, D, is removed by sending a *delete* message to D. To preserve the backbone, we must detect whether or not to change the parent pointer of C. From the figure, it is clear that no change should be made, however the question remains how can C determine this locally. To do this, we keep a copy of the backbone with the mobile unit. Because it is the movement of the mobile unit that causes the change in backbone, it is reasonable to keep this information. In the case depicted in Figure 10b, node C is already in the mobile unit's backbone list. Therefore, the backbone list is truncated just after C, and C's parent pointer does not change. If, however, the mobile unit now moves from C to E (Figure 10c), E's parent pointer should change to C from D. This is evident from the backbone list maintained by the mobile unit because E is present in the list. Therefore, by adding the backbone list in the mobile unit, it is possible for a node to locally detect when it needs to change parent pointers.

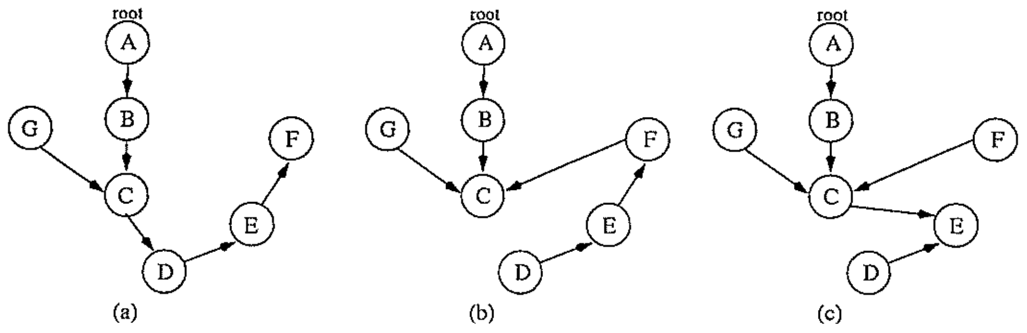

Fig. 10. Mobile unit movement from A to B

Next we need to discuss the graph shrinking. Graph shrinking occurs only along the tails. As stated previously, when a mobile unit moves to a backbone node creating a tail, a *delete* message is sent to the new tail. This delete message signals the node that it no longer has a parent in the backbone, and should therefore remove itself from the graph. In the process of removing itself from a graph, the node also signals its child with a *delete* message, therefore propagating the removal along the tail. When the *delete* finally arrives at node in the backbone, this node will ignore the *delete* because the message arrived from a node which is not its parent. In this process, the tail is effectively deleted without affecting the backbone.

4.2.2 *Guaranteed message delivery.* The next important aspect of this algorithm is message delivery. From Figure 10 it is clear that the path from the root to the leaf should be used for sending the message to the mobile unit. Therefore, we start the message at the root node and forward it along the child pointers (action 3). From the growth rules of the graph, each node will have only one child. This propagation continues until the message catches up with the mobile unit (action 3) and delivery occurs or the mobile unit arrives at a backbone node, essentially catching up with the message (action 2). As before, we must ensure that the delete messages do not interfere with the message propagation. To do this, we simply show that the backbone always exists and that it cannot be destroyed. To do this, we must show that there is never a delete message on the backbone. The details of this are not given, however the intuition is that a delete message will only be generated downward to a former child, and because delete messages are only accepted from parent nodes, the delete being propagated along a tail will stop at the backbone node since the backbone's parent is pointing toward the root and not toward the tail.

As soon as delivery is complete, the message is deleted and an ack is sent out to the parent to continue to delete the message copies. The ack propagates upward through the backbone until reaching the root. It should be observed that in the graph maintenance algorithm the backbone can be cut at any point. Therefore, if the ack only propagated along the original backbone until reaching the root, this cut would prohibit the continued propagation of the ack. To solve this problem, we note that it is the arrival of the mobile unit at a backbone node which causes the cut in the backbone. Therefore, the mobile unit can simply generate a new ack for the message that will propagate upward from this node. The original ack will not reach this node because it is no longer considered to be its child. Because the nodes in all tails will eventually be removed from the graph (deleting any message copies) or added at another point (deleting any message copies and initiating a new ack), all message copies will eventually be deleted from the graph.

4.2.3 *Extensions.* As described, this algorithm does provide guaranteed message delivery to mobile units because the message is stored at every node in the backbone until delivery occurs. However, the question naturally arises what happens if this storage is eliminated and a single copy of the message propagates down the backbone. In this case, it is possible for the mobile unit to run away from the message by moving in a figure eight pattern. For example, if the message travels down a chain of nodes while delivery is in progress and returns to the backbone at a point above the message, effectively putting the message on a tail. The mobile unit then continues to move along a series of nodes while the message continues on back toward the backbone. Again, the mobile unit returns to the backbone above the message, effectively running away from the message. This pattern does prevent delivery, however if the movement of the mobile unit is slow in comparison to the propagation of the message, it is unlikely that such a scenario would be encountered. Therefore, this approach provides reasonable limits on message delivery, while significantly reducing the storage necessary in the graph and also eliminating the need for a cleanup ack to remove message copies.

Because the growth of this graph follows the movement pattern of the mobile unit, it is likely that the graph is non-optimal as far as distance between the root and the mobile unit. Several algorithms from graph theory exist to optimize paths while maintaining connectivity. In a system with moderately moving mobile units, this type of optimization could be applied periodically to optimize the routes without losing any of the delivery properties.

5. CONCLUSIONS

In this paper we presented two algorithms adapted from distributed computing to perform tasks in mobility. The first algorithm adopted a snapshot algorithm to search for a mobile unit a fixed network and deliver a message. The second algorithm used a model of diffusing computations to track a mobile unit as it moves among nodes in a fixed network. Message delivery is then imposed on top of this changing structure.

One possible integration of these two ideas is to use tracking at the macromobility level and search for micromobility. In other words, the overall structure of the network would be two layers, the high level consisting of clusters of cells, and the low level containing the individual mobile support centers of each cell. Tracking would be maintained only on the high level, and a snapshot would be applied at the lower level. Although some assumptions about mobile unit movement during delivery might have to be made in order to get the same strong assumptions as we made before, this approach seems feasible.

Our primary contribution in this work is the introduction of a new approach to the study of mobility, one based on a model whose mechanics are borrowed directly from the established literature of distributed computing. Treating mobile units as messages provides an effective means for transferring results from classical distributed algorithm literature to the emerging field of mobile computing.

```
1  when mobile unit moves from A onto channel (A,B)
         A records B as child

2  when mobile unit moves from channel (A,B) onto B
         if B in mobile unit's list of nodes
           B keeps old parent
           mobile's list of nodes is truncated from B to the end
         else
           B sets parent to A
           B added to mobile unit's list of nodes
         B sends delete to child, if any
         B sets child pointer to null
         if message at B
           deliver message
           send ack to parent
           delete local message copy

3  when message arrives at B from A
         if A is parent(B)
           if mobile at B
             deliver message
             send ack to A
           else
             keep local copy of message
             send message to child

4  when ack arrives at A from B
         if message at A
           delete message
           send ack to parent

5  when delete arrives at A from B
         if B is parent(A)
           if message at B
             delete message
           send delete to child
           empty parent pointer
           empty child pointer
```

Fig. 11. Backbone and tails

REFERENCES

Acharya, A. and Badrinath, B. R. 1993. Delivering Multicast Messages in Networks with Mobile Hosts. In *13th International Conference on Distributed Computing Systems.* IEEE, New York.

Badrinath, B. R., Acharya, A., and Imielinski, T. 1994. Structuring Distributed Algorithms for Mobile Hosts. In *Proceedings of the 14th Intl. Conf. on Distributed Computing Systems.* Poznan, Poland, 21–8.

Badrinath, B. R., Acharya, A., and Imielinski, T. 1996. Designing Distributed Algorithms for Mobile Computing Networks. Technical Report DCS-TR-??, Rutgers University. submitted for publication.

Chandy, K. M. and Lamport, L. 1985. Distributed Snapshots: Determining Global States of Distributed Systems. *ACM Transactions on Computing Systems 3,* 1, 63–75.

Deering, S. E. and Cheriton, D. R. 1990. Multicast Routing in Datagram Internetworks and Extended LANs. *ACM Trans. on Computer Systems 8,* 2, 85–110.

Dijkstra, E. and Scholten, C. 1980. Termination Detection for Diffusing Computations. *Information Processing Letters 11,* 1.

Eriksson, H. 1994. MBONE: The Multicast Backbone. *Communications of the ACM 37,* 8, 54–60.

Gray, R., Kotz, D., Nog, S., Rus, D., and Cybenko, G. 1996. Mobile agents for mobile computing. Tech. Rep. PCS0TR96-285, Dartmouth College. May.

Ioannidis, J. and Maguire, Jr., G. Q. 1993. The Design and Implementation of a Mobile Internetworking Architecture. In *1992 Winter Usenix.*

Jacobson, V. and McCanne, S. Visual Audio Tool.

Johnson, D. B. 1996. Scalable Support for Transparent Mobile Host Internetworking. In *Mobile Computing,* T. I. Korth and Hank, Eds. Kluwer Academic Publishers, 103–128.

Kistler, J. J. and Satyanarayanan, M. 1992. Disconnected Operation in the Coda File System. *ACM Trans. Computer Systems 10,* 1, 3–25.

McCanne, S. and Jacobson, V. 1995. vic: A Flexible Framework for Packet Video. In *ACM Multimedia '95.* San Francisco, CA, 511–522.

MOY, J. 1994. OSPF Version 2. Internet draft, Internet Engineering Task Force. March 1994.

PERKINS, C. 1996. IP Mobility Support. Technical Report RFC 2002, IETF Network Working Group. October.

RANGANATHAN, M., ACHARYA, A., SHARMA, S., AND SALTZ, J. 1997. Network-aware Mobile Programs. Tech. Rep. CS-TR-3659, University of Maryland, College Park.

ZIMMERMAN, H. 1980. OSI Reference Model – The ISO Model of Architecture for Open Systems Interconnection. *IEEE Transactions on Communication 28*, 425–432.