Author Affiliations:

Contact Author:
Amy L. Murphy
University of Rochester
Rochester, NY, USA
Contact info for June 2003 to May 2004
email: murphy@cs.rochester.edu
phone: 39-02-2399-3698
fax: 39-02-2399-3411

Gruia-Catalin Roman
Department of Computer Science and Engineering
Washington University
St. Louis, MO, USA

George Varghese
Department of Computer Science and Engineering
University of California, San Diego
San Diego, CA, USA

Dependable Message Delivery to Mobile Units

A.L. Murphy, G.-C. Roman, G. Varghese

August 8, 2003

Abstract

Mobile computing is emerging as a novel paradigm with its own characteristic problems, models, and algorithms. Much effort is being directed to integrate mobile units with fixed networks, providing bridges to connect wireless to wired. The result is a fixed core of wire-connected static nodes and a fluid fringe of wireless mobile units, a computing system similar to the cellular telephone network. The model we put forward uses the graph of fixed nodes as a foundation and models the mobile units themselves as persistent messages moving through this network graph. Such a model allows algorithms from traditional distributed computing to be directly implemented in the mobile environment, however, it has been shown that the unique properties of mobility, such as limited bandwidth and disconnection, make such direct translation impractical. This paper presents a fundamentally different idea. Instead of recreating the functionality of distributed algorithms in the mobile domain, we show how distributed algorithms can be adapted to solve problems unique to the mobile environment. Specifically we focus on the problem of dependably delivering a message to a moving unit. We demonstrate this technique with two new algorithms, the first based on distributed snapshots and the second on diffusing computations.

Keywords: Mobile computing, communication, distributed snapshot, termination detection.

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 world with the computer at their side. 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. This results in at least two kinds of research questions. First, what is the precise relationship between mobile computing and traditional distributed computing. Second, how are particular tasks (e.g., maintaining file consistency, point to point communication, etc.) solved in a mobile setting.

This paper attempts to make contributions to both kinds of questions. On the modeling side, we describe a simple approach to modeling mobile units that has considerable similarity to the standard distributed computing model. This model in turn allows us to transfer results from classical distributed computing to the new mobile setting, leveraging off a large body of existing research in an emerging research area. On the computing side, we describe new algorithms for sending messages to mobile units.

Distributed versus Mobile Computing. A common model of a distributed computing system is a graph where the nodes represent computing components and the edges represent communication. With the exception of faults that can render parts of the network temporarily inoperational, the system is generally static. A mobile computing environment analogous to a cellular telephone system can be similarly modeled with two components. The first is a graph where the nodes represent base stations and the edges wired communication. The second part models the movement of mobile units among base stations as temporary, wireless connections to base stations. The resulting model is a fixed core of static components and a fluid fringe of mobile units. The similarities between the mobile computing model and the distributed computing model, as well as the ease of integrating this model with wired networks, have helped it become dominant in mobile computing research [2, 12].

Yet another model of mobility emerges from the study of code and data moving through a network of hosts [9, 20, 8]. In this case, the mobile components, commonly referred to as mobile agents, move entirely within the network, migrating by explicit message passing between hosts. We suggest a slight modification of this model in order to encompass both physical and logical mobility, thus moving the mobility model closer to the traditional distributed computing model. The basic idea is to treat mobile units as roving messages that preserve their identity as they travel across the network. For a cellular mobile system, this means that while a mobile unit is within a cell, it is modeled as a message residing at a node. When moving to a new cell, the handover protocol is modeled as the traversal of a channel between two nodes.

Algorithm Development. Our interest in this model rests with *its ability to facilitate the develop-*

ment of algorithms in mobile computing based on established algorithms of traditional distributed computing. To illustrate this point, this paper shows how snapshot algorithms can be adapted for unicast and multicast message delivery and how the idea of diffusing computations can be adapted to track and deliver messages to mobile units.

In the presentation, 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 and describes prior work in the area. 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. Section 4 outlines the diffusing computation approach to tracking mobile units. Section 5 outlines adaptations that make the approach viable in a model similar to the cellular telephone system. Finally, Section 6 concludes the paper.

2 Message Delivery

While disconnected operation or working in isolation is a practical use of mobile units [13], many applications require units to communicate with one another while on the move, exchanging voice and/or data. Thus a fundamental problem in mobile computing is the delivery of a message from a source to a mobile unit. In this section we discuss previous work on message passing in mobile environments, define our model of the mobile environment, and formally define the problem of message delivery.

2.1 Related Work

Standard solutions to message delivery to mobile units fall into two categories: *tracking* and *search*. Fundamentally the first involves knowing the current location of the mobile unit in either a centralized or distributed manner while the second maintains no such information and instead searches for the mobile unit in order to deliver a message. Both styles are applicable depending on the mobility scenario. For example, tracking mechanisms are most effective in systems with low or slow mobility and high traffic levels, while systems with high or fast mobility and moderate traffic are more amenable to search solutions. This paper considers solutions to both.

Most standard forms of message delivery rely on tracking. For example, in cellular systems, as a phone involved in an active session moves into an adjacent cell and detects a stronger signal from the new cellular tower, a handover is requested [10, 23]. The cellular system constantly keeps track of the association between phones and towers to forward voice packets to users. In Mobile IP [19], packet delivery is accomplished by the mobile unit registering its new location with its home agent, and having the home agent forward any packets for that mobile unit to the registered location. Other approaches propose changing the routers to adapt to the movement of the mobile units, e.g., intercepting packets en route to the home agent and directing them toward the mobile unit itself [18]. Such approaches involve fundamental changes to the routers and are less well accepted than Mobile IP.

One disadvantage to tracking arises if the mobile unit moves quickly from one base station to another. Each time the unit changes its point of attachment to the network, a tracking system must send update messages, even if the mobile unit is not actively receiving any messages. In fact, the transmission overhead of tracking information scales poorly with the speed of movement.

In search solutions, because the location of the mobile unit is not kept anywhere in the system, in order to deliver a message, the sender must either broadcast a search request to locate the mobile unit then forward the message to the resulting location, or the sender can simply broadcast a copy of the message. The first mechanism has been suggested for mobile ad hoc environments where there is no infrastructure along which to route packets [4]. This approach takes advantage of the natural broadcasting nature of wireless radio communication to send a message to all neighboring mobile units within range. This same style of route discovery is also useful in base station environments with moderate movement of mobile units 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, a search strategy can take advantage of the inherent organization of the Internet into domains and subnets to reduce the scope of the search.

While the *unicast* problem of delivering a message to a *single* recipient is important, *multicast* has also received attention. For example, multicast support through the MBONE has become a standard part of the Internet [7] and is finding use for audio and video conferencing [14, 11]) and video distribution. Additionally, the Mobile IP specification addresses the issues of enabling a mobile unit to function as either a sender or a receiver for multicast messages [19]. In this paper, we show how our algorithms can be adapted from unicast to multicast delivery with minimal effort.

2.2 Mobile Environment

We address the delivery of announcements within a network of fixed mobile support centers (MSCs) and radio base stations (RBSs). For simplicity, we assume each MSC controls only one RBS, and all neighboring MSCs have a fixed communication channel between them. This channel is used by both messages and mobile units.

In Figure 1 each cell represents an (MSC, RBS) pair, and the MSCs of all neighboring cells are connected by a fixed network link. 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 virtual channels between the physically adjacent cells. For simplicity, we also ignore the MSC/RBS destination. We return to both of these in Section 5, providing details of implementing virtual channels

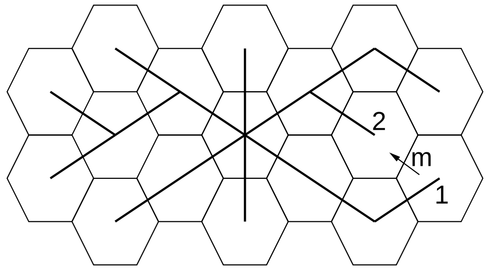


Figure 1: Cell based broadcast based on spanning tree. Each cell is a base station and adjacent cells can communicate on a wired channel.

and allowing multiple RBSs per MSC.

2.3 Model and Problem Definitions

As described in the introduction, the model we consider is one where the infrastructure of the mobile environment is viewed as a graph of nodes and channels, and both the mobile units and the data messages are represented as messages that travel through the network. More specifically, mobile units are viewed as *persistent* messages while data messages disappear from the system after delivery. For all temporary messages, in order to avoid confusion in terminology between control and data traffic, from this point forward, we use the term *announcement* to refer specifically to data messages while a *message* can be either data or control. A mobile unit can send and receive messages and announcements 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. When a mobile unit is on a channel, it is viewed as being temporarily disconnected from the network and unable to communicate.

Since we no longer differentiate between communication lines and physical movement, it is reasonable to question what happens when messages (both announcements and control 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. This appears to require that mobile units travel through space and reconnect to the next support center as fast as messages can be transmitted across a network channel. The FIFO behavior, however, can be realized by integrating the handover protocol with message passing. Essentially, in the cellular model, a mobile unit moves directly between cells; however, in the graph representation, the mobile must move onto a channel before arriving at the new cell. This is a natural assumption when the details of the handover are considered, the details of which are expanded in Section 5. Two other assumptions we make are that a node can deliver any announcements before the mobile unit moves to a new base station and that the network is connected (i.e., there is always some path to deliver the announcement to its destination mobile unit no matter which node it is located at). Finally, we assume bidirectional channels.

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

Because mobile units do not communicate directly with one another, the network must provide the mechanism to transmit the announcement. The original announcement is assumed to be in the local memory of some processing node, presumably left there by the mobile unit that is the source of the announcement. Since a mobile unit is not required to visit all nodes to gather its announcements, the announcement 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 and are the focus of the remainder of this paper.

3 Broadcast Search

Our first approach to announcement delivery takes a broadcast search approach, meaning that a message is broadcast throughout the network in search of the mobile unit. Although this seems like a simple method, applying it directly is not trivial due to the movement of the mobile unit during the broadcast. For example, it is possible for the mobile unit to move one step ahead of the broadcast and eventually pass the announcement in the opposite direction. This problem can be solved by storing the announcement at all nodes for an indefinite period, however Internet routers have neither the storage capability nor the intention to store application announcements. Therefore, announcements 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 while allowing rapid garbage collection in time proportional to one round trip delay on a single link.

3.1 Motivation

A straightforward broadcasting scheme designed for our model of mobility is to construct a spanning tree over the MSCs and send the announcement along this tree. In Figure 1, such a spanning tree is indicated by the solid lines. A disadvantage of this scheme is that a mobile unit may move and not receive the announcement. For example, consider a mobile unit located at cell 1, near the border of cell 2. Suppose the broadcast of an announcement begins at the center-most cell. Following the proposed spanning tree broadcast scheme, the MSC in the initiating cell broadcasts the announcement locally, next the announcement is forwarded on the outgoing links of the spanning tree. After successfully sending the announcement, the initiator deletes its copy of the announcement, minimizing the storage time. The MSCs downstream behave in a similar manner, broadcasting locally, forwarding the announcement to their children, and finally deleting their copy of the announcement.

If the mobile unit does not move away from cell 1, it will receive a copy of the announcement when it is broadcast by MSC_1. However, when the mobile unit is on the border between cell 1 and cell 2, it is possible for a handover to be initiated and for the mobile unit to lose contact with MSC_1 and pick up communication with MSC_2. If this handover occurs after MSC_2 deletes its copy of the announcement and before MSC_1 broadcasts its copy of the announcement, the mobile unit will not receive the announcement even though it was connected to the network during the entire broadcast lifetime of the announcement. Although in reality messages travel through the network much faster than a mobile unit can travel through space, because a handover requires very little time to complete, and the length of the path along the spanning tree could take longer to traverse than for the handover to complete, it is reasonable for a simple broadcast mechanism such as this to fail.

3.2 From Distributed Snapshot Algorithms to Announcement Delivery

To guarantee delivery in any circumstance, we propose an alternative broadcast algorithm that is based on the classical notion of distributed snapshots. Before addressing announcement delivery, we first note the general properties of snapshot algorithms especially those important in announcement delivery.

The goal of a snapshot algorithm is to provide a consistent view of the state of a network of nodes and channels. The state consists of the process variables, and any messages in transit among the nodes. A simple snapshot algorithm would freeze the computation until all messages are out of the channels, record the state of the processors (including outgoing message queues), then restart the computation. Although this is an impractical solution in most distributed settings, it provides the intuition behind a snapshot algorithm, in particular that the consistent global state is constructed by combining the local snapshots from the various processors. In general, a snapshot is started by a single processor and control messages are passed to neighboring nodes informing them that a snapshot is in progress thereby initiating local snapshots. The main property of snapshots that we exploit is that every message appears exactly once in the recorded snapshot state.

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 described here adapt them to perform announcement delivery in the dynamic, mobile environment. To move from the network of nodes and channels into the mobile computing environment, we return to the mobility model for the cellular structure of mobile support centers and radio base stations. As described earlier, these components and the wires connecting them

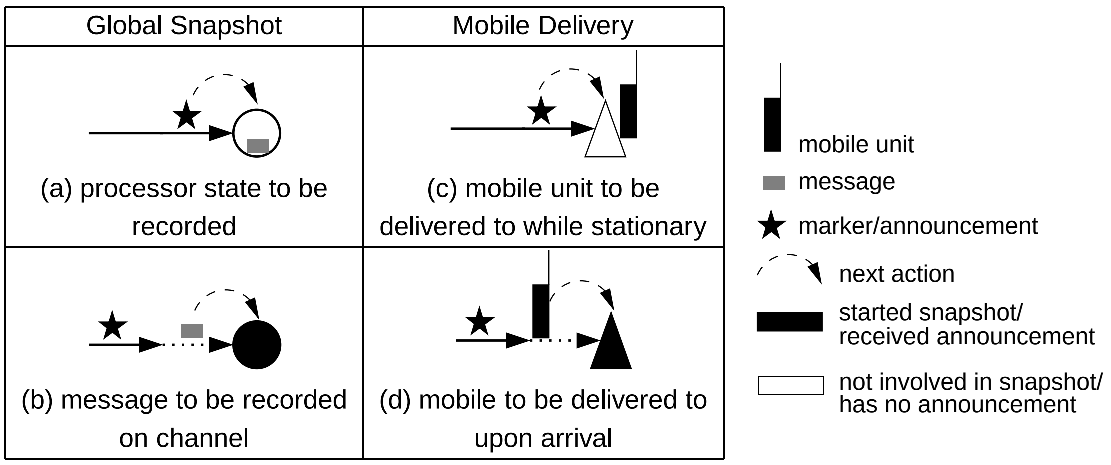

Global Snapshot
Mobile Delivery
(a) processor state to be recorded
(c) mobile unit to be delivered to while stationary
(b) message to be recorded on channel
(d) mobile to be delivered to upon arrival
mobile unit
message
marker/announcement
next action
started snapshot/ received announcement
not involved in snapshot/ has no announcement

Figure 2: Translation of concepts from global snapshots into mobile delivery. The curved arrow shows the processing of an element from a channel while the text describes the action triggered by such movement.

map directly to the network graph of standard distributed computing. The mobile units are simply represented as persistent messages in the distributed environment, meaning they are always somewhere in the system, either at a node (when in communication with a base station) or on a channel (during a handover).

At this point, we have a structure on which to run the snapshot. We note that because the mobile unit is a message and the snapshot records the location of messages, the global snapshot of the mobile system will show the location of the mobile unit. Therefore, one option is to simply deliver the announcement directly to this location; however, it is possible (and likely in systems with rapid movement) that the mobile unit will move between the time its position is recorded and when the announcement arrives at the recorded position. Therefore, we alter the snapshot recording to deliver the announcement by augmenting the control messages with a copy of the announcement itself and changing the recording of messages into the delivering of announcements. We further note that the global state of the system is no longer important for delivery, so no system state information is collected.

3.3 Snapshot Delivery Algorithm

Throughout this section, we use the Chandy-Lamport snapshot algorithm [5] and show its adaptation to announcement delivery. In making the transition to the mobile environment, we carry the restrictions of the original distributed algorithm, and clarify certain characteristics of the mobile model moving from the cell structure to the graph setting.

In the Chandy-Lamport algorithm, it is possible for the snapshot to be initiated at more than one location in the graph, however, we assume that the announcement will be located initially at one point in the network, therefore the snapshot will originate from a single MSC. The Chandy-

Lamport algorithm consists of two main localized actions to collect the local snapshot: the processing of the control messages (markers) and the arrival of the messages to be recorded.

- The *marker arrival rule* states that when a marker arrives at a node not involved in a snapshot, the node begins its local snapshot by recording the processor state, then sends the marker on all outgoing channels (Figure 2a). In the mobile environment, this is analogous to the announcement arriving at a node. If the mobile unit is present, it will receive the announcement, otherwise the node will remain in the local snapshot state and will store the copy of the announcement until the local snapshot is complete. The local snapshot is complete when the marker/announcement has arrived from all incoming channels.
- The *message arrival rule* states that if the message arrives at a node from channel C before the marker arrives on channel C, and the node is in the middle of the local snapshot, the message is to be recorded as on the channel during the snapshot (Figure 2b). In the mobile setting, this condition is the arrival of the mobile unit at an MSC that is storing a copy of the announcement. Therefore, the arrival of the mobile unit triggers the transmission of the announcement to the mobile.

We capture these actions in I/O Automata-like pseudo code shown in Figure 3. In addition to the announcement arrival and mobile arrival, we also include statements to terminate the local snapshot (cleaning up the state) and to allow the mobile unit to move within the network. Channels are assumed to be FIFO, and hold both mobile units and all messages.

We assume the system is initialized with the location of the mobile unit (*MobileAt*) and a single announcement copy at some node (*AnnAt*). Channels are assumed to be empty. We introduce one state variable quantified over the channels (*flushed*) that is used in identifying when the local snapshot is complete. Basically, a flushed channel has received a marker and when all incoming channels have received a marker, the local snapshot is complete.

The actions of Figure 3 describe the local node state transitions that are sufficient for message delivery. No global information is maintained. A node will be in one of three states: not yet aware of the snapshot (*unnotified*), taking a local snapshot (*notified*), and finished with the local snapshot (*finished*). In Figure 4, these states are represented by white, grey, and black respectively. All nodes (except the node where the announcement originates) are initially unnotified. An unnotified node such as E will eventually receive an announcement along one of its incoming channels (ANNARRIVES) (such as (B, E)). This action causes it to transition to the notified state, delivering the announcement if possible, storing a copy of the announcement, marking the channel the announcement arrived on as flushed, and sending announcement copies on all outgoing channels.

Once a channel is flushed, if the mobile unit arrives on that node, it is guaranteed to have seen the announcement at some other node (to have been recorded in some other local snapshot). Therefore, to avoid multiple delivery, if the mobile arrives on a flushed channel, delivery is not repeated (MOBILEARRIVES). If the announcement arrives at a notified node such as A

```
State
flushed_{A,B}   boolean, true if announcement traversed the link from A to B; initially
                false everywhere
AnnAt_A         boolean, true if announcement stored at A; initially true only where
                announcement starts
MobileAt_A      boolean, true if mobile unit at A; initially true only where mobile located

Actions
ANNARRIVES_A(B)   ;arrival at A from B
    Effect:
        flushed_{B,A}:=TRUE
        if ¬AnnAt_A
            send ann. on all outgoing channels
            AnnAt_A:=TRUE   ;save ann.
            if MobileAt_A
                deliver announcement
            endif
        endif

MOBILEARRIVES_A(B)   ;arrival at A from B
    Effect:
        MobileAt_A:=TRUE
        if ¬flushed_{B,A} and AnnAt_A
            deliver announcement
        endif

MOBILELEAVES_A(B)   ;leaves from A to B
    Preconditions:
        MobileAt_A and channel (A,B) exists
    Effect:
        MobileAt_A:=FALSE
        mobile unit moves onto (A,B)

CLEANUP_A   ;A finishes local snapshot
    Preconditions:
        Forall neighbors X, flushed_{X,A}=TRUE
    Effect:
        AnnAt_A:=FALSE   ;delete ann.
        Forall neighbors X, flushed_{X,A} :=FALSE
```

Figure 3: Snapshot Delivery Code

(ANNARRIVES), the channel it arrives on will be marked as flushed, but since the announcement is already stored, no additional copy is made. When all incoming channels have been flushed (as in B), the node's local snapshot is complete and the local state (including the flushed status of the channels and the stored announcement) are deleted (CLEANUP).

The final action, MOBILELEAVES , models the movement of a mobile away from a node. The mobile is simply placed on the channel and the state variables updated to reflect this change. This models random mobile unit movement. If a particular movement pattern is desired, it can be added to this action.

3.4 Properties

Because our announcement delivery algorithm 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. The three primary properties proven for the Chandy-Lamport distributed snapshot are: (1) there is no residual storage in the system at some point after the algorithm begins execution, (2) every message is recorded once, and (3) no message is recorded

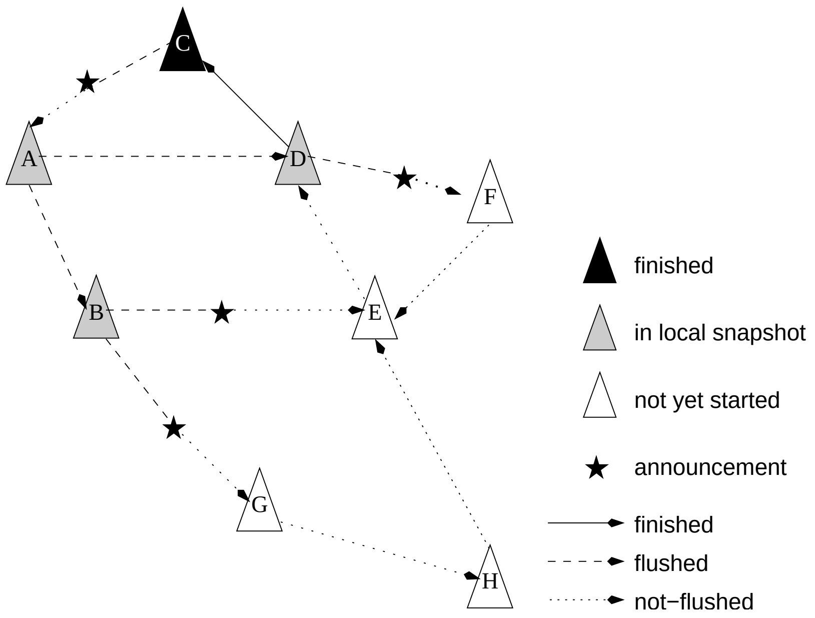


Figure 4: Phases of the nodes and channels during an execution of the snapshot delivery algorithm. Each triangle represents a base station. The mobile unit could be anywhere in the network.

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 announcement is delivered to the intended recipient, and (3) the announcement is delivered only once. In this section, we present a reduction style proof outline moving from the distributed snapshot properties to the mobile announcement delivery properties stated above.

To more explicitly show the relationship between the snapshot algorithm and announcement delivery, we provide the outline of a reduction proof from the Chandy-Lamport distributed algorithm to the adapted snapshot delivery algorithm, showing the mapping between the actions (such as marker arrival and announcement arrival) and the system variables (such as the marker and announcement).

In the Chandy-Lamport algorithm, a processor begins its local snapshot when it receives the first marker. When this occurs, the marker is sent on all outgoing channels and the state of the processor is recorded. If there are any messages at the node, they are recorded as part of the processor state (Figure 2a). If the node has already started its local snapshot when a message arrives along a channel (that the node has not seen the marker on), the message is recorded as being on the channel (Figure 2b). Recording continues until a marker is received on all incoming links.

We translate these actions directly to the mobile environment. The announcement corresponds to the marker, and the mobile unit corresponds to a message in the Chandy-Lamport algorithm. When an MSC receives the announcement for the first time, it sends copies on all outgoing channels and attempts delivery to any mobile unit present. If the mobile unit is at the MSC, it will receive the announcement (Figure 2c).

Just as a node continues recording until it has received the marker on all links in order to record messages on channels, the delivery algorithm *will keep a copy of the announcement until it receives a copy of the announcement from all neighbors* in order to deliver to a mobile unit in transit between base stations. Intuitively, this prevents the mobile unit from hopping from node to node eluding the announcement. Thus if the mobile unit arrives prior to the announcement on a channel, the MSC delivers the data as soon as the handover is complete (Figure 2d). This is possible because the MSC stores a local copy that arrived on another channel.

3.5 Extensions

One of the strengths of our approach to algorithm development is that it rapidly produces an algorithm in the mobile environment that can be easily extended. In this section we discuss several possible extensions including delivering multiple announcements simultaneously, delivering to rapidly moving mobile units, performing route discovery, multicasting an announcement, and working within the mobile agent environment.

Multiple announcement deliveries. To deliver multiple announcements simultaneously using snapshots, we can run several copies of the algorithm in parallel. This is analogous to having each MSC both index and store the incoming announcements and maintain separate channel status for each announcement 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 announcement in the system, as well as maintain channel status with respect to each announcement. 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 announcements, the nature of the network will determine how long an announcement is stored at the MSC.

Rapidly moving mobile units. Another advantage of 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 location 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.

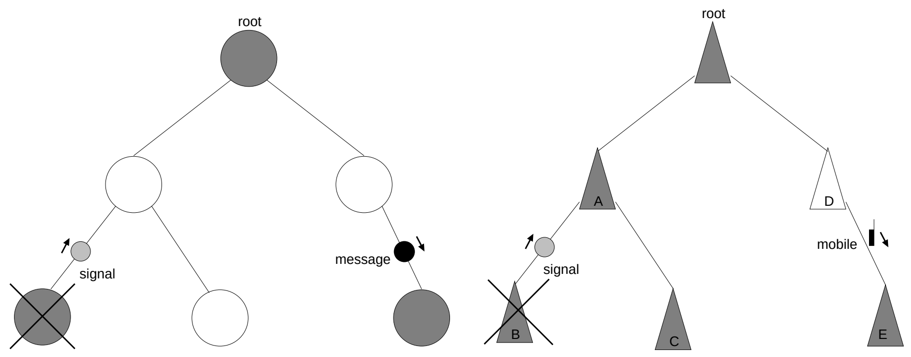
Route discovery. In more moderately changing environments, the overhead of sending the announcement to every node may be excessive. In these situations, the snapshot delivery algorithm can be modified to perform route discovery. When a source mobile unit S located at MSC$_S$ wishes to communicate with a destination mobile unit D located at MSC$_D$, S sends a *discovery* message using snapshot delivery. When this message is received by D, a *discoveryReply* is sent back to MSC$_S$, identifying D's location. Subsequent messages from S to D are sent directly to MSC$_D$. When a message fails to be delivered, the discovery process is repeated using snapshot delivery for the query.

Multicast. Another area of research in the mobility community is multicast, including some work on reliable multicast [1] but only under the assumption that the set of recipients is known. Our algorithm can trivially be extended 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 processing of the snapshot algorithm and by only changing the destination address from a unicast mobile unit identifier to a multicast address, it can be shown that every host accepting announcements on that address will receive the announcement. The reason for this can be found by looking back at the traditional distributed snapshots. In a snapshot, every message in the system is recorded in exactly one local snapshot. In our modified algorithm, *delivery* replaces *recording* and mobile units replace messages. Therefore, every mobile unit will be delivered to exactly one time. Although this description is concise, the importance of it should not be lost in its simplicity.

Mobile agents. Thus far we have only considered physical movement of mobile units, but another possible application that is characterized by rapid mobile movement is mobile agents where it is not a physical component that moves, but rather program code and data moving through the fixed network. Rather than connecting to a base station through a wireless mechanism, these mobile agents actually execute at a foreign host. They have the ability to move rapidly from one host to another and may not register each new location with a home. Therefore, delivering a message to a mobile code agent becomes an interesting application area in which rapid movement is not only feasible, but is the common case. The interested reader can find more details on applying snapshots in logical mobile environments in [16].

4 Tracking for Delivery

Now we turn our focus toward an approach to message delivery based on tracking the location of the mobile unit as it moves through the network. Unlike Mobile IP tracking, our approach does not require location updates to be sent to the home node each time. This section describes a tracking and delivery approach that comes from applying our algorithm development technique to the Dijkstra-Scholten diffusing computation/termination detection algorithm. After outlining this algorithm, we present another algorithm that is not directly based on diffusing computations, but was inspired by our previous investigation. The details of this work are available in [17].



(a) Distributed environment (b) Mobile environment

Figure 5: Dijkstra-Scholten trees of diffusing computations. Shaded nodes are idle, white nodes are active. (a) Applied to a standard distributed environment. (b) Applied to a mobile environment, tracking a single mobile unit. Note there is only one active node, and it is the node the mobile unit just left. A possible path of the mobile unit to build this tree would be: root, A, B, A, C, A, root, D, E.

4.1 From Diffusing Computations to Mobile Unit Tracking

Diffusing computations have the property that the computation initiates at a single root node while all other nodes are *idle*. The computation spreads to other nodes as messages are sent from *active* nodes. Dijkstra and Scholten [6] describe an algorithm for detecting termination of such computations in which the basic idea is to maintain a spanning tree that includes all active nodes, as shown in Figure 5a. A message sent from an active node to an idle node (*message* in Figure 5a) adds the latter to the tree as a child of the former. Messages sent among tree nodes have no effect on the structure but may activate idle nodes still in the tree. An idle leaf node can leave the tree at any time by notifying its parent (*signal* in Figure 5a). Termination is detected when an idle root is all that remains in the tree.

By applying our algorithm development technique, we adapt this tree maintenance algorithm to track the movement of a mobile unit as it travels among base stations. We define a node to be *active* when the mobile unit is present (or has started the handover process and is modeled on the channel), and therefore when the mobile unit arrives at a node, if that node is not already part of the tree, it is added. In Figure 5b, this corresponds to adding E as an active node when the mobile unit arrives and changing the status of node D to idle. Because all active nodes are in the tree of the diffusing computation, the Dijkstra-Scholten algorithm guarantees that the mobile unit will always be at a node in the tree (or on a channel leaving from a node in the tree). In other words, the tree of the diffusing computation defines a sub-region of the network where the mobile unit has recently traveled. As the mobile unit doubles back on its path, the node it arrives at transitions back to active and the node it departed becomes a leaf node that is cleaned up in the same way idle leaf nodes are removed in the original Dijkstra-Scholten algorithm (sending a signal message).

This tracking of a mobile unit by identification of a region containing the mobile unit is only part of our goal. Reliable message, or announcement, delivery is the other component that we achieve by designing an algorithm delivery algorithm that works on top of the diffusing computation tree. Our algorithm works by placing the announcement at the root of the tree and spreading it down the tree until the mobile unit (or a leaf node) is reached. To guarantee delivery to a mobile unit that is moving during the announcement propagation, we temporarily store the announcement at the intermediate nodes, and run a cleanup phase after the message is delivered to remove the extra copies.

By superimposing the delivery actions on top of the graph maintenance, the result is an algorithm that guarantees at least once delivery of an announcement while actively maintaining graph of nodes recently visited by the mobile unit.

It is not necessary for the spanning tree to be pruned as soon as a node becomes an idle leaf. Instead this processing can be delayed until a period of low bandwidth utilization. An application may benefit by allowing the construction of a wide spanning tree within which the mobile units travels, similar to the graph shown in Figure 5b. Tradeoffs include shorter paths from the root to the mobile unit versus an increase in the number of nodes involved in each announcement delivery.

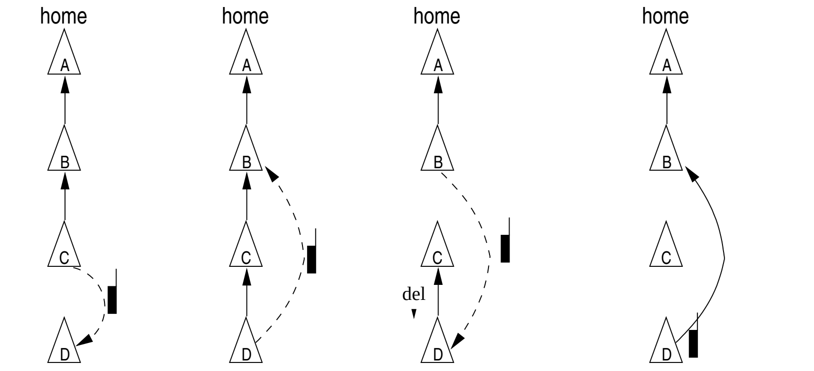
By constructing the graph based on the movement of the mobile unit, the path from the root to the mobile unit may not be optimal. Therefore, a possible extension is to run an optimization protocol to reduce the length of this path. Such an optimization must take into consideration the continued movement of the mobile unit as well as any announcement deliveries in progress. The tradeoff with this approach is between the benefit of a shorter route from the root to the mobile unit and the additional bandwidth and complexity required to run the optimization and simultaneously guarantee the delivery of announcements en route to the mobile unit.

Although in our algorithm only one mobile unit is tracked, the graph maintenance algorithm requires no extensions to track a group of mobile units. The resulting spanning tree can be used for unicast announcement delivery without any modifications and for multicast announcement delivery by changing only the announcement clean up mechanism.

4.2 Extension: Backbone-based Message Delivery

We now introduce a new tracking and delivery algorithm inspired by the previous investigation with diffusing computations. Our goal is to reduce the number of nodes to which the announcement propagates. To accomplish this we note that only the path between the root and mobile unit is necessary for delivery. In the previous approach, although the parts of the tree not on the path from the root to the mobile unit can be eliminated, announcements still propagate unnecessarily down these subtrees before node deletion occurs.

To avoid this, the algorithm presented in this section maintains a graph with only one path



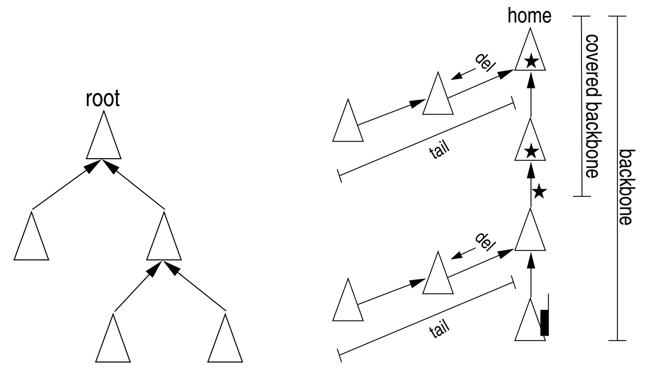
(a) Backbone extended (b) Backbone shortened (c) Tail node added (d) After movement completes

Figure 6: The parent pointers of the backbone change as the mobile moves to (a) a node not in the backbone, (b) a node higher in the backbone, and (c) a tail node. (d) shows the state after all channels have been cleared.

leading away from the root and terminating at the mobile unit. This path is referred to as the *backbone*. Nodes that were once part of the backbone but are no longer on this path between the root and the mobile unit form structures referred to as *tails*. Tails are actively removed from the graph, rather than relying on idle leaf nodes to remove themselves. Maintenance of the backbone requires additional information to be carried by the mobile unit regarding the nodes currently on the backbone, as well as the introduction of a *delete* message to remove tail nodes. The announcement delivery mechanism remains essentially the same as before, but the simpler graph reduces the number of announcement copies stored during delivery.

Intuitively the backbone nodes are the core of the algorithm because they represent the path between the root and the mobile unit that is necessary for announcement delivery. The tail nodes are leftover pieces that were formerly part of the backbone, but the doubling back of the mobile unit to backbone nodes makes these nodes unnecessary for message delivery. If we were not concerned with leaving unnecessary state lying around in the network, we could simply ignore these tail nodes, however for completeness, we include an active mechanism to shrink tails until they disappear. The complexities of the approach lie in properly maintaining the backbone and in cleaning up only tail nodes. Because nodes only have local knowledge, all decisions about dealing with arriving messages and announcements must be based on the information held at the node and carried by the message.

To understand how the backbone is kept independent of the tails, we examine how the graph changes as the mobile unit moves. It is important to note that by the definition of the backbone, the mobile unit is always either at the last node of the backbone, or on a channel leading away from it. Figure 6 shows how the backbone is affected as the mobile unit moves to each of the tree distinct types of nodes: (a) a node that is neither a backbone nor a tail node, (b) a backbone node and (c) a tail node.



(a) Sample diffusing computation

(b) Modified graph showing new structure

Figure 7: By adapting diffusing computations to mobility, we construct a graph reflecting the movement of the mobile. In order to deliver an announcement, the only part of the graph we need is the path from the root to the mobile, the *backbone*. Therefore we adapt the Dijkstra-Scholten algorithm to maintain only this graph segment and delete all the others.

In Figure 6a, the backbone is composed of nodes A, B, and C and the dashed arrow shows the movement of the mobile unit from node C to D where D is not part of the graph. This is the most straightforward case in which the backbone is extended to include D by adding both the child pointer from C to D (not shown) and the parent pointer in the reverse direction (solid arrow in Figure 6b).

In Figure 6b, the mobile moves to a node B, a node already in the backbone and with a non-null parent pointer. It is clear from the figure that the backbone should be shortened to only include A and B without changing any parent pointers, and that C and D should be deleted. To explicitly remove the tail created by C and D, a delete message is sent to the child of B. When C receives the delete from its parent, it will nullify its parent pointer, propagate the delete to its child, and nullify its child pointer.

If at this point the mobile moves from B onto D before the arrival of the delete (See Figure 6c), D still has a parent pointer (C) and we cannot distinguish this case from the previous case (where B also had a non null parent pointer). In the previous case the parent of the node the mobile unit arrived at did not change, but in this case, we wish to have D's parent set to B (the node the mobile unit is arriving from) so that the backbone is correct. To distinguish these two cases, we require the mobile unit to carry a sequence containing the identities of the nodes in the backbone. In the first case where the mobile unit arrives at B, B is in the list of backbone nodes maintained by the mobile unit, therefore B keeps its parent pointer unchanged, but prunes the backbone list to remove C and D. However, when the mobile arrives at D, only A and B are in the backbone list, therefore the parent pointer of D is changed to point to B. But, what happens to the delete message moving from C to D? Because C is no longer D's parent when the delete arrives, it is simply dropped and the backbone is not affected.

The delivery algorithm is then superimposed on top of the generated graph. In the previous section, the announcement propagated from the root down all edges of the tree. In the algorithm of this section, the announcement only propagates down the edges that are part of the backbone. It is still necessary to keep a copy of the message at every node until delivery occurs. Consider a case where the announcement is not stored, and instead simply propagates down the backbone. In Figure 6b, if the announcement were at node C when the mobile unit moved from node D to B, delivery would not occur because the mobile unit moved from a region below propagation to a region above propagation. Therefore, to guarantee delivery, as the announcement propagates down the backbone, a copy is stored at each node until delivery is complete. We refer to the portion of the backbone with an announcement as the *covered backbone*, see Figure 7b.

Delivery can occur either by the mobile unit moving to a location in the covered backbone, or the announcement catching up with the mobile unit at a node. In either case, an acknowledgment is generated and sent via the parent pointers toward the root to clean up the extra announcement copies. If the announcement is delivered when the mobile unit moves on to the covered backbone, a *delete* is generated toward the child and an *acknowledgment* is generated toward the parent. While the *acknowledgment* removes the copies of the announcement on the backbone, the *delete* removes the copies from the tails at the same time the tail nodes are removed from the graph.

Keeping the backbone sequence is a similar methodology to routing protocols passing complete paths to the destination as in BGP [21] to avoid loops. It has been argued that keeping such information in the packet greatly increases its size. However, in our case, the information is being kept by the mobile unit and we assume there is sufficient storage on such a device for this additional information. In a mobile agent system, the path can be trivially shortened by forcing the agent to return to its home node periodically. This is not as reasonable for a physically mobile system, and in the case where the backbone sequence grows beyond a reasonable limit, a secondary, optimization algorithm can be executed to shorten its length.

A simple extension of this algorithm is to allow for multiple concurrent announcement deliveries as in sliding window protocols. The announcements and all associated acknowledgments would have to be marked by sequence numbers so that they do not interfere, but the delivery mechanism uses the same graph. Therefore the rules governing the expansion and shrinking of the graph are not affected but the proofs of garbage collection and acknowledgment delivery are more delicate.

5 Reality Check

When moving from the distributed computing environment to the mobile environment, we made several assumptions about the nature of the network and the behavior of the components in the network. In this section, we reexamine these assumptions, showing why they are reasonable, or how the algorithm can be adapted to make them more reasonable. Specifically, we look at the

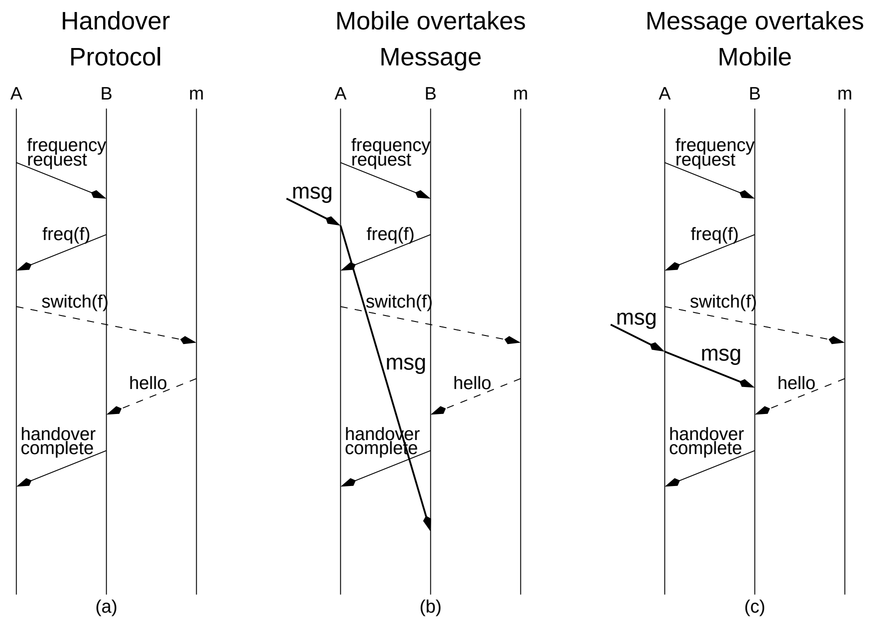


Figure 8: AMPS handover protocol (a) for mobile unit m moving from cell A to cell b. If messages are processed (i.e., broadcast to the mobile) immediately upon receipt, it is possible (b) for the mobile to move faster than the message along the channel, or (c) for the message to move faster than the mobile, thus breaking the FIFO channel property.

issues of non-FIFO channels, multiple RBSs per MSC, base station connectivity, reliable delivery on links, the involvement level of MSCs, and storage requirements.

FIFO Channels. One major issue when using the Chandy-Lamport algorithm is its reliance on FIFO channels. More specifically, in Section 3.2 we modeled both the mobile units and the messages as traveling on the same channel. This seems to be an unreasonable assumption given that mobile units move much more slowly through space than messages move through a fixed network. By looking in more detail at the handover protocol used when a mobile unit changes cells, we show how the FIFO assumption can be broken, and propose a simple mechanism to restore it.

One of the U.S. standards for analog cellular communication is AMPS [22], in which cellular telephones tune to only one frequency at a time. When the signal between the MSC and a mobile unit begins to degrade, the MSC searches for a neighboring MSC with a stronger communication signal indicating the mobile unit is moving into that particular cell. When a frequency is requested, a handover begins. Figure 8a shows the control messages exchanged as a mobile unit, m, moves from cell A to cell B. First the *frequency request* is exchanged between the MSCs. At this point the mobile unit is made aware of the handover by receiving a new frequency from its current MSC, A. After switching to the new frequency, the mobile sends a *hello* on the new frequency, alerting B that the mobile is now listening on the new frequency. Finally, B sends a *handover complete* to A, which releases the old frequency.

By using the AMPS 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 handover until the moment it changes the frequency it is tuned to.

Our primary concern is making the channels FIFO with respect to mobile units and messages (both control messages and announcements). Even if we assume that channels between MSCs are FIFO, reordering is possible because part of the handover takes place over wireless channels that are not synchronized with the wired channels. Specifically we address the two cases of non-FIFO behavior where (1) the mobile overtakes a message and (2) the message overtakes the mobile.

It is important to define the point at which the mobile logically moves onto the channel. We define this to be when communication with A is terminated by the transmission of the *switch* message. Similarly, the mobile moves off of the channel when the wireless transmission of the *hello* message is accepted at the destination cell, B in our example. As can be seen in Figure 8b, it is possible for a message sent on the wired channel before the *switch* message to arrive at the destination after the arrival of the mobile unit, breaking the FIFO ordering. Similarly, a message sent after the *switch* message can move quickly through the channel and arrive at the destination before the mobile (Figure 8c).

We propose a minor change in the protocol in order to involve both the wired and wireless channels in the handover. The only change to the source side (A in this case) is the wire transmission of a special message atomically with the wireless *switch* transmission. We call this message the *virtual mobile unit* (VMU) because it identifies the point on the wired channel at which the mobile leaves the source. All messages sent on the wired channel before the VMU were sent before the mobile unit left, and all messages after the VMU were sent after the mobile unit left. We correspondingly change the behavior of the destination (B in this case) to achieve this desired behavior, in other words to have the virtual mobile unit and the physical mobile unit arrive at the destination at the same time. Therefore, if the *hello* arrives before the VMU , all incoming messages on the wired channel are treated as if the mobile is not present even though communication is possible. Conversely, if the VMU arrives before the *hello*, all messages sent on the wired channel are buffered until the *hello* arrives. When both messages have arrived and have been processed, B continues processing all messages in the order in which they are received. By forcing the receiver to wait for both messages, the wired and wireless channels are synchronized, effectively yielding a single FIFO channel containing both mobile units and messages.

Multiple RBSs per MSC. Until this point we have only allowed one radio base station for each support center, however, in current cellular telephone systems, MSCs manage sets of RBSs. Because the algorithm we presented is intended to be run over the fixed network formed by the MSCs, the handover 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 announcement within the cells supported by a single MSC and maintain the constraints of guaranteed, single delivery. Because the MSC acts as a coordinator of the mobile units present at each of the RBSs, it is feasible to run the snapshot delivery algorithm among the RBSs, allowing it to terminate before any handovers to other MSCs are permitted. This simple solution shows how our snapshot algorithm can be used as a support layer for other algorithms.

Base station connectivity. Another possible concern with the model we presented 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 these physical wires may not exist. To allow our algorithms to function in such a setting, we propose adding virtual channels between adjacent cells and treating such channels the same as the real channels. In the implementation, however, we must be careful to ensure the FIFO nature of this virtual channel.

The same technique can be applied to support a limited form of disconnection. Suppose a mobile unit was likely to disconnect from cell A and at some time later connect to cell B. By adding a virtual channel between A and B and managing the disconnection as a long-lived handover, we can guarantee delivery even if the mobile unit disconnects during the delivery. While this requires additional memory support at the base stations to store the announcements for the duration of this disconnection, such storage is not required at all base stations, making this a reasonable approach for guarantees in the presence of disconnection.

Reliable delivery on links. Our delivery algorithms assume that message delivery across a link is reliable. Most of the Internet uses unreliable links such as 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 acknowledgments for multicast messages as is done, for example, in the intelligent flooding algorithm used in Links State Routing in OSI [24] and OSPF [15]. Another solution is to only provide best-effort service. Since lost messages can lead to deadlock we need to delete an announcement after a timeout even if it is still expected along a channel.

Involvement level of MSCs. For the snapshot algorithm to function, every MSC must be involved to guarantee delivery and termination. In a paper on running distributed computations in a mobile setting [3], 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 when designing distributed algorithms for execution over mobile units, however our goal is not to create a global snapshot containing information about the mobile units, but instead to employ the snapshot technique to a different end, namely announcement delivery. Additionally, the control messages of the snapshot are not processed by the mobile units, but rather by the fixed mobile support centers, and no resources of the mobile unit are expended, except to receive a message.

It is true that in order to guarantee delivery the mobile unit must be present in the system, however, this is a reasonable assumption because by definition there are no means to reach a disconnected mobile unit. It is worth nothing 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 announcement from the system, but without delivery.

Storage requirements. In snapshot delivery, we assume that the MSCs hold a copy of the announcement for delivery to the mobile units for a bounded period of time limited to the duration of the local snapshot. This is more efficient than another proposal [1] in which the announcement is broadcast to all nodes, each of which stores the announcement until notified that delivery has occurred. In our approach, the time for storage is bounded by the speed of network propagation and connectivity of the network. In a system with bi-directional channels, because the local snapshot terminates when the announcement arrives on all incoming channels, the duration of a local snapshot can be as short as one round trip delay between the MSCs. One can argue that it is not the place of the MSCs to be maintaining copies of announcements 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.

6 Conclusions

This paper makes two important contributions. First, it explores a model of mobility in which handovers are abstracted as the traversal of links among the base stations and mobile units, both physical and logical, are treated as persistent messages. The result is a model that unifies wired and cellular, wireless networking and facilitates the transfer of algorithmic knowledge between the two settings. Second, we offer a general methodology for reusing results from distributed computing in the area of mobile computing. Our main contribution is that we suggest not a direct usage of the existing algorithms, a strategy shown to have limited applicability, but a way to capitalize on the intellectual investments made in the field of distributed computing. The examples presented adapt a snapshot algorithm to search for a mobile unit and deliver a message, and use diffusing computations to track the movement of a mobile unit. The ease with which we built these new algorithms provides strong evidence of the efficacy of the general strategy advocated in this paper.

References

[1] A. Acharya and B.R. Badrinath. A framework for delivering multicast messages in networks with mobile hosts. *Journal of Special Topics in Mobile Networks and Applications (MONET)*, 1(2):199–219, October 1996.

[2] B.R. Badrinath, A. Acharya, and T. Imielinski. Structuring distributed algorithms for mobile hosts. In *Proceedings of the Fourteenth International Conference on Distributed Computing Systems*,

pages 21–28, Poznan, Poland, 1994.

[3] B.R. Badrinath, A. Acharya, and T. Imielinski. Designing distributed algorithms for mobile computing networks. *Computer Communications*, 19(4):309–320, April 1996.

[4] J. Broch, D.B. Johnson, and D.A. Maltz. The dynamic source routing protocol for mobile ad hoc networks. Internet Draft, March 1998. IETF Mobile Ad Hoc Networking Working Group.

[5] K.M. Chandy and L. Lamport. Distributed Snapshots: Determining Global States of Distributed Systems. *ACM Transactions on Computer Systems*, 3(1):63–75, February 1985.

[6] E.W. Dijkstra and C. Scholten. Termination detection for diffusing computations. *Information Processing Letters*, 11(1), 1980.

[7] H. Eriksson. Mbone: The multicast backbone. *Communications of the ACM*, 37(8):54–60, 1994.

[8] A. Fuggetta, G.P. Picco, and G. Vigna. Understanding Code Mobility. *IEEE Transactions on Software Engineering*, 24(5):342–361, May 1998.

[9] R. Gray, D. Kotz, S. Nog, D. Rus, and G. George. Mobile agents for mobile computing. Technical Report PCS0TR96-285, Dartmouth College, May 1996.

[10] J. Ioannidis and Jr. G.Q. Maguire. The design and implementation of a mobile internetworking architecture. In *1992 Winter Usenix*, 1993.

[11] V. Jacobson and S. McCanne. Visual audio tool.

[12] D.B. Johnson. Scalable support for transparent mobile host internetworking. In H. Korth and T. Imielinski, editors, *Mobile Computing*, pages 103–128. Kluwer Academic Publishers, 1996.

[13] J.J. Kistler and M. Satyanarayanan. Disconnected operation in the coda file system. *ACM Transactions on Computer Systems*, 10(1):3–25, 1992.

[14] S. McCanne and V. Jacobson. vic: A flexible framework for packet video. In *ACM Multimedia '95*, pages 511–522, San Francisco, CA, USA, 1995.

[15] J. Moy. OSPF version 2. Internet draft, Internet Engineering Task Force, March 1994 1994.

[16] A.L. Murphy and G.P. Picco. Reliable communication for highly mobile agents. *Journal of Autonomous Agents and Multi-Agent Systems, Special issue on Mobile Agents*, 5(1):81–100, March 2002.

[17] A.L. Murphy, G.-C. Roman, and G. Varghese. Tracking mobile units for dependable message delivery. *IEEE Transactions on Software Engineering*, May 2002.

[18] A. Myles and D. Skellern. Comparing four IP based mobile host protocols. *Computer Networks and ISDN Systems*, 26(3):349–355, 1993.

[19] C.E. Perkins. IP mobility support. Technical Report RFC 2002, IETF Network Working Group, October 1996.

[20] M. Ranganathan, A. Acharya, S. Sharma, and J. Saltz. Network-aware mobile programs. Technical Report CS-TR-3659, University of Maryland, College Park, 1997.

[21] Y. Rekhter and T. Li. A Border Gateway Protocol 4 (BGP-4). *RFC 1771*, March 1995.

[22] M. Steenstrup. *Routing in Communication Networks*, chapter 10. Prentice-Hall, 1995.

[23] F. Teraoka, Y. Yokore, and M. Tokoro. A network architecture providing host migration transparency. *ACM SIGCOMM Computer Communication Review (SIGCOMM'91)*, 21(4):209–220, September 1991.

[24] H. Zimmerman. OSI reference model – The ISO model of architecture for open systems interconnection. *IEEE Transactions on Communication*, 28:425–432, 1980.