A Distributed Snapshot Algorithm Adapted for Message Delivery to Mobile Units

A.L. Murphy, G.-C. Roman, G. Varghese
Department of Computer Science
Campus Box 1045, Bryan 509
Washington University in St. Louis
One Brookings Drive
St. Louis, MO 63130-4899
Phone: 314-935-7537, Fax: 314-935-7302
alm, roman, varghese@cs.wustl.edu

January 14, 1999

Abstract

Mobile computing is emerging as a novel paradigm with its own characteristic problems, models, and algorithms. Much effort is being expended on integrating mobile units with fixed networks, providing bridges to connect wireless to wired. The result is a fixed core of static nodes and a fluid fringe of mobile units, an overall system similar to the cellular telephone network. With this graph as a foundation, our approach to mobility treats the mobile units themselves a persistent messages moving through the network. This model allows algorithms from traditional distributed computing to be directly implemented in the mobile environment. However, it has been noted that because of the unique properties of mobility such as limited bandwidth and disconnection, it is not practical to do such a direct translation. This paper presents a fundamentally different direction. Previous work has focused on moving distributed algorithms into the mobile domain to solve the same types of problems, while we adapt distributed algorithms to solve issues unique to the mobile environment. Specifically, we propose an algorithm based on distributed snapshots which accomplishes guaranteed unicast and broadcast message delivery to rapidly moving mobile units.

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 [11]. 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 a new algorithm 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 [2, 10]. 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 [7, 16]. 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 handover 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.

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 a 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 handover 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 [6] and is finding wide use for conferencing (e.g., tools like VIC and VAT [12, 9]) 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 [8] and Mobile IP [15]) 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', the phone detects a stronger signal from C' and requests a handover from C to C'. After the handover, C' is now responsible for forwarding voice packets to the phone, and the cellular system keeps track of this association between C' 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. 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, e.g., intercepting packets en route to the home agent and directing them toward the mobile unit itself. [14] However, such approaches involve fundamental changes to the routers.

One disadvantage to tracking arises 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 high traffic and 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.

Although both search and tracking have merits based on expected modes of mobility, the focus of this paper is on search. In particular, we adapt the Chandy-Lamport [4] snapshot algorithm to message delivery. 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. 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. In the following section, we consider adaptations that make the approach viable in a model similar to the cellular telephone system. Finally, Section 6 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 where the mobile infrastructure is viewed as a graph of nodes and channels, both the mobile units and the data messages are represented as messages that travel through the network. Mobile units are viewed as persistent messages while data messages are eventually removed from the system after delivery. In order to avoid confusion in terminology between control and data traffic, from this point forward, we will use the term *announcement* to refer specifically to the data message being delivered while a *message* can be either data or control. 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. When 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 (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 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 handover protocol with message passing. The details of this will be expanded in Section 4. The other assumptions we make are that a node can deliver any announcements before the mobile unit moves on (i.e., movement is slower than communication) and that the network is fully 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).

The announcement delivery problem can now be formulated as follows: Given a fully connected network with FIFO channels and guaranteed message delivery, an announcement located at one of its nodes, and a mobile unit for 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 is another concern.

Because mobile units do not communicate directly with one another, the network must provide a 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 which 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 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 is a simple methodology, applying it directly is not trivial due to the movement of the mobile unit during the broadcast. 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 nor the intention of storing 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 (in practice, with unreliable links, delivery would be at most once) while allowing rapid garbage collection in time proportional to one round trip delay on a single link.

3.1 Motivation

We address the delivery of announcements within 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 neighboring MSCs have a fixed communication channel between them. This channel will pass the messages and the mobile units between the connected MSCs.

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 symbolic channels between the physically adjacent cells. The details of implementing this vary with the solution, and will be addressed in detail in a later section.

A straightforward broadcasting scheme in this model 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 miss the

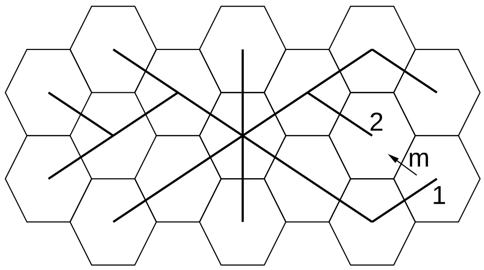

Figure 1: Cell based broadcast based on spanning tree.

announcement. To see this, suppose a mobile unit is located at cell 1, near the border of cell 2. Suppose an announcement is initiated and the broadcast begins. Following the proposed spanning tree broadcast scheme, the MSC in the initiating cell has its RBS broadcast the announcement locally, then 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 in order to minimize 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 out of cell 1, it will receive a copy of the announcement when it is broadcast by RBS_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 RBS_1 and pick up communication with RBS_2. If this handover occurs after RBS_2 deletes its copy of the announcement and before RBS_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 which is based on the classical notion of distributed snapshots. Before addressing announcement delivery, we first note the general properties of snapshot algorithms and those that will be 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, and initiating local snapshots. The main property of snapshots that we will exploit is that every message will appear 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 which we describe in this paper 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 cellular structure of mobile support centers and radio base stations. These components and the wires connecting them 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, and 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 rapidly moving mobile unit) 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 delivery of the message by augmenting the control messages with the announcement 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 needs to be collected.

3.3 Snapshot Delivery Algorithm

Throughout this section, we will use the Chandy-Lamport snapshot algorithm [4] 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. First, the Chandy-Lamport algorithm relies on unidirectional, FIFO channels. To model bidirectional channels, we place two unidirectional channels in opposite directions; however, to handle the FIFO assumption, extensions to the handover protocol are necessary. These will be presented in Section 5. 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. Finally, the model that we presented has several RBSs per MSC. To simplify the model, we will eliminate the RBS, but will return to it in Section 5.

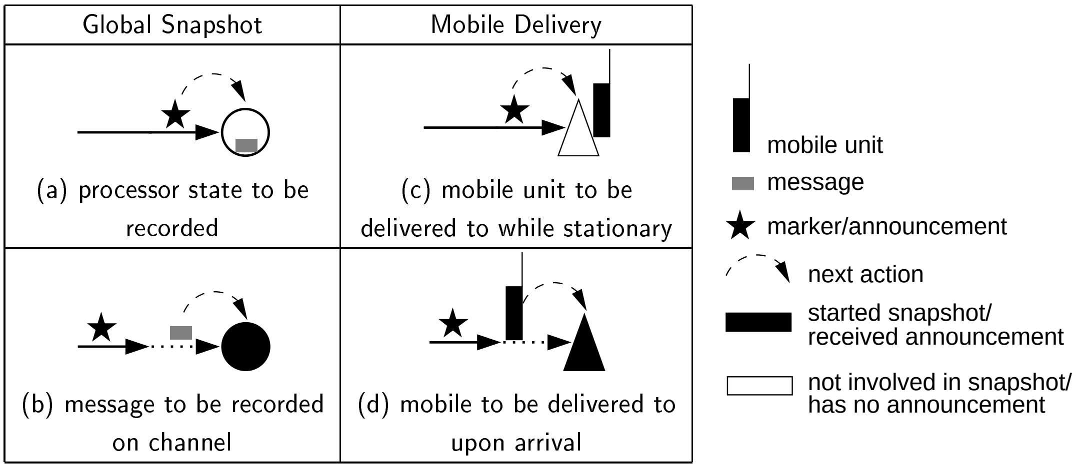

Figure 2: Translation of concepts from global snapshots into mobile delivery. The curved arrow shows the processing of an element from a channel while the text describes the action triggered by such movement.

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 which is storing a copy of the announcement. Therefore, the arrival of the mobile triggers the transmission of the announcement to the mobile.

We capture these actions in an I/O Automata-like pseudo code program 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 from a node to a channel. The 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*, MSC A in Figure 4) 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*) which is used to

<u>State</u>

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

<u>Actions</u>

```
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

identify when the local snapshot is complete. Basically a flushed channel has received a marker. As noted previously, when all incoming channels have received a marker, the local snapshot is complete.

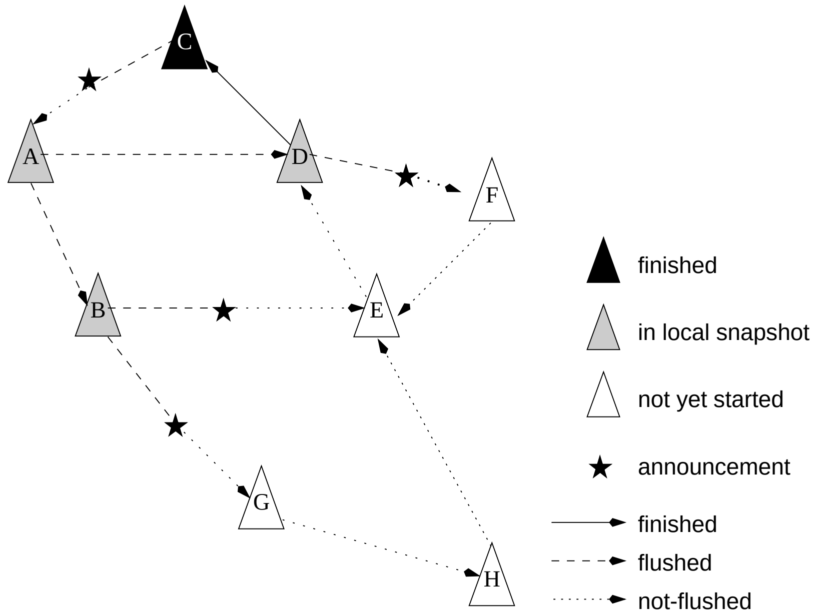

Figure 4: Phases of delivery

These actions describe the local node state transitions which are sufficient for message delivery. No global information must be 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) start 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 (MobileArrives), 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) can be 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 effectively achieves random mobile unit movement, however if a particular movement pattern is desired, it can be added to this action.

3.4 Properties

In the previous section we presented a global snapshot algorithm modified to perform announcement 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. 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 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 first the intuition behind why these properties hold, followed by a reduction style proof outline moving from the distributed snapshot properties to the mobile announcement delivery properties stated above.

To show that eventually all information concerning the announcement is removed from the system, we must show that the program will eventually reach a state where there are no announcements at any nodes, there are no announcements in any channels, and all channels are 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 announcement on each of its channels. In Figure 4, this is analogous to each node eventually becoming notified. Once this occurs, each of the channels is flushed, and the cleanup action is enabled, clearing the flushed variable and removing the copies of the announcement at each node. In the figure, the node will transition to finished. There will be no announcements in channels because once an announcement has passed through a channel, it is not possible for another announcement to be sent down that channel. Therefore, we have shown that eventually the system is clear.

Next we must show that the announcement is eventually delivered. First we note (from the previous paragraph) that eventually all nodes receive the announcement. 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 announcement on this channel. In Figure 4, an unnotified mobile unit could be located at a white node, or on a dotted channel, but in this case, we are describing a graph in which all nodes are already notified, therefore the mobile unit must be located on an unflushed channel segment in front of an announcement. If the mobile unit was behind the announcement, it would have already received the announcement. Therefore, because the mobile unit arrives on an unflushed channel, and by the assumption that all nodes have received a copy of the announcement, the node it is heading toward must have a copy of the announcement. Delivery will occur when the mobile unit moves onto the node.

Having shown the announcement will be delivered, it remains to be shown that the announcement 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. Delivery can occur in two situations: (1) by the mobile being on an unnotified node when the announcement arrives, and (2) by the mobile arriving (along an unflushed channel) to a notified node. To show each of these cases will not arise after delivery, we characterize a region of the graph called *will-be-notified* and define it as the set of all unnotified nodes, and the channels or channel segments which have not had a copy of the announcement pass through them. In Figure 4, this region corresponds to the white nodes (unnotified) and dotted channels (unflushed). Having defined this region, and noted that both of the above delivery cases occur when the mobile is in this region, it is sufficient to show that after delivery the mobile unit will not be in the *will-be-notified* region. To show this, 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 after delivery on such a path, there must always be an announcement. Therefore, in order for the mobile unit to move back into the *will-be-notified* region, it must overtake this announcement, and because of the FIFO assumption of the channels and the rules for dispersion of the announcement, this is not possible. Therefore, the announcement can only be delivered once.

To more explicitly show the relationship between the snapshot algorithm and announcement delivery, we provide a reduction proof outline 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 process begins its local snapshot when it receives the first marker. When this occurs, the marker is sent on all other 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 incident links.

We translate these actions to the mobile environment. The announcement corresponds to the marker, and the mobile node 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, the delivery algorithm *will keep a copy of the announcement until it receives a copy of the announcement from all neighbors.* Intuitively, this prevents the mobile from hopping from node to node eluding the announcement. Thus if the mobile arrives prior to the announcement on a channel, the MSC delivers the data as soon as the handover is complete (Figure 2d) because it has a stored local copy.

4 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 will look at the 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 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 to be 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 look in detail at the movement of messages and mobile units, specifically the handover of mobile units between cells. We will 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 [17], 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 potential indicating the mobile unit is moving into that particular cell, and an available frequency within that cell. When the frequency is requested, a handover begins. Figure 5a 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 which 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 messages overtakes the mobile.

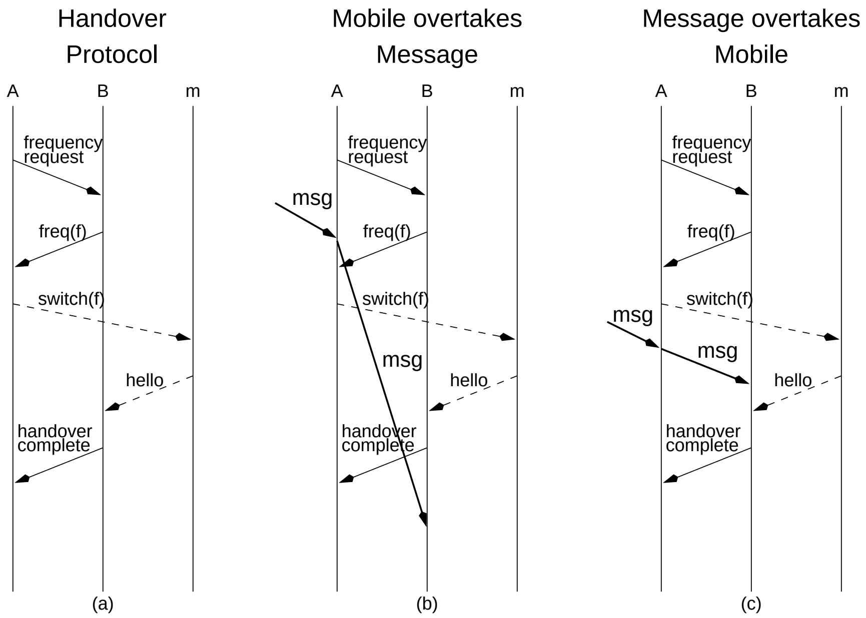

Figure 5: AMPS handover protocol (a). If messages are processed (i.e., broadcast to the mobile) immediately upon receipt, it is possible (b) for the mobile to move faster than the message along the channel, or (c) for the message to move faster than the mobile, thus breaking the FIFO channel property.

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, or when the *switch* message is transmitted. Similarly, the mobile moves off of the channel when the wireless transmission of the *hello* message is accepted at the destination. As can be seen in Figure 5b, it is possible for a messages 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 5c).

We propose a minor change in the protocol 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 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 before the VMU were sent before the mobile unit left, and all messages after the VMU were sent after the mobile unit left. Therefore, we change the behavior of the destination (B in this case) to yield this behavior. The goal is 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 for both mobile units and messages.

Multiple RBSs per MSC: Until this point we have only allowed one radio broadcast 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 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 handovers 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 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 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 handover was used for disconnected movement, 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 [18] and OSPF [13]. 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 [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 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 announcement 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 are 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 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 agents for a bounded period of time limited to the duration of the local snapshot. This is more efficient than another proposal [1] which broadcasts the announcement to all nodes, which then store the announcement until notified that delivery has occurred. This is essentially a *network propagation delay* factor. In a system with bi-directional channels, because the local snapshot terminates when the announcement arrives on all incoming channels, a local snapshot can be as short as a single round trip delay between the MSCs, or a *local propagation delay.* 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.

5 Extensions

The snapshot delivery algorithm is extendable to perform delivery of multiple announcements simultaneously, delivery to rapidly moving mobile units, route discovery, multicast, and working with the mobile code environment. Each of these will be examined in turn.

Multiple announcement deliveries: To deliver multiple announcements simultaneously using snapshots, we run several copies of the algorithm in parallel. This is analogous to having each MSC 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 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 [1] 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 an announcement is attempted to all mobile units in the system before the algorithm terminates. If the announcement 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 an announcement 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 announcement per link in each direction to multicast. By contrast, the algorithm used in IP DVMRP [5] 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.

Mobile code: Thus far we have only considered physical movement of mobile units, but another possible application that is characterized by rapid mobile movement is mobile code where it is not a physical component that moves, but rather a program moving along the fixed network doing computation at various network nodes. Rather than connecting to a foreign agent through a wireless mechanism, these mobile code 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.

6 Conclusions

In this paper we presented a design paradigm for mobile computing based on the idea of treating a mobile unit as a message in a distributed computation. Its mechanics are borrowed directly from the established literature of distributed computing. The particular example presented adapted a snapshot algorithm to search for a mobile unit in a fixed network and deliver a message. The ease of extending this algorithm to reliable multicast in a mobile environment without knowing the recipients, as well as working in the mobile code arena are added benefits to the approach. More generally, treating mobile units as messages provides an effective means for transferring results from classical distributed algorithm literature to the emerging field of mobile computing. Ongoing work includes applying the same techniques to other algorithms. Specifically, we have adapted termination detection for diffusing computations to track the current location of a mobile unit in a distributed manner. An announcement delivery mechanism is then superimposed on the tracking graph to perform reliable message delivery.

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] K.M Chandy and L. Lamport. Distributed snapshots: Determining global states of distributed systems. *ACM Transactions on Computing Systems*, 3(1):63–75, 1985.

[5] S.E. Deering and D.R. Cheriton. Multicast routing in datagram internetworks and extended lans. *ACM Transactions on Computer Systems*, 8(2):85–110, 1990.

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

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

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

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

[10] 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.

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

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

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

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

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

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

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

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