Reliable Multicast in Mobile Networks

Ravi Prakash
The University of Texas at Dallas
Richardson, TX 75080
Email: ravip@utdallas.edu

André Schiper
Swiss Federal Institute of Technology (EPFL)
Lausanne
Email: Andre.Schiper@epfl.ch

Mansoor Mohsin
The University of Texas at Dallas
Richardson, TX 75080
Email: mmohsin@utdallas.edu

***Abstract*— This paper describes a distributed algorithm for reliable multicast in mobile cellular networks. In the proposed solution the multicast message is flooded to all the base stations over reliable channels. The base stations then collectively ensure that all mobile nodes belonging to the multicast group get the message. The originality of our solution is that it is fully decentralized. Each base station can independently decide when to flush a message from its buffer. Even if a node moves from one cell to another while a multicast is in progress, delivery of the message to the node is guaranteed. Simulation experiments show that using the proposed solution, memory requirements at the base stations are significantly smaller than by using centralized solutions.**

I. Introduction

This paper describes a distributed algorithm for reliable multicast in mobile networks. An early centralized solution for reliable multicasts in cellular networks was proposed by Acharya and Badrinath [1]. In their solution a single node is responsible for determining when the message has been delivered to all the target nodes. As a result, this node has to bear greater responsibility than other nodes. Moreover, if delivery of a message to target nodes in some part of the network is delayed, then base stations in all other parts of the network have to continue to store the message in their respective buffers. In our solution, each base station independently decides when to purge a message from its buffer. Even when the multicast is in progress in one part of the network, base stations in other parts of the network can safely purge their buffers. This results in low buffer occupancy.

Multicasting protocols for static networks like Distance-Vector Multicast Routing Protocol (DVMRP) [8] and Core Based Tree (CBT) [2] are best-effort multicasting protocols. So, they do not guarantee that a multicast message will be delivered to all members of the multicast group. Attempts towards increasing the reliability of multicasts have been made in the context of mobile ad hoc networks. A solution based on CBT has been presented in [5]. Another solution, that is consistent with the core-based as well as source-based tree approaches is described in [7]. Unlike other protocols that try to find the shortest path to the source or the core, in this solution a multicast group member tries to find the nearest forwarding node in the multicast tree. Another solution, based on anonymous gossip, is described in [3]. Neither of these three protocols guarantee delivery of the multicast messages to all members of the multicast group. Part of the problem is the inherent dynamism of the mobile network.

We assume a model similar to that in [1], i.e., a collection of mobile networks, each with an access point/base station, and membership in multiple multicast groups. Each base station can independently decide when to flush a message from its buffer, thus reducing the buffer requirements, but still guaranteeing that every member of the multicast group receives a multicast message. The solution can be easily extended to cluster-based mobile ad hoc networks.

We believe that due to node mobility, significant communication overheads may be incurred in maintaining multicast trees: either source-based or core-based. Such overheads may be difficult to justify, especially if the *communication to mobility ratio (CMR)* is low. Hence, in the proposed solution first the multicast message is reliably flooded to all the base stations. After that, the base stations collectively ensure that all mobile nodes belonging to the multicast group get the message. Even if a mobile node, belonging to a multicast group, moves from one cell to another while a multicast to that group is in progress, delivery of the message to the mobile node is guaranteed.

II. System Model

We consider a system composed of multiple, interconnected wireless LANs. The region served by an access point will be referred to as its service region. Henceforth, we will also refer to the i^{th} access point as i^{th} leader, or L_i. The region served by L_i will be referred to as R_i. Each mobile node will register with one of the leaders within wireless communication range. So, there could be zero or more mobile nodes in a region. Message communication for mobile nodes in a region is coordinated by that region's leader. Communication between a leader and the nodes in its region is unreliable.

The leaders can communicate with each other over a wired backbone network. We assume that there exists a protocol for reliable delivery of messages over this backbone network. All communication between a pair of leader nodes is FIFO.

Two regions, R_i and R_j, are said to be neighboring regions if they overlap or share a common boundary. Each leader knows the identities of the leaders of its neighboring regions. A mobile node, M_x, can move out of one region only into one of the neighboring regions. When such a movement occurs from region R_i to neighboring region R_j, the responsibility for reliable delivery of messages to M_x are *handed-off* from L_i to L_j. The *hand-off protocol* is illustrated in Figure 1 and is triggered concurrently with the handoff at the physical/MAC

sublayer.[1] The hand-off starts with M_x sending a message to L_j (denoted by m_1) informing L_j that it is moving from R_i to R_j. On receiving this message, L_j sends an *inform* message to L_i (denoted by m_2 in the figure). When L_i's acknowledgment (m_3) for this message reaches L_i the hand-off is complete. At this point, M_x is said to have de-registered from L_i and registered with L_j. Message m_3 may also carry information about multicast messages that L_i had been trying to deliver to M_x prior to the hand-off.

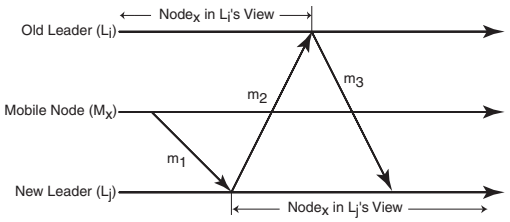

Fig. 1. Handoff of mobile node M_x from region R_i to region R_j. $m_1 = M_x$ leaving R_i, entering R_j; $m_2 = inform(x,i,j)$; $m_3 = ack\_handoff(x,i,j)$.

We define the view, V_i, of leader L_i as the set of mobile nodes for which L_i is responsible. A mobile node joins the view of a leader as soon as the leader becomes aware of the hand-off of that mobile node into its region. A mobile node leaves the view of a leader when the leader receives an inform message from the neighboring leader into whose area the node is moving. So, in Figure 1, mobile node M_x is in the views of both L_i and L_j for a certain duration during the hand-off. This *overlap* of views is crucial for the proposed solution.

At any time there may be multiple non-empty multicast groups in the network. In the following sections we focus on one such group, namely group y.

III. Algorithm

A. Basic Idea

Let msg_y be a message that is supposed to be reliably multicast to all mobile nodes belonging to multicast group y. If the source is a mobile node, the source sends the message to its leader in a reliable fashion (i.e., keeps transmitting until an ACK is received from the leader). Subsequently, the leader reliably floods the message over the wired backbone network to all leaders in the network. On receiving the message, leader L_i broadcasts msg_y to all mobile nodes in V_i that belong to group y, and expects acknowledgments from all such mobile nodes. If all expected acknowledgments are received by L_i, then L_i sends a *Finish* message corresponding to msg_y to all its neighboring leaders. Leader L_i purges msg_y from its buffer when both the following conditions are satisfied:

1) L_i has generated its own *Finish* message corresponding to msg_y, and
2) L_i has received the corresponding *Finish* messages from all its neighboring leaders.

[1]We assume that mobile nodes sense the beacons that leaders transmit at regular intervals. If the time average strength of the current leader's beacon falls below a lower threshold and that of a neighboring leader is above an upper threshold, the mobile node is to be handed off.

Let a mobile node M_x be present in V_i at the time L_i receives msg_y. Let M_x, which is a member of multicast group y, move out of R_i and into a neighboring region R_j before acknowledging the receipt of msg_y. Then: (i) L_i requests L_j to ensure that msg_y is delivered to M_x, and (ii) L_i deletes M_x from its view and removes M_x from the list of nodes from which acknowledgment for the reception of msg_y is expected. Similarly, let M_x enter R_j before L_j has purged msg_y. Then, L_j assumes the responsibility of reliably delivering msg_y to M_x.

B. Data Structures

Each mobile node M_x maintains a set, G_x, which contains identities of all multicast groups to which M_x belongs. Each leader L_i maintains V_i that represents its view, i.e., the set of mobile nodes for which L_i is responsible. In addition, L_i also maintains the following data structures:

- Nbr_i: neighbors of region R_i.
- $Member_{i,y}$: members of multicast group y in L_i's view. We have $Member_{i,y} \subseteq V_i$.
- $Messages_{i,y}$: messages for multicast group y that L_i has received, but not purged, i.e., group y messages that are still in L_i's buffer.

The messages that leader L_i has the responsibility of reliably delivering are held in two data structures, one for messages that are multicasts to some group y, and another for point-to-point unicast to one specific node M_x. The point-to-point mode is used by leader L_i when some node M_x, member of group y, enters region L_i at a time when L_i's broadcast to group y has terminated. The two data structures are the following:

- The set $Multicast_i$ contains tuples (msg_y, y), where msg_y is a message that has to be multicast to the members of group y, and not all nodes $M_x \in R_i$ that belong to group y have yet acknowledged msg_y.
- The set $Unicast_i$ contains tuples (msg_y, M_x), where msg_y is a message that has to be send to M_x, and M_x has not yet acknowledged msg_y.

To know when (msg_y, y) can be removed from $Multicast_i$, we need an additional data structure:

- $AckPending_{i,y,msg_y}$: mobile nodes in V_i that belong to group y and have yet to acknowledge the receipt of a group y message msg_y. Note that msg_y is an element of $Messages_{i,y}$. Also, $AckPending_{i,y,msg_y}$ is a subset of $Member_{i,y}$.

$AckPending_{i,y,msg_y}$ gets created at L_i when L_i receives msg_y and adds it to $Messages_{i,y}$. L_i purges this set at the same time that it purges msg_y from $Messages_{i,y}$. Finally, we need a data structure that represents the state of message msg_y with respect to region R_i:

- $state_i(msg_y) = white$, if the leader L_i has not yet received msg_y as part of the standard broadcast of messages among region leaders.
- $state_i(msg_y) = green$, if $msg_y \in Messages_{i,y}$ and $AckPending_{i,y,msg_y} \neq \emptyset$. Such a message msg_y is periodically broadcast by the leader L_i.

- $state_i(msg_y) = yellow$, if $msg_y \in Messages_{i,y}$ and $AckPending_{i,y,msg_y} = \emptyset$. Such a message msg_y is no more periodically broadcast by the leader L_i.
- $state_i(msg_y) = red$, if $msg_y \notin Messages_{i,y}$ and $AckPending_{i,y,msg_y} = \emptyset$. Such a message msg_y has been purged by the leader L_i.

We now give an informal description of the algorithm.

C. Arrival of message msg_y for group y at L_i

First, L_i adds msg_y to $Messages_{i,y}$, creates the set $AckPending_{i,y,msg_y}$ and initializes it with all nodes in $Member_{i,y}$. Then, L_i periodically broadcasts a copy of msg_y to all nodes in $Member_{i,y}$ that are in R_i. Upon receiving msg_y node M_x sends back ack_{x,y,msg_y} to L_i.

D. Arrival of ack_{x,y,msg_y} at L_i

L_i deletes M_x from $AckPending_{i,y,msg_y}$. If, as a result, of this deletion $AckPending_{i,y,msg_y}$ becomes empty then L_i sends $finish_{i,y,msg_y}$ to the leaders of all neighboring regions.

If L_i has generated $finish_{i,y,msg_y}$ and has received $finish_{j,y,msg_y}$ from all neighboring leaders L_j then L_i: (i) deletes msg_y from $Messages_{i,y}$, and (ii) eliminates the $AckPending_{i,y,msg_y}$ set corresponding to msg_y.

An ack_{x,y,msg_y} message can also be received as a result of a unicast message from L_i to M_x. In this case L_i simply removes (msg_y, M_x) from the set $Unicast_i$.

E. Node Mobility and View Change

Let mobile node M_x enter region R_j coming from a neighboring region R_i. Node M_x sends the *M_x leaving R_i, joining R_j* message to L_j. To simplify the algorithm, we assume here that the transmission of this message is reliable. Later in the paper we relax this assumption.

a) Operations by L_j: On receiving *M_x leaving R_i, joining R_j* from M_x, L_j adds M_x to its view, V_j. Also, L_j sends *inform(x, i, j)* to L_i (message m_2 in Fig. 1). For all groups y, such that $y \in G_x$, L_j adds M_x to $Member_{j,y}$. For all messages msg_y such that $msg_y \in Messages_{j,y}$ and $y \in G_x$, L_j performs the following operations:

- if the state of the region with respect to msg_y is green, then M_x is added to $AckPending_{j,y,msg_y}$,
- if the state is yellow, then (msg_y, M_x) is added to $Unicast_i$.

b) Operations by L_i: On receiving *inform(x, i, j)* from L_j, node L_i deletes M_x from V_i and from all $Member_{i,y}$, where $y \in G_x$. Then L_i sends *ack_handoff(x, i, j)* message to L_j (m_3 on Fig. 1). Moreover, if $M_x \in AckPending_{i,y,msg_y}$ for some message msg_y then L_i asks L_j to deliver msg_y to M_x on L_i's behalf (msg_y is also carried by m_3 on Fig. 1), and deletes M_x from $AckPending_{i,y,msg_y}$.

F. Arrival of msg_y at L_j from L_i for node M_x

This event occurs when message m_3 (Fig. 1) arrives at L_j, carrying some messages msg_y. For each such message msg_y, L_j performs the following operation. If the state of the region with respect to msg_y is yellow or red, then (msg_y, M_x) is added to $Unicast_i$. This transfers to L_j the responsibility of delivering msg_y to M_x.

G. Ensuring Message Delivery to Fast Moving Nodes

Let R_j be the neighbor of regions R_i and R_k such that R_i and R_k are not adjacent to each other. It is possible that L_k has sent a *Finish* message corresponding to msg_y to all its neighboring leaders and has also received *Finish* messages from all of them. So, as per the protocol (Sect. III-D), L_k would purge the copy of message msg_y from its buffer. However, L_i and L_j might still have msg_y in their respective buffers.[2] Let a fast moving mobile node M_x, which is a member of group y, move from R_i to R_j and then from R_j to R_k. M_x stays in each region only long enough to join and subsequently leave their views. However, M_x does not stay long enough in regions R_i and R_j for their leaders to deliver the message to M_x. So, now M_x is in a region whose leader has already purged its copy of msg_y.

In such a situation, too, it is guaranteed that the mobile node will ultimately receive the message. As described in Sections III-E and III-F, L_i will first send msg_y to L_j for delivery to M_x. Subsequently, following the same procedure, L_j will forward the message to L_k for delivery to M_x. So, even though L_k had purged the copy of M_x, another copy of the message will reach L_k along the path taken by M_x.

IV. Pseudo-code

We now give the details of the protocol. We distinguish between three parts: (1) multicasting of some message msg_y by some node M_x (2) execution at leader L_i to deliver msg_y and (3) execution at mobile node M_x.

A. Multicasting of message msg_y to group y by node M_x:

- M_x repeatedly sends message (msg_y, y) to its leader L_i until an acknowledgment is received.
- L_i broadcasts (msg_y, y) along the reliable backbone channels to all leaders.

B. Execution at leader L_i to deliver msg_y:

Periodically
(1) $\forall (msg_y, y) \in Multicast_i$: $broadcast(msg_y, y)$;
(2) $\forall (msg_y, M_x) \in Unicast_i$: $send(msg_y, M_x)$ to M_x;
On receiving msg_y
(3) $state_i(msg_y) = green$
(4) create($AckPending_{i,y,msg_y}$);
(5) $Messages_{i,y} = Messages_{i,y} \cup \{msg_y\}$;
(6) $AckPending_{i,y,msg_y} = Member_{i,y}$;
(7) $Multicast_i = Multicast_i \cup \{(msg_y, y)\}$;
On receiving ack_{x,y,msg_y}
(8) $Unicast_i = Unicast_i - \{(msg_y, M_x)\}$;
(9) if $\exists AckPending_{i,y,msg_y}$
(10) { $AckPending_{i,y,msg_y} = AckPending_{i,y,msg_y} - \{M_x\}$;
(11) if $AckPending_{i,y,msg_y} == \emptyset$
(12) Execute code for $AckPending_{i,y,msg_y} == \emptyset$
(13) }
Code for $AckPending_{i,y,msg_y} == \emptyset$
(14) { $state_i(msg_y) = yellow$:
(15) $Multicast_i = Multicast_i - \{(msg_y, y)\}$;
(16) send $finish_{i,y,msg_y}$ to all L_j: $R_j \in Nbr_i$;
(17) if received $finish_{j,y,msg_y}$ from all $R_j \in Nbr_i$
(18) { $Messages_{i,y} = Messages_{i,y} - \{msg_y\}$;
(19) delete($AckPending_{i,y,msg_y}$);
(20) $state_i(msg_y) = red$
(21) }

[2]This may happen if L_j has sent a *Finish* message to L_i and L_k, but not yet received a *Finish* message from L_i.

(22) }
On receiving "inform(x, i, j)"
(23) $LocUnicast = \emptyset$;
(24) $\forall\ msg_y$: $M_x \in AckPending_{i,y,msg_y}$
(25) { $LocUnicast = LocUnicast \cup \{(msg_y, M_x)\}$;
(26) $AckPending_{i,y,msg_y} = AckPending_{i,y,msg_y} - \{M_x\}$;
(27) if $AckPending_{i,y,msg_y} == \emptyset$
(28) Execute code for $AckPending_{i,y,msg_y} == \emptyset$
(29) $\forall\ msg_y$: $(msg_y, M_x) \in Unicast_i$
(30) { $LocUnicast = LocUnicast \cup \{(msg_y, M_x)\}$;
(31) $Unicast_i = Unicast_i - \{(msg_y, M_x)\}$;
(32) }
(33) send "ack_handoff(x, i, j)" and "(M_x, LocUnicast)" to L_j;
(34) $V_i = V_i$ - $\{M_x\}$;
(35) $Member_{i,y} = Member_{i,y}$ - $\{M_x\}$ $\forall y:\ y \in G_x$
On receiving "M_x leaving R_j, entering R_i"
(36) $V_i = V_i \cup \{M_x\}$;
(37) $Member_{i,y} = Member_{i,y} \cup \{M_x\}$; $\forall y : y \in G_x$;
(38) $\forall\ msg_y : msg_y \in Messages_{i,y} \wedge y \in G_x$
(39) { if $state_i(msg_y) = green$
(40) $AckPending_{i,y,msg_y} = AckPending_{i,y,msg_y} \cup \{M_x\}$;
(41) if $state_i(msg_y) = yellow$
(42) $Unicast_i = Unicast_i \cup \{(msg_y, M_x)\}$
(43) }
(44) send "inform(x, j, i)" to L_j
On receiving "ack_handoff(...) along with "(M_x, unicast)"
(45) $\forall\ msg_y : (msg_y, M_x) \in unicast$
(46) if $state_i(msg_y) = yellow \vee state_i(msg_y) = red$
(47) $Unicast_i = Unicast_i \cup \{(msg_y, M_x)\}$

C. Execution at mobile node M_x:

On receiving msg_y:
duplicate messages are discarded;
send ack_{x,y,msg_y} to regional leader from which msg_y is received;
On sensing need to switch to new leader L_j while in the view of L_i:
send "M_x leaving R_i, joining R_j" to L_j;

V. Proof of Correctness

We say that "region R_i has message msg_y for node M_x" if either (1) $(msg_y, M_x) \in Multicast_i$ and $M_x \in AckPending_{i,y,msg_y}$, or (2) $(msg_y, M_x) \in Unicast_i$.

Lemma 1: Let t_0 be the time at which message msg_y is multicast by the leader of some region R to group y, and let M_x be some mobile node member of group y. If at some time $t > t_0$ node M_x is in a white region R_i, then eventually either (1) R_i has message msg_y for M_x, or (2) M_x moves to a region R_j that is not red.

Proof. The diffusion algorithm between leaders ensures that every white region eventually becomes green. If at the time when this happens, node M_x is still in R_i, then by lines 6 and 7 region R_i has $msg_y$for M_x. Otherwise, M_x moves to R_j while R_i is still white. We prove that R_j cannot be red.

A red region is surrounded only by red or yellow regions. This holds because (1) a node becomes red only after having received the *finish* message from all its neighbors (line 17), and (2) the message *finish* is sent only by the leader of region R_i in the *yellow* state (line 13). So, as R_i is white, the neighbor region R_j cannot be red. □

Lemma 2: If node M_x, a member of group y, is forever in region R_i, and R_i has message msg_y for M_x, then M_x eventually receives msg_y.

Proof. If M_x is forever in region R_i, then (1) M_x is not removed from $AckPending_{i,y,msg_y}$(line 26) and (2) (msg_y, M_x) is not removed from $Unicast_i$ (line 31) unless M_x sends ack_{x,y,msg_y} to the leader L_i (which happens only if M_x receives msg_y). So, until M_x receives msg_y, we have $(msg_y, y) \in Multicast_i$ or $(msg_y, M_x) \in Unicast_i$, i.e., the leader L_i periodically either multicasts msg_y or sends msg_y. As channels are assumed to be fair-lossy, M_x eventually receives msg_y. □

Lemma 3: If node M_x member of group y moves from region R_i to a non-white region R_j, and if R_i has message msg_y for M_x before M_x moves, then after the leader L_j has received the $ack\_handoff(x, i, j)$ message from the leader L_i, region R_j has message msg_y for M_x.

Proof. From the pseudo-code, it follows directly that if (1) $M_x \in AckPending_{i,y,msg_y}$ (line 24), or (2) $(msg_y, M_x) \in Unicast_i$ (line 29), then when M_x moves to region R_j, the leader L_i sends message $(M_x, LocUnicast)$, with $(msg_y, M_x) \in LocUnicast$, to the leader L_j (lines 25, 30). Upon reception of this message, L_j does the following. If region R_j is green, i.e., $(msg_y, M_x) \in Multicast_j$, then M_x is added to the set $AckPending_{i,y,msg_y}$(line 40), i.e., R_j has message msg_y for M_x. If region R_j is yellow or red, (msg_y, M_x) is added to the set $Unicast_j$ (line 47), i.e., R_j has message msg_y for M_x. □

Proposition 1: Let message msg_y be multicast by the leader of region R_0 to group y, and let M_x be some mobile node member of group y. Then eventually either (1) M_x receives m, or (2) M_x keeps moving form one region R_i to another, and each region R_i has message msg_y for M_x.

Proof. Let us assume that M_x never receives msg_y. We have to prove that M_x keeps moving from one region to another, and eventually each region has message msg_y for M_x.

Let R_i be the latest white region to which node M_x belongs. By Lemma 1, either region R_i eventually (1) becomes green, or (2) M_x moves to a green or yellow region R_j.

If R_i becomes green while M_x is still in region R_i, then we have: $(msg_y, M_x) \in Multicast_i$ and $M_x \in AckPending_{i,y,msg_y}$. By Lemma 2, as M_x never receives msg_y, M_x must eventually leave region R_i.

We have two cases to consider. Case 1: M_x moves to region R_j before R_i becomes green. Case 2: M_x moves to region R_j after R_i becomes green.

Case 1. If R_j is green, then by lines 40, R_j has message msg_y for M_x. If R_j is yellow, then after executing line 42, R_j has message msg_y for M_x.

Case 2. If M_x moves to region R_j after R_i becomes green, then region R_i has message msg_y for M_x, and by Lemma 3, after having received message $ack\_handoff(x, i, j)$, region R_j also has message $msg_y$for M_x.

So in both cases R_j has message $msg_y$for M_x. By Lemma 2, as M_x does not receive msg_y, M_x must move to another region R_k. By Lemma 3, R_k has message msg_y for M_x. So if M_x does not receive msg_y, M_x must keep moving from one region to another, and each region has msg_y for M_x. □

Proposition 1 does not ensure that M_x eventually receives msg_y. However, it ensures that if M_x eventually remains in some region, then M_x eventually receives msg_y.

In order to prove that M_x receives msg_y without remaining forever in some region, we need some additional synchrony assumption about the time a node must stay in a region to ensure reception of msg_y. This time duration can be expressed as a multiple of the maximum message propagation delay.

VI. Discussion

A. Handling the unreliability of the "M_x leaving R_i entering R_j" message

Until now we have assumed that the message M_x *leaving* R_i *entering* R_j is reliably transmitted between M_x and L_j. We now relax this assumption. So M_x must keep sending this message until it is acknowledged by L_j. Only then M_x considers itself registered in region R_j. However, if M_x moves fast and leaves R_j for R_k before having received the acknowledgment from L_j we can have the following cases:

1) L_j has received the M_x *leaving* R_i *entering* R_j message, has sent the *inform(x, i, j)* message to L_i and has taken the responsibility for delivering messages to M_x. The acknowledgment from L_j to M_x has been lost.
2) L_j has not received M_x *leaving* R_i *entering* R_j message. L_i is still responsible for delivering messages to M_x.

However, M_x cannot distinguish between these two cases. So if M_x moves to R_k it does not know whether to send M_x *leaving* R_i *entering* R_k to L_k, or rather M_x *leaving* R_j *entering* R_k. So, instead, M_x sends M_x *leaving* (R_i,R_j) *entering* R_k, telling L_k to ask both L_i and L_j to transfer to L_k the responsibility of messages for M_x. Only one of the two leaders has actually the responsibility, and will transfer it to L_k. In other words, the changes to the protocol are the following:

- a mobile node M_x manages a list LR_x of regions it might come from, and upon entering region R_j periodically sends M_x *leaving* LR_x *entering* R_j to L_j.
- If M_x receives an acknowledgment from L_j, then M_x sets LR_x to empty.
- Upon receiving M_x *leaving* LR_x *entering* R_j for the first time, L_j sends an *inform(x, l, j)* to all leaders l of regions in LR_x. Only one of them, say L_i, has M_x in V_i and sends the "ack_handoff" message to L_j. Moreover, each time L_j receives M_x *leaving* LR_x *entering* R_j it sends an acknowledgment to M_x.
- Upon leaving R_j, node M_x adds R_j to LR_x.

VII. Simulation

We implemented the protocol using the CSIM[3] simulator. Since the protocol runs at the application layer, we have not addressed routing and MAC layer issues. However, our simulator does capture the behavior of the lower layers of the protocol stack through judicious use of message propagation delays and timers.

A. Simulation Setup

The topology of the base stations is fixed and predefined at the beginning of the simulation. Every base station has either two, three or four neighboring base stations.

In the beginning of the simulation, mobile nodes are placed uniformly in the regions of all the base stations. We model the movement of a mobile node as a *jump* from one region to another neighboring region. The destination region is chosen randomly among all the neighboring regions. The location of a mobile node is represented solely in terms of the base station in whose region it resides.

[3]CSIM is discrete-event simulation package for use with C programs.

B. Simulation Parameters

The different parameters used in the simulation and their values are given in Table I.

TABLE I
Simulation Parameters

Parameter	Value
(base-stations, mobile-nodes)	(9,100), (16,150), (25,200)
Multicast Groups	10
Nodes in a Multicast Group	25
Simulation Time	7200 s
Node Move Interval	900 s
Message Send Interval	150 s
(wired, wireless) Delay	(0.01,0.1) s, (0.1,1.0) s, (0.5,5.0) s

In the above table, *node move interval* is the time a mobile node stays in a region before it moves into a new region, and *message send interval* is the time between sending successive multicast messages by a mobile node. Not all mobile nodes belonging to a multicast group will experience the same delay for a given multicast message. Values of the last four parameters are exponentially distributed, with the values given in the above table as mean.

C. Results

Table II show the results of the simulation experiments. The performance metric used in the simulation is multicast latency, i.e., the time between sending a multicast message and the corresponding receive by a mobile node. We consider two cases – nodes that performed a handoff while the multicast was in progress and nodes that did not perform a handoff during a multicast. We also consider the overall finish time of a multicast, i.e., the time between sending a multicast message and the last node receiving that message. This is indicated by *Multicast Finish Time* in Table II. *Buffer Occupancy Time* is the mean duration for which a message is kept in a buffer at a base station.

TABLE II
Wired Delay = 0.01 s, Wireless Delay = 0.1 s

	Number of Base Stations		
	9	16	25
No Handoff	0.211 s	0.214 s	0.218 s
Handoff	0.214 s	0.208 s	0.233 s
Multicast Finish Time	0.383 s	0.397 s	0.412 s
Buffer Occupancy Time	0.230 s	0.176 s	0.144 s

The mean multicast latency with no handoff is about 0.211 s when there are 9 base stations. This corresponds to 0.1 s on the wireless link from the multicast source to its base station, 0.1 s on the wireless link from a base station to a mobile node in its region, and the rest is the latency on the wired links between the base stations. This is equal to $2t_{lm}+t_l$, where t_l is the upper bound on message propagation delay between leaders and t_{lm} is the corresponding delay between leader and mobile node. The mean multicast latency with handoff is 0.214 s, which is slightly higher than with no handoff. This is because in addition to all the messages involved when there is no handoff, there are additional messages exchanged between the neighboring base stations to perform a handoff. These additional messages are the *inform* and *ack_handoff* message. Hence, the latency is $2t_{lm}+3t_l$.

The buffer occupancy time for base stations is considerably smaller than the multicast finish time. This is because a base station independently decides when to purge a message from its buffer, based solely on its local information and the status of its neighbors. We also see that as the size of the network increases, the buffer occupancy time decreases. To explain why this happens, let x be the last mobile node that receives a multicast message, and let X be the base station of x. All the neighboring base stations of X will purge the multicast message from their buffer only after x receives the multicast message, resulting in a higher buffer occupancy time. This effect is diminished in a large network, since the node that receives the multicast message last only affects base stations in its locality. This is in contrast to the centralized solution of Acharya and Badrinath [1], where unless all the mobile nodes of a multicast group receive the message, all the base stations keep the message in their buffer. The message is purged only after the last mobile node receiving the multicast message sends an *ack* message to its base station and that base station forwards it the coordinating node. The coordinator then sends a *purge* message to all the base stations which purge the message from their buffer upon receiving it. Hence, after the last node receives a multicast message, the base stations keep the message in their buffer for $t_{lm} + 2t_l$ time units. Moreover, by examining Table II, we see that as the size of the network increases the multicast finish time also increases, but the buffer occupancy time decreases. This means that as the network gets bigger, the centralized solution gets worse and the proposed solution gets better. Hence, the proposed distributed solution is more scalable than the centralized one.

Tables III and IV show the results of the same simulation experiment with increasing values of propagation delay. We see that the latency, multicast finish time, and buffer occupancy scale linearly with the propagation delay. However, by keeping the message sending rate constant and increasing the propagation delay, we inject more traffic into the network. Hence, as we increase propagation delay, the percentage of nodes receiving a multicast message that involved a handoff increases. This is illustrated in Table V. It also shows that the the percentage of nodes that experienced a handoff while receiving a multicast message is very small. Hence, it is important to have low cost delivery when there is no node movement. On the other hand, slightly higher cost on movement is acceptable because their percentage is small.

TABLE III

WIRED DELAY = 0.1 S, WIRELESS DELAY = 1.0 S

	Number of Base Stations		
	9	16	25
No Handoff	2.16 s	2.20 s	2.29 s
Handoff	2.59 s	2.23 s	2.67 s
Multicast Finish Time	3.76 s	3.99 s	4.16 s
Buffer Occupancy Time	2.17 s	1.83 s	1.47 s

VIII. CONCLUSION

We presented an algorithm that guarantees reliable delivery of multicast messages in a mobile network. Responsibility for reliable delivery is distributed among several nodes, which makes the solution scalable.

TABLE IV

WIRED DELAY = 0.5 S, WIRELESS DELAY = 5.0 S

	Number of Base Stations		
	9	16	25
No Handoff	11.4 s	12.0 s	12.3 s
Handoff	15.3 s	15.6 s	14.5 s
Multicast Finish Time	19.7 s	20.1 s	21.0 s
Buffer Occupancy Time	11.4 s	8.96 s	7.48 s

TABLE V

% OF MULTICASTS RECEIVED THAT INVOLVED A HANDOFF

	Number of Base Stations		
(wired, wireless) delay	9	16	25
0.1 s, 0.01 s	0.0261%	0.0372%	0.0101%
1.0 s, 0.1 s	0.412%	0.231%	0.217%
5.0 s, 0.5 s	2.16%	1.48%	1.35%

The solution incurs no multicast-tree maintenance overheads. Only a subset of nodes, referred to as leaders, need to maintain state information about multicast messages while attempts are being made to deliver these messages. Also, if a multicast message gets delivered to all the target nodes in a certain part of the network, copies of the message can be purged by the leaders in that part regardless of the progress made in other parts of the network.

In the proposed solution, the membership of each multicast group stays fixed. In the future, we plan to extend the proposed solution to allow mobile nodes to dynamically join and leave multicast groups. Our goal is also to extend the solution to cluster-based mobile ad hoc networks, where cluster heads could potentially perform the task that base stations do in supporting the multicast.

Acknowledgments

Ravi Prakash's work was supported by the National Science Foundation CAREER grant number CCR-0093411 and by the 2001 Summer Research Institute of EPFL, Lausanne. André Schiper's work was supported by the National Competence Center in Research on Mobile Information and Communication Systems (NCCR-MICS), a center supported by the Swiss National Science Foundation under grant 5005-67322.

REFERENCES

[1] A. Acharya and B. R. Badrinath. A Framework for Delivering Multicast Messages in Networks with Mobile Hosts. Journal on Mobile Networks and Applications 1 (2), pages 199–219, 1996.

[2] T. Ballardie, P. Francis, and J. Crowcroft. Core Based Trees (CBT). In *Proceedings of ACM SIGCOMM*, pages 85–95, Ithaca, N.Y., 1993.

[3] R. Chandra, V. Ramasubramanian, and K. P. Birman. Anonymous Gossip: Improving Multicast Reliability in Mobile Ad-Hoc Networks. In *Proceedings of the 21st IEEE International Conference on Distributed Computing Systems (ICDCS 2001)*, pages 275–283, Mesa, Arizona, April 2001.

[4] E. W. Dijkstra and C. S. Scholten. Termination detection for diffusing computations. *Information Processing Letters*, 11(1):1–4, 1980.

[5] S. K. S. Gupta and P. Srimani. An Adaptive Protocol for Reliable Multicast in Mobile Multi-hop Radio Networks. In *Proceedings of 2nd IEEE Workshop on Mobile Computing Systems and Applications (WMCSA'99)*, pages 111–122, New Orleans, February 1999.

[6] S. T. Huang. Detecting Termination of Distributed Computations by External Agents. In *Proceedings of the 9th International Conference on Distributed Computing Systems*, pages 79–84, 1989.

[7] T. Ozaki, J. B. Kim, and T. Suda. Bandwidth-Efficient Multicast Routing for Multihop, Ad-Hoc Wireless Networks. In *Proceedings of IEEE INFOCOM 2001*, pages 1182–1191, 2001.

[8] C. Partridge, D. Waitzman, and S. Deering. RFC 1075, Distance Vector Multicast Routing Protocol. Internet Engineering Task Force, November 1988.