Scalable Multicasting: The Core-Assisted Mesh Protocol

Ewerton L. Madruga and J.J.Garcia-Luna-Aceves *
{madruga,jj}@cse.ucsc.edu
Computer Engineering Department
Baskin School of Engineering
University of California
Santa Cruz, CA 95064

Abstract

Most of the multicast routing protocols for ad-hoc networks today are based on shared or source-based trees; however, keeping a routing tree connected for the purpose of data forwarding may lead to a substantial network overhead. A different approach to multicast routing consists of building a shared mesh for each multicast group. In multicast meshes, data packets can be accepted from any router, as opposed to trees where data packets are only accepted from routers with whom a "tree branch" has been established. The difference among multicast routing protocols based on meshes is in the method used to build these structures. Some mesh-based protocols require the flooding of sender or receiver announcements over the whole network. This paper presents the Core-Assisted Mesh Protocol, which uses meshes for data forwarding, and avoids flooding by generalizing the notion of core-based trees introduced for internet multicasting. Group members form the mesh of a group by sending join requests to a set of cores. Simulation experiments show that meshes can be used effectively as multicast routing structures without the need for flooding control packets.

*This work was supported by the Defense Advanced Research Projects Agency (DARPA) under Grant F30602-97-2-0338 and by CNPq-Brazil.

Report Documentation Page

Form Approved
OMB No. 0704-0188

1. REPORT DATE	2. REPORT TYPE	3. DATES COVERED
1999		**00-00-1999 to 00-00-1999**

4. TITLE AND SUBTITLE	5a. CONTRACT NUMBER
Scalable ulticasting: The Core-Assisted Mesh Protocol	5b. GRANT NUMBER
	5c. PROGRAM ELEMENT NUMBER
6. AUTHOR(S)	5d. PROJECT NUMBER
	5e. TASK NUMBER
	5f. WORK UNIT NUMBER
7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES) **University of California at Santa Cruz,Department of Computer Engineering,Santa Cruz,CA,95064**	8. PERFORMING ORGANIZATION REPORT NUMBER
9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)	10. SPONSOR/MONITOR'S ACRONYM(S)
	11. SPONSOR/MONITOR'S REPORT NUMBER(S)

12. DISTRIBUTION/AVAILABILITY STATEMENT
Approved for public release; distribution unlimited

13. SUPPLEMENTARY NOTES

14. ABSTRACT

15. SUBJECT TERMS

16. SECURITY CLASSIFICATION OF:			17. LIMITATION OF ABSTRACT	18. NUMBER OF PAGES	19a. NAME OF RESPONSIBLE PERSON
a. REPORT **unclassified**	b. ABSTRACT **unclassified**	c. THIS PAGE **unclassified**		**27**	

1 Introduction

The basic approach for supporting many-to-many communication (multicasting) efficiently in computer networks consists of establishing a routing tree among a group of routers. Multicast routing protocols based on trees have been proposed and implemented for wired and wireless networks and the Internet (e.g., [1, 8, 9, 2, 4, 20, 23]).

The topology of a wireless mobile network can be very dynamic due to the mobility of routers and the characteristics of the radio channels. Maintaining a routing tree for the purposes of multicasting packets in these networks can incur substantial control traffic. Hence, using routing graphs that have more connectivity than trees and yet prevent long-term or permanent routing loops from occurring is desirable in wireless mobile networks; we call these routing graphs *multicast meshes*. The Forwarding Group Multicast Protocol (FGMP) [3] and the On-demand Multicast Routing Protocol (ODMRP) [14] are multicast routing protocols that build routing meshes rather than routing trees to disseminate multicast packets within groups. To establish group meshes, these protocols flood control packets in the entire wireless network. The difference between these two protocols is that the receivers initiate the flooding in FGMP, and the senders initiate the flooding in ODMRP. Although multicast meshes are better than multicast trees in dynamic networks, approaches to mesh building based on flooding incurs excessive overhead in large networks.

This paper verifies and analyzes the core-assisted mesh protocol (CAMP), which builds multicast meshes without having to flood the wireless network with control or data packets and routes multicast packets from any group source over the shortest from source to receivers defined in the group's mesh. CAMP builds multicast meshes following a receiver-initiated approach based on cores. Sections 2 to 6 describe the design principles and operation of CAMP. Section 7 shows that CAMP builds multicast meshes correctly and that multicast packets do not loop in multicast meshes. Section 8 describes the results of simulation experiments used to compare CAMP's performance against the performance of ODMRP, which uses flooding to build meshes. Previous work [11] has shown that CAMP performs much better than tree-based multicast routing protocols in mobile wireless networks. Section 9 provides our concluding remarks.

2 Overview of CAMP

CAMP [12] differs from most prior multicast routing protocols in that it builds and maintains a *multicast mesh* for information distribution within each multicast group. A multicast mesh is a subset of the network topology that provides at least one path from each source to each receiver in the multicast group. CAMP ensures that the shortest paths from receivers to sources (called reverse shortest paths) are part of a group's mesh. Packets are forwarded through the mesh along the paths that first reach the routers from the sources, i.e., the shortest paths from sources to receivers that can be defined within the mesh. CAMP does not predefine such paths along the mesh. A router keeps a cache of the identifiers of those packets it has forwarded recently, and forwards a multicast packet received from a neighbor if the packet identifier is not in its cache. The key difference between a mesh and a tree structure is how data packets are accepted to be processed. A router is allowed to accept unique packets coming from any neighbor in the mesh, as opposed to trees where

a router can only take packets coming from routers with whom a *tree branch* has been established. Therefore, keeping the branch information updated is one extra challenge protocols based on trees have to face in a mobility scenario.

Because a member router of a multicast mesh has redundant paths to any other router in the same mesh, topology changes are less likely to disrupt the flow of multicast data and to require the reconstruction of the routing structures that support packet forwarding. Figure 1 illustrates the differences between a multicast mesh and the corresponding shared multicast tree; routers that are members of the multicast group are dark. The multicast mesh and tree shown in the figure include routers that have host receivers, hosts that are senders and receivers, and routers that act only as relays. Router g is the last receiver to join the multicast group, and does so in the multicast mesh through either router f or h; consequently, router c does not become a member of the mesh.

CAMP extends the basic receiver-initiated approach introduced in the core-based tree (CBT) protocol [1] for the creation of multicast trees to enable the creation of multicast meshes. Cores are used to limit the control traffic needed for receivers to join multicast groups. In contrast to CBT, one or multiple cores can be defined for each mesh, cores need not be part of the mesh of their group, and routers can join a group even if all associated cores become unreachable.

A host first determines the address of the group it needs to join as a receiver. The host then uses that address to ask its attached router to join the multicast group using IGMP [7]. Upon receiving a host request to join a group, the router sends a join request towards a core if none of its neighbors are members of the group; otherwise it simply announces its membership using either reliable or persistent updates. If cores are not reachable from a router that needs to join a group, the router broadcasts its join request using an *expanded ring search* (ERS) that eventually reaches some group member. When one or multiple responses are sent back to the router, it chooses any of these responses to use as a path to the mesh. The mappings of multicast addresses to (one or more) core addresses are disseminated from each core out to the network as part of group membership reports.

The Core-Assisted Multicast Routing Protocol provides also an alternate way for routers connected to sender-only hosts to join the mesh. Whenever a router senses multicast packets originated at a host directly attached to it, this designated router will join the mesh in *simplex* mode if it's not a member yet. The simplex join request, just as a regular duplex join request, will travel towards one of the available cores and is acknowledged in the same fashion. The conceptual difference is that data packets should travel in only one direction: from the sender-only host to the mesh and not the opposite. This is an attempt to contain data traffic closer to the areas of the mesh where receivers are present. A router can leave the group when there are no other hosts or routers depending on it simply by advertising the change in group membership to their neighbors. More details about simplex joins as well as the handling of topology changes are presented in previous work [12].

A router leaving a multicast group issues a *quit notification* to its neighbors, who in turn can update their data structures. No acknowledgments are requested for quit notifications, because in contrast to multicast routing trees, multicast meshes do not dictate the paths taken by multicast packets. Quit notifications are sent as part of multicast routing updates.

In order to minimize delays, CAMP ensures that all the reverse shortest paths between sources and receivers are part of a group's mesh by means of *heartbeat* and *push join* (PJ) messages. A router receiving a heartbeat for a given multicast group and source retransmits the heartbeat if

its successor towards the source of data traffic (determined with the unicast routing protocol) is already a mesh member. When a member router receives a heartbeat and detects that its successor is not part of the multicast mesh, it sends a *push join* (PJ) to that neighbor router and waits for an ACK from that router.

3 Information used in CAMP

Each router maintains a routing table (RT) built with the unicast routing protocol. This table is also modified by CAMP when multicast groups need to be inserted or removed. CAMP assumes the existence of a *beaconing* protocol, usually embedded into the unicast routing protocol or available as a separate network service.

At router i, the RT made available to CAMP specifies, for each destination j, the successor (s_j^i) and the distance to the destination (D_j^i).

Other than the unicast routing table, CAMP relies on the following data structures:

- CAM : table mapping cores to multicast groups.
- $CORES_g$: set of routers acting as cores to multicast group g.
- $CACHE_i$: cache of multicast data packet control information.
- MRT_i : the multicast routing table, containing the set of groups known to router i.
- AT_i^g : table containing anchor information pertaining to router i. This table is split in two subsets:
  - A_i^g : list of neighbors that have router i as their anchor for multicast group g.
  - $A2_i^g$: list of neighbors who are anchors to router i in multicast group g.
- N_i^g : router i's list of neighbors that are known to be members of the multicast group g.
- LS_i^g : list of senders that are directly attached to router i and send data traffic to multicast group g.
- LR_i^g : list of receivers directly attached to router i, who want to receive data packets from multicast group g.
- $PEND_i^g$: list of either join or simplex join requests to multicast group g originated at or forwarded by router i for whom acknowledgment is pending.
- $PENDPJ_i^g$: list of push join requests to multicast group g originated at or forwarded by router i for whom acknowledgment is pending.
- BK_i^g : list used for periodic "book-keeping" of senders and associated anchors.

The packet-forwarding cache $CACHE_i$ maintains the identifier of packets recently processed by router i. Basically, the information kept in this data structure comes straight from the IP packet header, which are source address, destination address (group address), packet identification and fragment offset. The address of the neighbor that relayed that packet is also stored. The main role of the packet forwarding cache is to avoid packet replication by keeping track of packets already received by the router. Caching packets is only feasible for low-bandwidth channels. Although restricted to symmetric networks, an alternative to packet caching is the use of reverse path forwarding [6], where routers only accept data packets from their successor to the packet source.

An anchor for router i in group g is a neighbor router that is a successor (next hop) in the reverse shortest path to *at least one source* in the group g. Therefore, a router determines its anchor to a given source by using the unicast routing table. For each multicast group g, an entry in the table AT specifies those neighbors that router i uses as its *anchors* for the group. The table AT_i has an entry for each of the multicast groups in which router i is a member.

When MRT_i or AT_i is updated, router i sends a multicast routing update (MRU) to all its neighbors reporting changes in its group membership and anchors per group. An MRU contains one or more entries, and each entry specifies:

- A group address.
- An operation code specifying a quit notification, simplex membership notification, or a duplex membership notification.
- In membership notifications, the list of anchors needed by router i for the group and/or the advertisement list of active data traffic source nodes and transient cores in the group.

The main objective of communicating anchor information among routers is to prevent routers that are required by their neighbors to forward multicast packets from leaving groups prematurely.

The list of data source nodes is propagated only by duplex nodes, and therefore is flooded to the mesh only. A discussion on the propagation scope and period of the list of temporary cores is done in Section 4.

In an ad-hoc network, changes in multicast-group memberships should be disseminated together with routing-table updates, and routers hear the reports from their neighbors and remember which neighbors belong to which group. To save bandwidth, routers should exchange multicast routing information together with their unicast routing-table updates. Hence, a routing-table update would consist of a unicast part and a multicast part. However, we describe CAMP independently of the unicast routing protocol with which it is used.

Detecting the failure or addition of a link to a neighbor is part of the routing protocol used in conjunction with CAMP. For CAMP to work correctly, it is necessary for the associated routing protocol to work correctly in the presence of router failures and network partitions. This implies that CAMP cannot be used in conjunction with a routing protocol based on the classic distributed Bellman-Ford algorithm such as the routing protocol of the DARPA packet radio network [15], which is prone to routing loops and count-to-infinity problems. However, there are several recent examples of routing protocols that can be used in conjunction with CAMP [5, 16, 19, 22].

With minor extensions, CAMP can also work with on-demand unicast routing protocols, but these extensions are beyond the scope of this work.

4 Types of Cores

Physical channels in ad-hoc networks are usually characterized by low bandwidth. Therefore, any protocol designed for such an environment should always try to minimize control traffic. Multicast protocols that use cores as part of the routing structure depend on advertisement messages to propagate the list of cores currently available. With the purpose of targeting such overhead of multicast sessions on the internet, Ohta and Crowcroft [17] propose alternate static strategies, as the use of DNS as a means to disseminate core information. Basically, by adding the *CORE* resource record to DNS, one could set up all cores needed for any given multicast group. As pointed by Perkins [18], the intrinsic lack of hierarchy in ad-hoc networks may make DNS as it is today a hard fit for such networks. But still some name or session directory service is likely to be available.

CAMP makes use of the idea of making multicast group information available through such services, and has two different classes of cores: *static* and *transient* cores. A core defined as static has its ID obtained at system start-up from a directory service like DNS and saved locally, or alternatively is defined manually by the network administrator. On the other side, transient cores assume this role when no static core is available and are expected to exist temporarily, as the name implies.

Eventually, when a router needs to join the multicast mesh or is simply trying to use one of the mechanisms to keep the mesh connected, no static core may be reachable. After a back-off to make sure no other router in the area is already taking over this role, the router will become a transient core, and keep its status for as long as there isn't reachable static cores. When static cores become reachable again, there is a transition phase where static and transient cores co-exist, but the latter ones eventually age out. The back-off time is smaller for routers which have data traffic sources directly attached to it, which naturally minimizes delays, as source-based trees do, and also minimizes the need for push join requests.

The key difference between the two classes of cores is advertisement. Since the addresses of static cores is well-known, they do not need to advertise their existence, as opposed to transient ones that must advertise their addresses to the network periodically. One should note that, although CAMP is designed to use transient cores only in exceptional situations, the scheme is flexible enough to allow the choice between the bandwidth efficiency of static cores, which do not propagate advertisement messages, and the flexibility of transient cores, that come up on demand and whose definition can be based on policies like direct connection to traffic sources, gateway to a wired network, and so on. The simple absence of static core definition will let the network define transient cores dynamically, as the first duplex and simplex join requests come out.

One important issue regarding control message overhead in ad-hoc networks is the core advertisement period used by transient cores. As opposed to sender/receiver advertisement in ODMRP [14] and FGMP [3], which need a period on the order of hundreds of milliseconds, CAMP can rely on advertisement periods with tens of seconds or even minute granularity. In the sender-initiated protocols, senders need to be known as widely and as fast as possible so that nodes all over the network can join the mesh with minimal delays. In CAMP, if a particular region of the network

does not have information about any transient core, any router locally can become one and start a mesh component momentarily. As discussed in session 6.2, this component will merge with the remainder of the mesh as the ID of other transient cores become available locally.

5 Joining and Quitting the Multicast Mesh

In CAMP, as previously discussed, there are two different mechanisms for a router to join a multicast group mesh. If the router is attached to a receiver router, it sends a *join request* to its selected core. If instead it's attached to a host that is a source of multicast data traffic, the router will send a *simplex join request* to its preferred core. Additionally, after detecting data or receiving a list of active senders, a mesh member may request that all routers in its shortest path to a given source become mesh members with a *push join request*.

In the first mechanism, if a router joining a group has no neighbors that are already members of the multicast group, then it selects its successor to the nearest core as the relay for the join request. After the router selects a relay, it sends a join request to all its neighbors. A join request specifies the intended relay, the address of the multicast group that the sending router needs to join. If, on the other side, a neighbor is known to be a mesh member, the router joins the group without the need to send out a join request, and sends an update announcing its new membership status.

After sending a join request, a router waits for the first acknowledgment to its request, and retransmits the request after a request-timeout. The router persists sending join requests for a number of times (e.g., four) as long as the unicast routing table indicates that there are physical paths towards *any* of the group cores and none of its neighbors are group members. Each retransmission of a request is addressed to an intended relay chosen as described above. This is similar to the basic mechanism used in CBT; however, because data packets flow along different paths over the multicast mesh depending on the source, there is no need to ensure a single loopless path to the chosen core. This means that the use of selected relays towards any core is simply used to *limit* the search from the routers towards the multicast mesh, but being able to reach a core is not necessary to join a group.

Any router that is a duplex member of a multicast group and receives a join request for the group is free to transmit a *join acknowledgment* (ACK) to the sending router. An ACK specifies the sender of the join request and the multicast group being joined. To reduce channel traffic, the router specified as the relay of a join request can be allowed to reply first by means of a timeout mechanism after a join request is received.

When the origin or a relay of a join request receives the first ACK to its request or the first ACK to a join request for the same multicast group, the router becomes part of the multicast group. In the case of a relay, the router sends an ACK to the previous relay or origin of the join request, even if that neighbor has already sent an update stating that it is a member of the multicast group.

If non-member routers were allowed to send packets to a multicast mesh, the only way to reach the mesh without flooding would be through one of its cores. Accordingly, cores could become hot spots if multiple non-member sources existed, and the paths followed by the packets sent by those sources could be very inefficient due to router mobility in an ad-hoc network. Unlike other

protocols that allow non-member routers to send packets to a multicast tree for dissemination within the tree, CAMP requires that the router attached to any source of packets for the group join the multicast mesh. To avoid the dissemination of multicast packets to routers that join a group only to allow a source-only host to send packets to the group, CAMP allows routers to belong in a multicast mesh in simplex mode, rather than as regular members. This characteristic of mesh members is used during packet forwarding to avoid the dissemination of data to parts of the network where exclusively sender-only hosts are present.

CAMP ensures that all the reverse shortest paths between sources and receivers are part of a group's mesh by means of *heartbeat* and *push join* (PJ) messages. Periodically, every single entry in $CACHE$ is verified. The router looks up its RT to check whether the neighbor that relayed the packet is the reverse path to the source for every cache entry. A heartbeat or a PJ is sent towards every source stored in the cache that had the number of packets coming from the reverse path under a threshold. Push join requests may also be sent after a member gets an update message containing a list of active data traffic sources. If for any of the sources, the next-hop is not known to be part of the mesh a push join request will be sent.

A router receiving a heartbeat for a given multicast group and source retransmits the heartbeat if its successor towards the source of data traffic (determined with the unicast routing protocol) is already a mesh member. When a member router receives a heartbeat and detects that its successor is not part of the multicast mesh, it sends a *push join* (PJ) to that neighbor router and waits for an ACK from that router.

A router receiving a PJ forwards it to the next relay if: (a) it is the specified intended relay and (b) it has a path to the end point of the PJ. The relay specified in the forwarded PJ is the router's successor to the end point of the PJ. A router discards a PJ for which it is not the intended relay or for which it is the intended relay but has no path to the end point of the PJ.

A router that receives a PJ sends an ACK if: (a) it is the intended relay, (b) it is already a member of the group specified in the PJ, and (c) it has a path to the end point of the PJ. CAMP determines two types of push join acknowledgments — regular ACK, sent by duplex members and ACK_SIMPLEX, sent by simplex members. Given the fact that simplex mesh members do not accept packets coming from duplex members, it's important that there's no interleave of duplex and simplex routers between the initiator of a push join request and the router directly attached to the source. When acknowledgments start coming back from the source, duplex members will always send regular ACKs, and simplex members will become duplex when they receive a regular ACK. Therefore, if there's at least one duplex mesh member in the path from initiator to the source, all nodes from that duplex member all the way to the initiator must become duplex if they're not yet.

Receivers use a slightly different procedure to leave a multicast group in CAMP than in CBT. A router leaves a multicast group when it is not attached to any hosts that are members of the group *and* it has no neighbors for whom it is an *anchor*.

A router leaving a multicast group issues a *quit notification* to its neighbors, who in turn can update their MRTs. No acknowledgments are requested for quit notifications, because in contrast to multicast routing trees, multicast meshes do not dictate the paths taken by multicast packets. Quit notifications are sent as part of multicast routing updates.

6 Handling Mobility and Mesh Partitioning

6.1 Unreachable Cores

CAMP reduces control traffic associated with the establishment and maintenance of multicast meshes by using multiple cores per group that routers can use as *landmarks* to orient their join requests. Therefore, a router can try to join a mesh orienting its unicast join requests to any of such *landmarks* and can redirect its join requests when topology changes. If none of the cores of a group are reachable given the unicast routing information currently available when a router needs to send a join request, this router uses an Expanded Ring Search (ERS) to reach the mesh. This router first sends a mesh search message specifying itself as the requester. Any router receiving such a message forwards it appending its ID to the path of the message, if the ERS can proceed and the router is not a member of the mesh. A router that receives the mesh search message and is a mesh member replies with an acknowledgment. When the mesh search requester gets the first acknowledgment to its message, it sends a *join* request along the path it obtained with the acknowledgment. The router retransmits its search message after a time out if it does not receive an ACK.

Hence, CAMP has no single point of failure and can use as many cores as desired for a given mesh. In contrast, CBT and PIM-SM require a single core to be used in order to provide a multicast tree at all (i.e., avoid to detect loops in the multicast tree and detect partitions). ERS could be used when the single core fails, of course, but that still leaves CAMP as a more efficient approach, because ERSs are used less often by providing multiple cores (no single point of failure). A proposal to accommodate multiple cores and still provide multicast trees has been made recently [21], but the mechanisms in such a proposal appear much too complex for a dynamic network and no similar solutions have been proposed for ad-hoc networks.

6.2 Keeping Meshes Connected

A multicast mesh may be partitioned due to the mobility of routers or the partition of the network itself. In such a case, CAMP has the ability to continue the operation of all mesh components, because routers do not rely on a single core to join the mesh. In any tree-based protocol based on receiver-initiated joining, the tree component including the core or rendezvous point can continue to operate, but the other must terminate the multicast group (or make use of ERS, for example, for every join request) until a path to the core is re-established.

In addition, CAMP is able to merge mesh components as long as there is physical connectivity between mesh components. To ensure that two or more mesh components with cores eventually merge, the protocol requires (a) routers to periodically check if the next-hop to its selected core is part of the mesh, and (b) that all active cores in the mesh send periodical messages to each other, forcing routers along the path that are not members to join the mesh.

The periodic verification of the membership of the next-hop to the selected core is done by all members. If the next-hop to the core is not known to be a mesh member, a duplex or simplex join request is sent towards the core again. The verification is needed because the mobility of the nodes can place non-member routers for instance between receivers, senders and cores. Checking

periodically the membership of the next-hop to each sender is not feasible because CAMP does not keep track of senders. Therefore, making sure the reverse path to cores are part of the mesh provides at least a sub-optimal way for receivers to get packets generated by the senders in the group.

The messages exchanged among cores are *core explicit joins* (CEJ) that specify the multicast group, the intended relay of the CEJ, the intended core, and a gap flag. The flag is an information used by the receiver of a CEJ to determine whether there are non-members in the path between two cores. When the flag is kept reset all along the path between the two cores, no acknowledgment to CEJ needs to be sent back.

A router receiving the CEJ with the gap flag set to 0 forwards the CEJ to the next relay if (a) it is the specified relay and (b) it has a path to the specified core. Furthermore, if the relaying router is not a member of the mesh, it sets the gap flag to 1 in its CEJ.

A core receiving the CEJ with the gap flag set to 1 sends an ACK. The ACK is forwarded all the way back to the core that originated the CEJ; ACKs force relaying routers to join the mesh as in a PJ or a regular join. Alternatively, a router receiving the CEJ with the gap flag set to 0 forwards the CEJ to the next relay if: (a) it is the specified relay, (b) it has a path to the specified core, and (c) it is not a member of the group.

The possibility of using multiple cores in a multicast group provides fault tolerance to the protocol. But CAMP is able to merge mesh components even when no core is reachable. As explained previously, in such exceptional situations, a node uses expanded ring search when it needs to join the mesh. When a search fails because no mesh member is close enough, the originator of the search request backs-off, checks again for the availability of a core and if still no one is found, the originator becomes a transient core, as explained in Section 4. As the existence of one or more transient cores is propagated to the network, core-explicit join requests start being exchanged and different components eventually merge again.

7 CAMP Correctness

In this section we show that, if there is physical connectivity among all the members of a multicast group, CAMP builds connected multicast meshes over which multicast packets flow without loops, and that it builds connected mesh components when the wireless network becomes partitioned. A mesh component is a set of routers that belong to the same multicast group, such that data can flow bidirectionally between any pair of routers in the group. We assume that the set of static cores of any multicast group is known to all routers in the wireless network, and that the link layer supports the reliable delivery of CAMP control packets and the identification of the neighbors of a router.

The following theorem shows that data packets do not traverse loops in routers running CAMP, under the assumption that the time-to-live (ttl) of a data packet is

Theorem 1 *No multicast data packet is accepted or forwarded more than once by any given router running CAMP.*

Proof: From the definition of data forwarding in CAMP, router i discards any incoming packets

whose header information (i.e., the address of the packet's source node, the sequence identification assigned by this source and a packet fragment offset) is already available in $CACHE_i$. We assume router i is able to use the information available in the unicast routing table to assess the longest delay (LD) in the network. Any router i in the mesh keeps each packet header in $CACHE_i$ for a time $t > LD + \xi$, where $\xi > 0$. Therefore, any incoming multicast data packet that was already seen by router i must have its header information in $CACHE_i$ even when it came back to router i over the path that incurs the longest delay, and will be discarded. Consequently, data packets are never processed more than once by a router.□

Because of node mobility, changes in group membership, and link failures, a multicast mesh need not be connected at every instant. In terms of the connected components of a mesh and its cores, we can define a multicast mesh M as $M = K \cup X$, where K is the set of mesh components with at least one core router and X is the set of mesh components without any core. There are four possible cases for the values of $|K|$ and $|X|$ in terms of the number of mesh components and the cores in them, namely:

1. $|K| = 1$ and $|X| = 0$: The mesh is not partitioned into components and has at least one core.
2. $|K| > 1$ and $|X| = 0$: There are multiple mesh components with at least one core.
3. $|K| \geq 1$ and $|X| \geq 1$: There are mesh components both with and without cores in them.
4. $|K| = 0$ and $|X| \geq 1$: The mesh is partitioned into components, and none of them have cores.

For each of the above cases, we need to show that CAMP builds mesh components within a finite time, and that components with physical connectivity among one another through routers not in the mesh reassemble into a larger component within a finite time.

The following lemma shows that CAMP permits a source of data packets outside a multicast group to send data to the group by means of a simplex join.

Lemma 1 *A simplex join request has no deadlocks and finishes in a finite time after the network stops changing.*

Proof: We need to show that no router stays with a simplex join pending indefinitely. More specifically, we need to show that the initiator x of the request, for whom $PEND_x^g = x$, becomes a simplex member of the multicast group g or returns back to the non-member state.

After initiator x sends out the request, the request travels through none or at least one non-member router, and there are only five possible cases to consider:

1. The request travels all the way to the selected core.
2. It travels hop-by-hop to a duplex or simplex mesh member.
3. It reaches a non-member or a simplex router z with a pending join request, i.e., $PEND_z^g \neq \emptyset$.

4. The request reaches a non-member router z with another simplex join request pending.

5. Either the request or its acknowledgment is lost by one of the intermediate routers in the path between the request initiator and the selected core.

In the first two cases, an acknowledgment to the simplex join request is sent back to router x, which then becomes a simplex mesh member. In the following two cases, an acknowledgment is sent towards the initiator as soon as the router z gets its own acknowledgment for its original request. If this original request was of duplex type, router z will send an acknowledgment of duplex type. After receiving this duplex acknowledgment, the very next router y acting on behalf of router x and with a pending simplex request due to router x's request will become a simplex member, even though what was received was a duplex acknowledgment. Clearly, in the cases above, nothing prevents the mesh to be extended and router x to become a simplex member. Finally, in the last case, the pending join simplex request times out and is sent again; the request is retransmitted only by the initiator, router x, for a finite number of times. If all retransmissions time out and do not succeed, router x goes back to non-member status.

The path built by a join simplex request for a router x as the request traverses the network is finite and simple, i.e., no router repeats itself on the path. Denote the routers on the path by (y_1,...,y_p). The path either ends when y_p is (a) the selected core, (b) a router other than the selected core that is already a member of the mesh or a node with a pre-existing pending join/join simplex request, or (c) the router that had its join request or acknowledgment lost. In the first two cases, the router y_p must give a reply within a finite time; this implies that router y_{p-1} receives an acknowledgment in a finite time. By induction on the number of routers on the path a simplex join request traverses, every router y_i, where $1 \leq i < p$, and x get a reply to its request within a finite time. In cases 3 and 4, as soon as router z gets an acknowledgment to the pending request, this router will become a member, and these cases proceed as case 2. In the last case, all y_i will have their pending simplex join requests timed out. If after a finite number of retransmissions, router x does not succeed in its join request, it will go back to its non-member status. Thus, y_i become mesh members or stay non-members in a finite time.

Therefore, because every initiator of a simplex join request becomes a mesh member or goes back to non-member status, a request of this type does not have deadlocks and finishes in a finite time. □

Now we need to show that CAMP allows receivers to join the multicast mesh. If the router closest to the receiver host is already a duplex member, this router does not need to send any requests. If the router has neighbors that are already members of the group, the router can announce becoming a member. However, if that router is not a mesh member yet or just a simplex member, and if the router does not have neighbors that are members of the group, then the router must send a join request towards a selected core. We need to show that such request completes in a finite time and that CAMP does not deadlock in the event that the request cannot be completed successfully, allowing the router closest to the receiver host to send the request again if it is still appropriate.

Lemma 2 *A join request has no deadlocks and finishes in a finite time after the network stops changing.*

Proof: As in Lemma 1, we need to show that no router stays with a join request pending indefinitely. More specifically, we need to show that the initiator x of the request, for whom $PEND_x^g = x$, becomes a duplex member of the multicast group g or returns back to its previous state, non-member or simplex member.

After initiator x sends out the request, the request travels hop-by-hop through none or at least one non-member/simplex router. There are only 4 possible cases:

1. The request travels all the way to the selected core.
2. It travels to a duplex mesh member.
3. It reaches a non-member or a simplex router z with a pending join request, i.e., $PEND_z^g \neq \emptyset$.
4. Either the request or its acknowledgment is lost by one of the intermediate routers in the path between the request initiator and the selected core.

In the first two cases, an acknowledgment to the simplex join request is sent back to router x, which then becomes a simplex mesh member. In the third case, an acknowledgment is sent towards the initiator as soon as the router z gets its own acknowledgment for its original request. Router z will become a duplex member and send an acknowledgment. Clearly, in the cases above, nothing prevents the mesh to be extended and router x to become a simplex member.

In the last case, the pending join request will time out and will be sent again. The request is retransmitted only by the initiator, router x, for a finite number of times. If all retransmissions time out and do not succeed, router x goes back to its previous membership status.

Just as in a join simplex request, the path built by a join request for a router x as the request traverses the network is finite and simple, i.e., no router repeats itself on the path. The proof for that claim is the same as the one shown in Lemma 1. Therefore, because every initiator of a join request becomes a mesh member or goes back to its previous status, a request of this type does not have deadlocks and finishes in a finite time. □

After a receiver joins a mesh, it periodically receives advertisements sent from active traffic sources, which are flooded within the mesh. When receiving one of such advertisements, a router determines if its successor to the given source is already part of the multicast mesh. If not, a push join request is sent in the direction of this source to incorporate the shortest path to the mesh. To show that a push-join request does not deadlock, we require the definition of the *no-type-interleave* property.

Definition 1 *Consider a router* r*, a data traffic source* s *and an intermediate router* i *in the path from* r *to* s*. A path of routers between* r *and* s *does not have a random sequence of duplex and simplex routers and is said to have the* no-type-interleave *property iff:*

1. *All routers in the path have the same membership type, either duplex or simplex, as router* r*, or,*

2. *All routers from* s *to* i*, including* i *are simplex members, and all routers from* i *to* r *are duplex members, including* r.

Because a multicast mesh as defined by CAMP may include simplex and duplex members, and given the fact that simplex members are not allowed to accept data packets from duplex members, in order for a receiver host connected to router r get multicast data packets from a source s, the path between r and s must have the *no-type-interleave* property.

Lemma 3 *A push join request has no deadlocks and finishes in a finite time after the network stops changing.*

Proof: Consider the path which consists of all routers in the path between the router that is directly connected to the data traffic source host s and the initiator x of the push join request, for whom $PENDPJ^g_{x,s} = x$. In order to prove this lemma, we need to show that x does not wait indefinitely for the request to complete and that the path between x and s has the no-type-interleave property in a finite time.

After initiator x sends it out, the request travels hop-by-hop towards the intended source s. The possible cases are:

1. The request travels across all nodes, regardless of their membership type, all the way to router d directly connected to the source.
2. It reaches a router z with a push join request already pending for source s, i.e., $PENDPJ^g_{z,s} \neq \emptyset$.
3. Either the request or its acknowledgment is lost by the intermediate router i in the path between x and s.

In the first case, if d is duplex, a duplex acknowledgment is sent. All simplex nodes in the path will receive, forward the duplex acknowledgment towards x and become duplex.

But if d is simplex, it will send a simplex acknowledgment towards x. If there is an intermediate node i between x and s and that node is the closest duplex member to s, node i will not forward a simplex acknowledgment but a duplex one instead. All nodes between x and i which are not duplex already will become duplex members because each node in this path forwards the duplex acknowledgment.

In the second case, an acknowledgment to request originator x will be sent as soon as router z gets an acknowledgment for the push join request that was previously pending. The type of acknowledgment sent by router z will depend on its membership type after handling the feedback for the previous request. If duplex, router z will send a duplex acknowledgment, and all routers between z and x will be duplex. If router z is simplex, an acknowledgment of this type will be sent and forwarded until it reaches either originator router x or another duplex member.

It is clear from these two cases that a path between the push join request originator and the traffic source does not randomly interleave or alternate routers of different types, and therefore the no-type-interleave property applies.

In the last case, the push join request pending at the originator will time out and will be sent again. The request is retransmitted only by the initiator, router x, for a finite number of times. If all retransmissions time out and do not succeed, router x removes the pending push join request to sender s from the list. If needed, a new push join request to this sender will be triggered again by the protocol.

The path between the push join request originator router x and the source of data traffic is finite and simple, i.e., no router repeats itself on the path. Denote the routers on the path by (y_1,...,y_p). The path starts with y_1, which is the router that is the closest to x, and ends with y_p, which is the router that is the closest to s. When the push join request reaches y_p, it must provide a feedback in a finite time. This implies that the router y_{p-1} will receive an acknowledgment in a finite time. By induction on the number of routers on the path between x and s, x gets a feedback to its request within a finite time. If either an acknowledgment or the request itself gets lost in the path, router x will not receive a feedback before its timer expires and will retransmit the request for a finite number of times.

Therefore, a successful push join request forms a path with no-type-interleave between originator and traffic source without deadlocks and in a finite time. When not successful, the original request does not prevent the protocol to try again later if appropriate. □

Theorem 2 *When the multicast mesh has at least one core and is not partitioned into multiple components, CAMP allows traffic sources to send data, receivers to join the mesh and incorporate shortest-paths to sources in the mesh.*

Proof: Lemmas 1, 2 and 3 have shown that a join, a simplex join, and a push join requests have no deadlocks and finish in a finite time. Therefore, the theorem is true. □

The next relevant mesh partitioning scenario is Case 2, where the mesh is broken into $|K|$ different components, with $|K| > 1$, and all components have at least one core. The mechanisms for sources and receivers to join the mesh are just like the ones in Case 1. But because the mesh is partitioned, data traffic or advertisements from senders located in other components will not be received locally, and consequently shortest-paths to these senders cannot be incorporated to the mesh before components merge. For this and similar scenarios, CAMP requires the periodic exchange of messages among all cores. This exchange consists of *core-explicit-join* (CEJ) requests, whose role is to force all routers between each pair of cores to join the mesh. In a given multicast group g, an active core, which has received requests from routers willing to join the multicast group, will send CEJs to the other cores in $CORES_g$ in a round-robin fashion, that is, every time to a different core. This is to prevent a core to always send CEJs to other cores in the same component.

Therefore, in order to show that CAMP allows mesh components to merge, we need to show that core-explicit join requests do not deadlock and finish in a finite time.

Lemma 4 *A core-explicit-join (CEJ) request has no deadlocks and finishes in a finite time after the network stops changing.*

Proof: A core-explicit join request uses the same mechanism as a join simplex request (Lemma 1), in the sense that it is a unicast communication between two nodes. The major difference is the

fact that the originator of the request is an active core rather than a regular router willing to join the multicast group. Furthermore, CEJs always go from one core to the other, and feedback comes only from the destination core and not by mesh members in between. Therefore, a CEJ has no deadlocks and finishes in a finite time. □

Theorem 3 *When the multicast mesh is partitioned into multiple components and all of those components have at least one core, CAMP allows components to merge, traffic sources to send data, receivers to join the mesh and incorporate shortest-paths to sources in the mesh.*

Proof: Lemmas 1 to 4 have shown that join, join simplex, push join and core-explicit join requests have no deadlocks and finish in a finite time. Therefore, the theorem is true. □

7.1 Mesh Components with and without Core Routers

Case 3 is the next relevant mesh partitioning scenario that needs to be considered to prove the correctness of CAMP. In this scenario, there are mesh components K_i, where $1 \leq i \leq r$ and $r \geq 1$, with at least one core and other components X_j, where $1 \leq j \leq s$ and $s \geq 1$, that do not have any core. For any value of r, proofs of Case 2 also apply for K here.

The fact that no core is available for any given component X_i does not prevent receivers and sources in such a mesh component from joining the multicast group. Either cores available in K or members already existing locally can validate the join request. Although any successful join request originated in a component X_i and sent to a component K_j will cause those components to merge, this does not guarantee the merging of different components when there are no more senders or receivers to join the multicast group.

Theorem 4 *When the multicast mesh is partitioned into multiple components and some of those components do not have at least one core, CAMP allows components to merge, traffic sources to send data, receivers to join the mesh and incorporate shortest-paths to sources in the mesh.*

Proof: It has been shown that join, join simplex, push join and core-explicit join requests have no deadlocks and finish in a finite time. CEJ requests will merge components in K, but not components in X because these do not have core routers available.

In order to allow different components in a multicast group g to merge in situations like that, CAMP requires every duplex or simplex router i to look up its MRT_i^g and check the membership status of the neighbor that works as the next hop to the selected core. This check is periodically done and also every time the beaconing protocol signals CAMP that a neighbor was added to or removed from the local database. If the router i is duplex, a join request will be sent if the next hop is either a simplex member or non-member (not available in MRT_i^g). If the router i is simplex, a join simplex request will be sent only if the next hop is not a mesh member.

Therefore, in a component X_i, where no core is available locally, there will be at least one router r who will need to send either type of join request due to the membership type of its next hop to its selected core. From Lemmas 1 and 2, we know both join simplex and regular join requests

don't incur in deadlocks and finish in a finite time. So the first successful join request originated by router r will move its component X_i from X after it merges with a component in K. This same process will happen to every component in X, and when the network stops changing, we'll be back to Case 2. Therefore, the theorem is true. □

7.2 No Mesh Components with Core Routers

Finally, Case 4 is the last of the relevant scenarios for mesh partitioning in CAMP. Here, the mesh has one or multiple components, and none of them have a core available. This situation is expected to happen rarely, and should be seen as exceptional.

Theorem 5 *When the multicast mesh is not partitioned or is partitioned into multiple components and none of the components have at least one core, CAMP allows components to merge, traffic sources to send data, receivers to join the mesh and incorporate shortest-paths to sources in the mesh.*

Proof: The absence of cores prevents any router to send join requests of any type. In components with senders, data packets can still be sent locally, and therefore push join requests can be generated also locally. Besides the validation of join requests, recall that the reverse path to cores is periodically checked by members to prevent mesh partitioning.

When either join requests or verification of reverse-path to core fail to find a reachable core, the router i that originates the request or verification may become a transient core. Specifically in simplex or duplex requests, router i must try to join the mesh using Expanded Ring Search (ERS) first. But when everything else fails, as explained in Section 4, the router backs off for a given amount of time, and if no other core has become available in the meantime, router i will become a transient core. When the first transient cores start becoming available, their IDs will be propagated to the entire network, and the proofs for the previous cases will apply. Therefore, the theorem is true. □

8 Performance Comparison

8.1 Experiments

In large ad-hoc networks, no multicast protocol proposed to date that is based on sender-initiated joining is scalable with the number of nodes in the network or the number of sources and groups in the network. Examples of this type of protocols based on routing trees are DVMRP and PIM-DM; an example of this type of protocols based on graphs other than trees is FGMP and ODMRP. The reason that these protocols are not scalable is that sources must frequently flood either data packets or control packets to *all* the network in order to establish a routing structure. If the network size is large, or the number of groups and sources per group is large, this approach is not applicable. In order to confirm that, we implemented one of the flood-based mesh approaches, ODMRP, to observe how it compares to CAMP in a network with mobile nodes.

In essence, ODMRP requires that all senders that are active transmitting data packets periodically flood the network with a sender advertising packet. All routers directly connected to hosts willing to participate in the multicast group will process those advertising packets, and update a member table. This table lists all senders whose advertisements were received and the neighbor routers used as next hop toward those senders. Periodically and when new sender advertisements are received, the member table is also broadcast, and intermediate routers listed in member tables as next hop to a sender will set a data forwarding flag, become group members and keep/broadcast a member table themselves. Just like CAMP, ODMRP keeps a data packet cache. Data packets are forwarded if the forwarding flag is set and the data packet is not already in the packet cache. FGMP is very similar to that approach, except for the fact mentioned above that receivers are the entities that flood membership advertisement packets, and senders keep track of receivers in the member table. Both ODMRP and FGMP have scalability problems because of the design decision to flood control packets, and specially FGMP due to the fact senders have to keep track of all receivers in a multicast group. Our simulation results illustrate the scaling problems of the mesh approach used by ODMRP.

The interesting aspects for performance comparison between CAMP and other multicast protocols are the average delays, percentage of packet loss incurred due to node mobility, and the number of control packets received by each node. The percentage of packets lost at a receiver is simply the amount of packets sent by the traffic source that was not seen by the specific receiver. Therefore, the smaller the percentage is, the better the protocol behaves. Obviously, the average packet delay measured at each receiver excludes lost packets.

We ran a number of experiments to study this aspect of CAMP's performance and to compare it against the other multicast approaches. Figure 2 shows the topology of the dynamic network used in the simulations. The network has 30 routers, numbered from 1 to 30, and two senders, A and B. The links shown in the diagram illustrate the initial mesh connectivity in the simulation, i.e., the existence of a link between two nodes imply there's bidirectional connectivity between these two nodes. All nodes in the simulation of the multicast routing protocols are receivers. In CAMP, this means all nodes are *duplex* members. *Router 16* was chosen as core for all simulations.

Experiments ran for 350 seconds and the same conditions were applied to the simulation runs for CAMP and ODMRP; specifically, the same number of packets was sent from the given source, the same pattern of router mobility was applied, and the same MAC and routing protocols were used. The simulations used a single broadcast channel, so that the transmission of a node is received by all its neighbors. The channel bandwidth is 1 Mbit/sec. The Floor Acquisition Multiple Access (FAMA) protocol[10] was used to access the broadcast channel, and the wireless internet routing protocol (WIRP) [13] with *hop count* as distance metric was used to generate the unicast routing-table entries at routers for CAMP. ODMRP does not need a unicast routing protocol. Radio links are bidirectional. The timers of updates in CAMP and sender advertisement in ODMRP determine how fast the network adapts to topology and group membership changes. Although the draft specification available for ODMRP [14] requires this timer to be set to 400 msecs and does not clearly indicate a way to compute this timer for different network sizes and capacity, the update timers for both protocols are set to three seconds. This is an attempt to be fair to this protocol, since 3 seconds is the period used by CAMP to send updates. Naturally, if its timers are set to 400 msecs, we expect ODMRP to yield a much greater overhead than the one shown here.

Two sets of experiments were run regarding mobility and mesh membership. In the first, 15 routers or 5 routers move through the network. All 30 nodes are receivers in one multicast group. The mobile nodes are represented in white in Figure 2. When only 5 routers are mobile, those are routers 17, 18, 20, 28 and 30. The speed at which mobile nodes moved randomly in all simulations was 67.5 miles/hr (30 meters/sec).

In this first set of experiments, data traffic is originated either by source A, which is directly attached to the core (*router 16*), or by both source A and B, which is attached to *router 29*. In the experiments where the source of data traffic is sender A, the load was 4 packets/second. In the experiments where both senders A and B transmitted packets, each one sent 2 packets/second to try to keep the same number of data packets in the network. Data packets are 500 bytes long.

In the second set of experiments, 15 nodes are mobile out of 30 nodes. The difference is that only six nodes are receivers in the multicast group — nodes 6, 9, 16, 20, 23 and 25. Please note that all receivers are mobile, except for router 16. In this second set, simulations had either only one source — source A attached to router 16, or 4 sources — besides A, sources attached to routers 6, 23 and 25, and each of them generate data traffic at a rate of 1 pkt/sec. Given the routers they're attached to, those routers are also mobile. They're not depicted in Figure 2, and are referred to as sources $s6$, $s16$, $s23$ and $s25$. All other simulation parameters are the same as in the first set of experiments.

8.2 Performance of Protocols

Figures 3 to 6 summarize the comparison between CAMP and ODMRP over the first set of experiments we have run. Dashed lines represent ODMRP and solid ones represent CAMP. Figure 3(a) shows that CAMP renders smaller delays than ODMRP in the case of a single source sending 4 packets/second and 15 nodes moving in the network. And the main reason for this difference in average is shown in Figure 3(b). The longer delays incurred in ODMRP is a consequence of the flooding of control packets per source needed in ODMRP. The number of control packets received by CAMP routers represent around 60—70% of the number seen by ODMRP routers. For the same traffic load and mobility patterns, both protocols perform similarly, when there's one source of data traffic, as shown by Figure 3(c). As the number of sources grows, CAMP performs even better than ODMRP, as shown in Figure 4. In Figure 4(a), one can observe that, like routers 1, 2 and 3, almost half of the routers in the network show shorter delays for both senders A and B when running CAMP. As illustrated by Figure 4(c), as far as packet losses are concerned, CAMP loses consistently fewer packets when more than one source send data packets.

In the case where there's only 5 mobile nodes, the trend is similar to the case above, as shown by Figures 5 and 6. Regarding packet losses, with the exception of some routers in Figure 6(c), CAMP routers tend to consistently loose slightly fewer packets than their ODMRP counterparts, when two senders transmit data packets. In Figure 6(c), some of the CAMP routers lost considerably more packets. Routers 1, 2, 10, 13, 20 and 30 start the experiments all located in the upper left corner of the network, as shown by Figure 2. Some CAMP updates were lost and intermediate routers took longer to start acting as anchors for that part of the network. Routers 18 and 28 are initially in the same network area, but are not as negatively impacted, because early on during the simulation run they move to other parts of the network.

The results for the second set of experiments are shown from Table 1 to Table 6. The numbers show how the protocols perform when 15 nodes are mobile in the network of 30 nodes, and only 6 nodes are receivers. The comparison on these tables is based on the column *ratio* for average data packet delay and incoming control packets, and on the column *difference* for percentage of missed packets. The ratios are always computed by the division of the figure yielded by CAMP by the figure yielded by ODMRP. The difference is computed by subtracting the result obtained by ODMRP from the result obtained by CAMP.

Tables 1 to 3 show average packet delay, percentage of packet losses and the number of incoming control packets for the case where only the sender attached to router 16 generates data packets. Although there's a difference in the volume of control packets as shown in Table 3, this does not impact the network, which is able to absorb the overhead for both protocols. CAMP and ODMRP have very similar delays for all receivers, as the *ratio* column in Table 1 shows. Because of a combination of lost CAMP updates and slow convergence of the unicast routing protocol, receivers 6 and 20 lose much more data packets when running CAMP. Specifically for the case of receiver 20, the slow convergence of WIRP, in two different moments during the simulation, prevented push join requests to be sent by not providing receiver 20 with a next hop to the sender. The problems with the loss of update messages in the two experiments indicate that CAMP can improve its performance with a reliability mechanism for this type of control messages.

When the number of sources of data traffic was increased to four, the protocol overhead grew proportionally for both protocols as shown by Table 6. But now the volume of control packets of ODMRP brings down the performance of the receivers regarding average packet delay. In this experiment, the delays started to show considerable differences in favor of CAMP. The receivers when running CAMP see delays that represent 52% to 78% of the delays when they run ODMRP, as shown in Table 4. All receivers consistently see smaller delays when they run CAMP. Table 5 does not show a considerable difference regarding packet losses.

9 Conclusions

We have presented and verified the core-assisted mesh protocol (CAMP) for the establishment and maintenance of multicast meshes over which multicast routing can be accomplished efficiently in wireless mobile networks. CAMP establishes multicast meshes by means of sets of cores to allow group members to join the meshes without flooding data or control packets throughout the wireless network. We compared CAMP against ODMRP, which also establishes multicast meshes but requires flooding an entire wireless network with control packets to set up its routing structure. Our simulation experiments show that the receiver-initiated approach used for mesh joining in CAMP performs and scales better than the sender-initiated approach of ODMRP.

References

[1] A. Ballardie, P. Francis, and J. Crowcroft, "Core Based Trees (CBT): An Architecture for Scalable Inter-Domain Multicast Routing," *Proc. ACM SIGCOMM'93*, pages 85–95, October 1993.

[2] E. Bommaiah, M. Liu, A. McAuley, and R. Talpade, “AMRoute: Adhoc Multicast Routing Protocol,” Internet Draft, http://www.ietf.org/internet-drafts/draft-talpade-manet-amroute-00.txt

[3] C. Chiang and M. Gerla, “On-Demand Multicast in Mobile Wireless Networks,” *Proc. IEEE ICNP 98*, Austin, Texas, October 14-16, 1998.

[4] M.S. Corson and S.G. Batsell, “A Reservation-Based Multicast (RBM) Routing Protocol for Mobile Networks: Overview of Initial Route Construction,” *Proc. IEEE INFOCOM ’95*, Boston, MA, April 1995.

[5] M.S. Corson and V. Park, “A Highly Adaptive Distributed Routing Algorithm for Mobile Wireless Networks,” *Proc. IEEE INFOCOM ’97*, Kobe, Japan, April 1997.

[6] Y.K. Dalal and R.M. Metcalfe, “Reverse Path Forwarding of broadcast packets.” *Communications of the ACM*, 21(12):1040-1048, December 1978.

[7] S. Deering, “Host extensions for IP multicasting,” RFC-1112, August 1989.

[8] S. Deering, C. Partridge and D. Waitzman, “Distance Vector Multicast Routing Protocol,” RFC-1075, November, 1988.

[9] S. Deering et al. “An Architecture for Wide-Area Multicast Routing,” *Proc. ACM SIGCOMM’94*, Cambridge, MA, 1994.

[10] C. L. Fullmer and J.J. Garcia-Luna-Aceves, “Solutions to Hidden Terminal Problems in Wireless Networks,” *Proc. ACM SIGCOMM 97*, Cannes, France, September 14-18, 1997.

[11] J.J. Garcia-Luna-Aceves and E. Madruga, “A Multicast Routing Protocol for Ad-Hoc Networks”, *Proc. IEEE INFOCOM’99*, New York, NY, March, 1999.

[12] J.J. Garcia-Luna-Aceves and E. Madruga, ‘The Core-Assisted Mesh Protocol”, accepted for publication in *IEEE Journal on Selected Areas in Communications*, Special issue on Ad-Hoc Networks, 1999.

[13] J.J. Garcia-Luna-Aceves, C.L. Fullmer, E. Madruga, D. Beyer and T. Frivold, “Wireless Internet Gateways (WINGS)”, *Proc. IEEE MILCOM’97*, Monterey, California, November 2–5, 1997.

[14] M. Gerla, et al., “On-Demand Multicast Routing Protocol,” Internet Draft, http://www.ietf.org/-internet-drafts/draft-ietf-manet-odmrp-00.txt, November, 1998.

[15] G.S. Lauer, “Packet Radio Routing,” *Chapter 11. Routing in Communications Networks.* (M.Streenstrup, ed.), Prentice-Hall, Englewood Cliffs, NJ, 1995.

[16] S. Murthy and J.J. Garcia-Luna-Aceves, “An Efficient Routing Protocol for Wireless Networks,” *Mobile Networks and Applications*, ACM/Baltzer, Vol. 1, No. 2, 1996.

[17] M. Ohta and J. Crowcroft, “Static Multicast,” Internet Draft, http://www.ietf.org/internet-drafts/draft-ohta-static-multicast-01.txt, October, 1998.

[18] C.E. Perkins, “Mobile networking in the Internet,” *Mobile Networks and Applications*, ACM/Baltzer, Vol. 3, 1998, p319-334..

[19] C.E. Perkings and P. Bhagwat, “Highly Dynamic Destination-Sequenced Distance-Vector Routing (DSDV) for Mobile Computers.” *Proc. ACM SIGCOMM’94*, London, UK, 1994.

[20] C.E. Perkins and E.M. Royer, “Ad Hoc On Demand Distance Vector (AODV) Routing,” Internet Draft, http://www.ietf.org/internet-drafts/draft-ietf-manet-aodv-01.txt, August, 1998.

[21] C. Shields and J.J. Garcia-Luna-Aceves, “The Ordered Core-Based Tree Protocol.”*Proc. IEEE INFOCOM ’97*, Kobe, Japan, April 1997.

[22] M. Spohn and J.J. Garcia-Luna-Aceves, “Scalable Link-State Internet Routing,” *Proc. VI IEEE International Conference on Network Protocols (ICNP’98)*, Austin, TX, October, 1998.

[23] C.W. Wu, Y.C. Tay and C-K. Toh, “Ad-hoc Multicast Routing protocol utilizing Increasing id-numbers (AMRIS): Functional Specification,” Internet Draft, http://www.ietf.org/internet-drafts/draft-ietf-manet-amris-spec-00.txt, November, 1998.

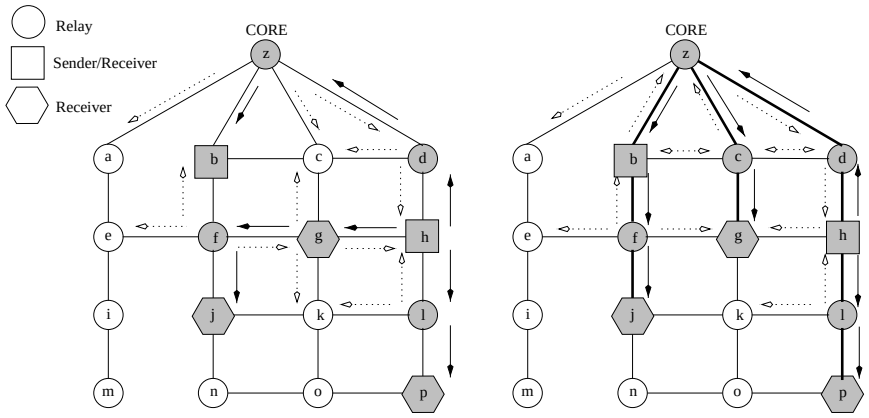


Figure 1: Traffic flow from router h in a multicast mesh (left) and in the equivalent multicast shared tree (right).

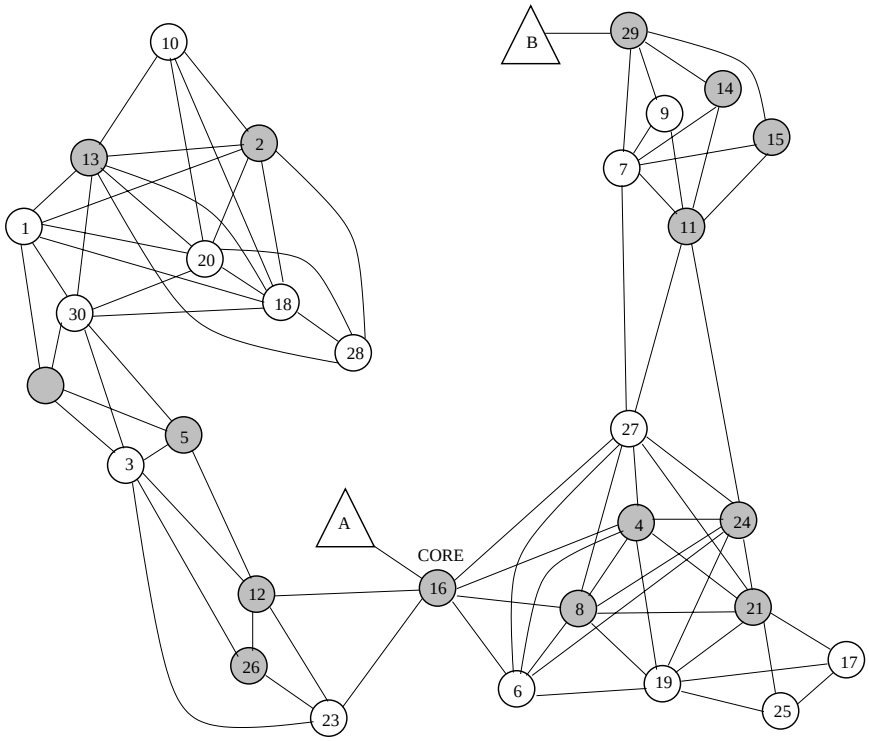
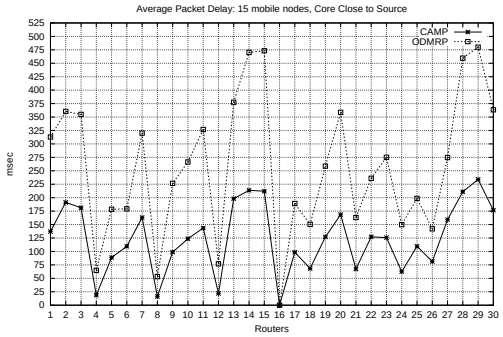
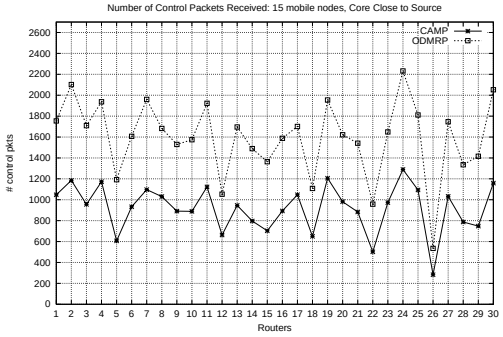


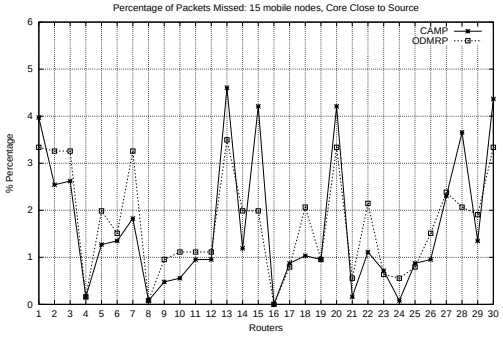
Figure 2: Network topology used in experiments: Lines connecting nodes indicate the initial mesh connectivity, all routers are receivers of the multicast group, Router 16 is the core, and A and B are sources. Mobile nodes are indicated in white.



(a)

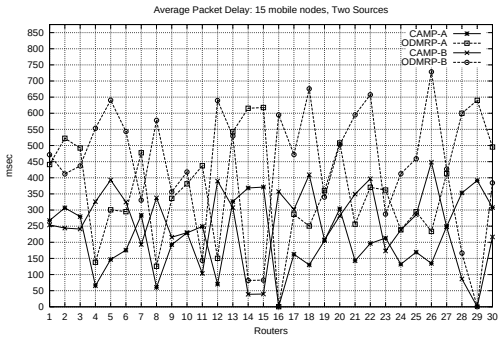


(b)

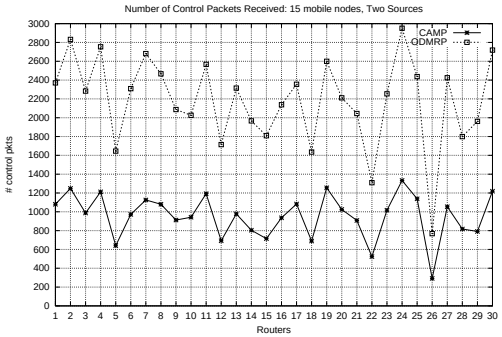


(c)

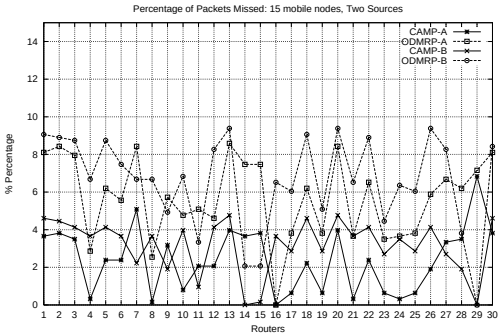
Figure 3: Average packet delay (a), number of incoming control packets (b) and percentage of missed data packets (c) for routers in a network of 30 nodes, where 15 of these nodes are mobile. Data traffic from source A, which is the one close to the core.



(a)

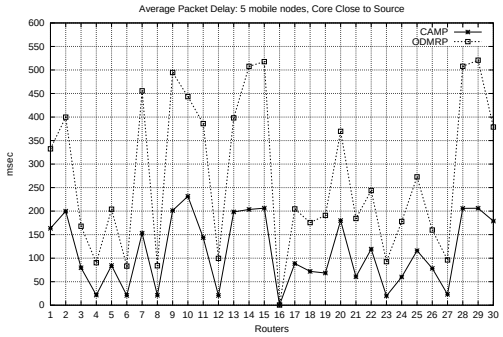


(b)

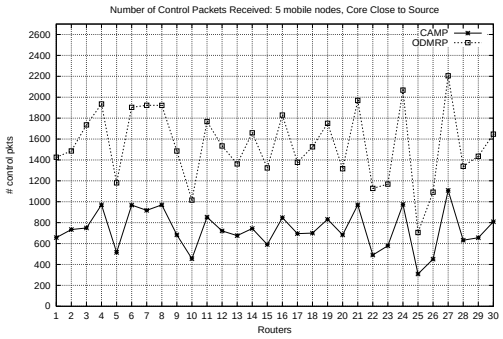


(c)

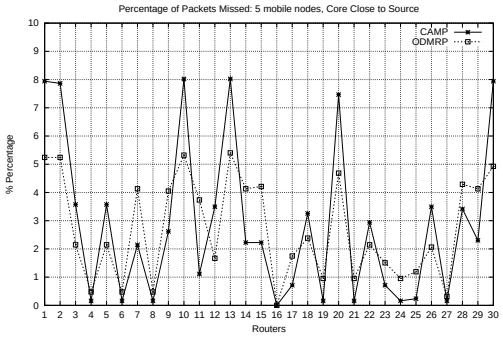
Figure 4: Average packet delay (a), number of incoming control packets (b) and percentage of missed data packets (c) for routers in a network of 30 nodes, where 15 of these nodes are mobile. Data traffic from both sources A and B. In this figure, "CAMP-A" represents the results for routers running CAMP due to traffic from Source A.



(a)

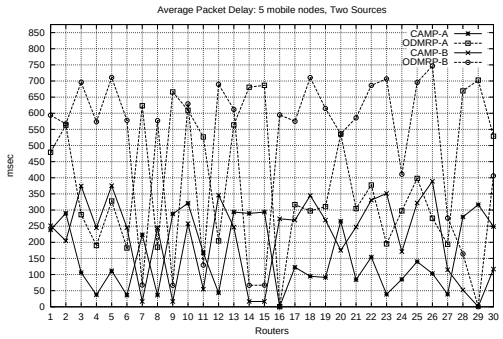


(b)

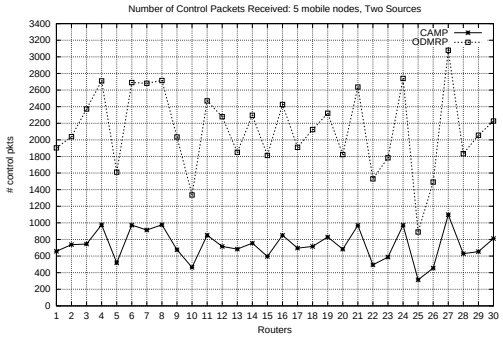


(c)

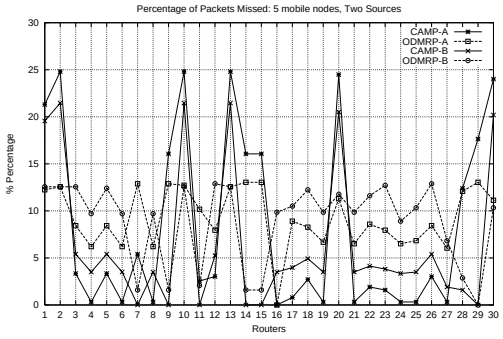
Figure 5: Average packet delay (a), number of incoming control packets (b) and percentage of missed data packets (c) for routers in a network of 30 nodes, where 5 of these nodes are mobile. Data traffic from source A, which is the one close to the core.



(a)



(b)



(c)

Figure 6: Average packet delay (a), number of incoming control packets (b) and percentage of missed data packets (c) for routers in a network of 30 nodes, where 5 of these nodes are mobile. Data traffic from both sources A and B. In this figure, "CAMP-A" represents the results for routers running CAMP due to traffic from Source A.

	s16		
receiver	camp	odmrp	ratio
6	107	98	1.09
20	168	170	0.99
23	99	96	1.03
25	91	93	0.99
9	89	102	0.87
16	0	0	1.00

Table 1: Average packet delay seen by receivers for traffic coming from source *s16*, directly attached to router 16. The ratio is $camp/odmrp$.

	s16		
receiver	camp	odmrp	diff
6	25.89	4.69	21.21
23	4.37	4.13	0.24
25	10.64	2.54	8.10
9	0.95	4.29	-3.34
16	0.00	0.00	0.00
20	44.24	9.61	34.63

Table 2: Percentage of data packets missed by receivers for traffic coming from source directly attached to router 16. The difference is $camp - odmrp$.

	control packets		
receiver	camp	odmrp	ratio
6	213	1137	0.19
23	225	1130	0.20
25	290	1262	0.23
9	248	1052	0.24
16	579	1362	0.43
20	143	1004	0.14

Table 3: Total number of incoming control packets at each receiver for multiple sources. The ratio is $camp/odmrp$.

	s16			s23			s25			s6		
receiver	camp	odmrp	ratio	camp	odmrp	ratio	camp	odmrp	ratio	camp	odmrp	ratio
6	243	331	0.73	340	508	0.67	302	463	0.65	0	0	1.00
20	324	524	0.62	446	625	0.71	461	653	0.71	210	403	0.52
23	245	341	0.72	0	0	1.00	317	465	0.68	385	525	0.73
25	272	350	0.78	258	434	0.60	0	0	1.00	295	457	0.65
9	230	366	0.63	293	492	0.59	355	530	0.67	447	678	0.66
16	0	0	1.00	245	385	0.63	266	406	0.65	220	365	0.60

Table 4: Average packet delay for multiple sources. The ratio is $camp/odmrp$. Source *s16* means the one attached to router 16.

	s16			s23			s25			s6		
receiver	camp	odmrp	diff	camp	odmrp	diff	camp	odmrp	diff	camp	odmrp	diff
6	4.14	9.24	-5.10	7.96	9.24	-1.27	7.32	12.42	-5.10	0.00	0.00	0.00
20	11.46	11.46	0.00	16.56	10.83	5.73	18.15	14.65	3.50	11.46	4.14	7.32
23	1.59	5.73	-4.14	0.00	0.00	0.00	4.46	9.24	-4.78	9.55	8.60	0.96
25	1.91	7.01	-5.10	3.82	7.01	-3.18	0.00	0.00	0.00	9.87	9.87	0.00
9	0.32	6.05	-5.73	5.41	10.19	-4.78	6.05	10.51	-4.46	10.51	10.51	0.00
16	0.00	0.00	0.00	4.14	6.05	-1.91	5.10	9.87	-4.78	9.24	7.96	1.27

Table 5: Percentage of data packets lost for multiple sources. The difference is $camp - odmrp$.

	control packets		
receiver	camp	odmrp	ratio
6	475	2832	0.17
23	552	2921	0.19
25	835	3140	0.27
9	366	2625	0.14
16	528	2842	0.19
20	257	2712	0.09

Table 6: Total number of incoming control packets at each receiver for multiple sources. The ratio is $camp/odmrp$.