WIRELESS COMMUNICATIONS AND MOBILE COMPUTING
Wirel. Commun. Mob. Comput. (2010)
Published online in Wiley Online Library (wileyonlinelibrary.com). DOI: 10.1002/wcm.1016

RESEARCH ARTICLE

An efficient mesh-based multicast routing protocol in mobile *ad hoc* networks

Eric Astier[1], Abdelhakim Hafid[1]* and Sultan Aljahdali[2]

[1] Network Research Lab, University of Montreal, Canada
[2] College of Computer Sciences and Information Systems, Taif University, Saudi Arabia

ABSTRACT

Mesh-based multicast routing protocols for mobile *ad hoc* networks (MANETs) build multiple paths from senders to receivers to deliver packets even in the presence of links breaking. This redundancy results in high reliability/robustness but may significantly increase packet overhead. This paper proposes a mesh-based multicast protocol, called centered protocol for unified multicasting through announcements (CPUMA), that achieves comparable reliability as existing mesh-based multicast protocols, however, with significantly much less data overhead. In CPUMA, a distributed core-selection and maintenance algorithm is used to find the source-centric center of a shared mesh. We leverage data packets to center the core of each multicast group shared mesh instead of using GPS or any pre-assignment of cores to groups (the case of existing protocols). The proposed centering scheme allows reducing data packet overhead and creating forwarding paths toward the nearest mesh member instead of the core to reduce latency. We show, *via* simulations, that CPUMA outperforms existing multicast protocols in terms of data packet overhead, and latency while maintaining a constant or better packet delivery ratio, at the cost of a small increase in control overhead in a few scenarios. Copyright © 2010 John Wiley & Sons, Ltd.

KEYWORDS
MANETs; multicast routing protocols; mobility

***Correspondence**
Abdelhakim Hafid, Network Research Lab, University of Montreal, Canada
E-mail: ahafid@iro.umontreal.ca

1. INTRODUCTION

Ad hoc networks are infrastructure-less, dynamically reconfigurable wireless networks that consist of nodes that act as routers. In such an environment, we face the problem of providing a multicast routing protocol capable of handling high mobility, high traffic load and the ability to handle multiple sources and multiple large multicast groups. Depending on how the routes connect the multicast members with each other, we can basically distinguish two major categories of protocols [1,2]: mesh-based and tree-based protocols [3].

The key difference between multicast meshes and multicast trees is that in a multicast mesh data packets are transmitted over more than one path [4,5]. In a mesh-based protocol, if one path is broken other redundant paths deliver the multicast packets; network structure reconstruction is less frequent and produces lower control overhead. A mesh-based protocol thus benefits from an increased robustness at a cost of redundancy in data transmissions and thus lowered efficiency. Existing mesh-based approaches seldom try to reduce the data packet overhead; concentrating solely on robustness. Mesh-based approaches that rely on the senders to maintain the mesh have the drawback of multiple control packet floods per multicast group. Some mesh-based approaches select one or more receivers as multicast group leaders (referred to as core nodes) to maintain the mesh and reduce network wide flooding [6,7].

In this paper, we propose centered protocol for unified multicasting through announcements (CPUMA), a mesh-based protocol that provides robustness and reduces overhead, compared to existing protocols, by (1) periodically centering the core of the mesh at the intersection of data sources; (2) not allowing nodes on the periphery of the mesh to rebroadcast data packets emanating from inside the mesh in order to reduce unnecessary data packet forwarding; and (3) creating forwarding paths toward the nearest mesh member instead of the core of the mesh to reduce latency and take advantage of the robustness of the mesh sooner than later. Without centering the core node, receivers will form a mesh around a core node that may move to the edge of the network creating long single-use paths. The paths, created to a core node that is at the center of the sources, are shorter, more robust around the area data packets must traverse and are able to reach multiple receivers.

Forcing mesh members on the periphery of the mesh to not forward packets received from within the mesh, results in a considerable amount of data packets not forwarded in the network. This considerably reduces data overhead compared to existing protocols (e.g., PUMA [8]). Source nodes outside the mesh forward data packets toward the nearest mesh member instead of the core node. The sooner a data packet reaches the mesh the sooner it can take advantage of multiple paths in the mesh; this allows reducing lost data packets and thus increasing robustness. Forwarding data packets from the source to the nearest mesh member allows also CPUMA to properly elect the core node to be the node at the center of the sources.

The remainder of the paper is organized as follows. Section 2 presents related work. Section 3 describes details of the proposed multicasting protocol. Section 4 shows the effectiveness of the protocol *via* simulations. Section 5 concludes the paper.

2. RELATED WORK

Multicast *ad hoc* on-demand distance vector (MAODV) is a well known tree-based protocol that creates and maintains a bi-directional shared-tree for each multicast group [9,10]. Receivers join a tree by broadcasting a route request join packet, and receiving a route reply join packet when a tree member is found. Nodes along the path, to the node that responded, add routing entries when they receive route replies, thus creating the forwarding path. The freshest route with the highest sequence number and least amount of hops to the tree is grafted to the tree by a multicast activation packet. The activation packet is forwarded along the forwarding path until reaching a node in the tree. The first node to join a group in each multicast tree is the group leader. Periodic group hellos are transmitted by the group leader to announce and maintain the tree. A broken link in the tree is detected by the failure to receive transmissions from a neighboring node closer to the group leader. The node (one of the nodes connected by this link) that is the farthest from the group leader attempts route reconstruction to repair the broken link. In a high mobility scenario, link breaks are to be expected and cause the tree to be in a constant state of reconstruction. In a high traffic load scenario, hello packets are lost due to collisions which yield 'apparent link breaks' and triggers unnecessary route reconstruction [11]. This constant reconstruction results in the flooding of control packets further exacerbating the problem and degrading performance significantly.

Robust multicasting in *ad hoc* networks using trees (ROMANT) is a tree-based protocol that solved the problem of fixing broken links in MAODV by avoiding it altogether and instead reusing the group hellos to periodically reconstruct the group [11]. ROMANT implements a distributed algorithm to elect one of the receivers of a multicast group as the core of the group. Receivers periodically transmit a 'join' announcement, selecting the neighbor node closer to the group leader as the next hop. Receivers know which neighbor node is closest to the group leader from the reception of group hellos. Thus, a 'join' announcement triggers the generation of join announcements by all nodes lying on a shortest path between the receiver and the core. The announcements eventually reach the group leader to form the tree. Senders send data to the group along the shortest path between the sender and the core of the group. Once the data packet reaches a tree member, it is flooded within the tree. The protocol rebuilds an optimal tree every 3 s. If a broken link is detected between rebuilds *via* the lack of an implicit acknowledgement, nodes can use alternate next hops. However, like other tree-based protocols, broken branches result in packets being lost [12,13].

The Protocol for Unified Multicast Announcements (PUMA) can operate as a tree or a mesh-based protocol. It evolved from ROMANT and it uses a single type of control packet, called the Multicast Announcement (MA) [8]. The MA is used to elect cores, join and leave the mesh, update the mesh, and allow nodes outside of the mesh to find routes toward the core. Cores are elected by a distributed algorithm. A node without a route to a multicast group core declares itself as the core and transmits a MA to its neighbors. The neighbors propagate the best received MA, considering a high node ID better than a low node ID. Each receiver connects to the core along all the shortest paths between it and the core forming a mesh with all the nodes along the shortest paths to the core. A 'parent' field in the MA contains the address of the neighbor closest to the core. A non-member forwards multicast data packets if it is the parent of the sending node. Once a core is chosen, it remains the core unless the network is partitioned or the core fails. The static core election does not take into account the distance from the core to the sources or node mobility. This can result in considerable data packet overhead because a core at the 'edge' of a mesh, away from source nodes, will have long forwarding paths to reach the receivers and therefore experience high latency.

Vaishampayan *et al*. [14] propose a protocol, called ARPMM (Adaptive Redundancy Protocol for Mesh based Multicasting), that improves the performance of CPUMA. The protocol enables a thinner mesh to be created when low mobility or high density scenarios provide enough reliability without the need for full redundancy. The protocol removes redundant data forwarding paths when it measures reliability to be high and adds paths when reliability drops. It adapts the network structure from a tree (single forwarding path) to a thin mesh or to a fully redundant mesh depending on the network mobility, node density and reliability requirements. ARPMM could be applied/used to improve the performance of any mesh-based protocol and in particular CPUMA. Thus, ARPMM can be used in combination with CPUMA.

The core hierarchical election for multicasting in *ad hoc* networks protocol (CHEMA) [15] has been shown to outperform PUMA [15] in static topologies. However, CHEMA makes the assumptions that (a) the transmit power can be raised so that core nodes can reach all mesh members in one-hop communication; and (b) mesh data packets

use non-interfering channels to avoid collisions. PUMA and CPUMA do not make these restricting assumptions.

Multicasting on Directional Antennas (MODA) is a protocol that evolved from PUMA with the aim of reducing data packet overhead. It does this by using GPS to set the core at the center of the mesh and covering two hops instead of one when forwarding data packets. Each sender tries to forward data packets to a node two hops closer to the core of the mesh. Once the core node receives the data packet, it makes use of directional antennas [16] to make multiple transmissions in different directions to reach nodes two hops away from it. A node one hop away from a transmitting node receives the data packet but does not rebroadcast it, thus lowering data packet overhead. However, GPS is not always available or appropriate in all situations, and there are other ways to determine the 'center' of a group of nodes.

Various distributed center-location algorithms have been proposed to approximate the minimal-cost tree spanning all members of a multicast group [17]. The knowledge requirements of such algorithms include the sources list, members list, and distance information. Factors to determine which node should be the center include: the maximum distance, the average distance, and the maximum diameter to the members, the sources, or all nodes in the network. However, these algorithms generate considerable control overhead, require knowledge of the network topology, or do not scale since they must keep track of all nodes to elect a centered core [18].

A mesh member in PUMA does not know all of the members of the multicast group, but the mesh members of the multicast group do know all of the sources from the broadcasted data packets. Much like MAODV failed to leverage Group Hellos, PUMA fails to leverage the knowledge gained from the data packets and instead it simply selects the highest receiver ID as the core of the mesh. The core in PUMA is left to wonder the network and create a non-optimized mesh structure.

CPUMA, we propose in this paper, uses the source and hop count information retrieved from data packets to calculate a mesh member's average distance (measured in hop count) to the sources. It elects and maintains the core node in the source-based center of the multicast group mesh, selectively rebroadcasts data packets to reduce data packet overhead and creates forwarding paths toward the nearest mesh member to reduce latency. It uses a single control packet type and does not significantly increase control overhead.

3. CPUMA

3.1. Overview

CPUMA is a mesh-based protocol that implements a distributed algorithm to elect and maintain one mesh member (not necessarily a receiver) as the core of the multicast group. Periodic Multicast Announcements originated at the core, and broadcasted to every node in the network contain all the information needed to enable the protocol to function. Every receiver connects to the elected core along the shortest routes, and these nodes form a mesh. A source node analyses the MAs it receives and sends a data packet to the multicast group along the shortest path to the nearest mesh member (not necessarily the core). When the data packet reaches a mesh member, it is flooded within the mesh. Nodes maintain a list of sources and the shortest hop count from the source. This information is obtained from data packets and CPUMA header; it is used by each mesh member to calculate the average minimum distance (measured in hop count) to the sources. The average minimum distance is simply the sum of the smallest hop counts to each source divided by the number of sources. This minimum distance is referred to as the weight of the member with respect to being the center of the mesh. The mesh member with the lowest weight is elected as the core. A Mesh member will periodically monitor its weight and if it is lower than the weight of the current core, it will elect itself as the new core.

3.2. The multicast announcement

The functions performed by CPUMA allow nodes to join and leave the multicast group, participate in core election, as well as inform all nodes of their distance to the core, their distance to the mesh, and the next hop toward the mesh. Each node can calculate its distance to the core of the multicast group, and its distance to the nearest mesh member in the multicast group. To realize these functions, CPUMA makes use of multicast announcements; these announcements are first broadcasted by the core and then altered and rebroadcasted by each recipient. A MA includes the following fields (Table I):

- Core ID: The address of the elected core.
- Core weight: The weight of the elected core.
- Group ID: The address of the multicast group.
- Sequence number: The sequence number in the latest MA received for that group.
- Parent: The address of the next hop toward the core if the current node is a mesh member, otherwise, the address of the next hop toward the nearest mesh member.
- Distance to core: One plus the distance to the core of the neighbor in the connectivity list of this multicast group with the smallest distance to the core.
- Distance to mesh: Set to zero for all Receivers and Mesh members; for the other nodes, it is set to one plus the distance to the mesh of the neighbor in the connectivity list of this multicast group with the smallest distance to the mesh.

Table I shows the format of each CPUMA multicast announcement. MAs from multiple multicast groups are aggregated together, eliminating the need for multiple MA broadcasts for each multicast group. Table II shows the structure of the CPUMA Header. The CPUMA Header is

Table I. CPUMA multicast announcement format.

Core ID	Core weight	Group ID	Sequence number	Parent	Distance to core	Distance to mesh

included in all transmitted packets. The CPUMA header includes (Table II):

- MA count: Number of MAs contained in the control packet (zero if data packet).
- Hop count: Number of times data packet has been forwarded (zero if control packet).
- Reserved [1]: Empty (for future use).

After the CPUMA Header, the packet may contain one or more MAs if it is a control packet or the data being transmitted if it is a data packet. CPUMA does not combine MAs and data together in one packet.

3.3. Connectivity and source lists

Every node in the network maintains a connectivity list using the MAs it receives from its neighbors. An element in the connectivity list contains the neighbor ID, MA reception time, and all the values of the fields in the MA as they were received. A node will use the connectivity list to build its own MA. The connectivity list is updated with the highest sequence number announcement from each neighbor for each group and the time it was received. The sequence number is generated by the core node and incremented every time it sends a periodic MA. If a node receives a MA for a known group with a better core (lower weight or equal weight and higher ID), it deletes the current connectivity list for that group and creates a new connectivity list starting with the MA it has just received. The connectivity list allows a node to find the neighbor with the smallest distance to the mesh, the smallest distance to the core and its multicast parent. The node chosen as the multicast parent depends on the status of the current node. It is the next hop along the shortest route to the core if the current node is a mesh member; the current node will select the neighbor with the smallest distance to the core in its connectivity list as its parent. If the current node is not a mesh member, its parent is the next hop along the shortest route to the nearest mesh member; the current node will select the neighbor with the smallest distance to the mesh in its connectivity list as its parent.

Every member in the multicast group also maintains a source list. The source list contains the multicast group ID, the source address, and the last packet ID received from each source, all extracted from the data packets of each source. The time the last data packet was received as well as the hop count to the source are added to each entry in the list. The CPUMA header contains the hop count from the source, which is initialized to zero when the source first broadcasts its data packet and is incremented by one every time it is forwarded. Since data packets are flooded within the mesh, nodes maintain a packet ID cache to drop duplicate data packets. Mesh members update the hop count and time received of the source list before dropping duplicate packets. The source list keeps the smallest hop-count from duplicate data packets. Higher packet IDs replace older entries. Entries older than the source timeout (e.g., 3 s) are not used when calculating node weights.

Table II. CPUMA header.

MA count	Hop count	Reserved [1]

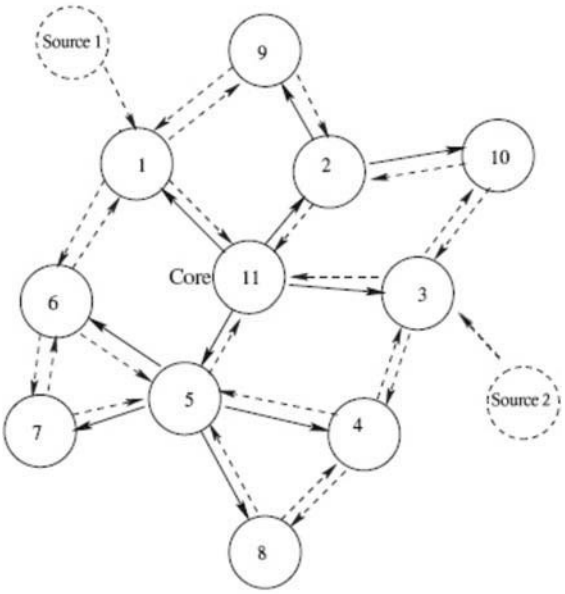

Fig. 1. Mesh broadcasting multicast announcements.

For better understanding, let us consider the example (Figure 1) that shows the broadcasting of MAs initiated by the core. Tables III and IV show the source list and the connectivity list, respectively, maintained by node 6.

Table III. Source list at mesh member 6.

Source	Hop count	Packet ID	Time
Source 1	2	309	12 190
Source 2	4	204	12 250

Table IV. Connectivity list at mesh member 6.

Neighbor	Distance to core	Parent	Distance to mesh	Time
1	1	11	0	12 152
5	1	11	0	12 180
7	2	5	0	12 260

Table V. MA fields: values for mesh member 6.

Field	Value
Core ID	11
Core weight	2
Group ID	224.0.0.1
Sequence number	79
Distance to core	2
Parent	5
Distance to mesh	0

Node 6 will transmit a MA with the values shown in Table V; the core weight is 2 since the core node is an average of 2 hops away from both sources (Figure 1); the parent could have been node 1 or 5, but the multicast announcement for node 1, in this example, is received first and is the one selected.

3.4. Core election and centering

A receiver that wishes to join a multicast group from which it has not received a MA considers itself the core of that group. It starts sending periodic MAs with the following values (Table VI):

Every *multicast announcement interval* (e.g., 3 s), the node will increase the sequence number by 1 and rebroadcast the MA to its neighbors.

Unless receiving a MA for a new group, or an existing group with a new core, nodes wait a short period of time before generating their own announcements. Nodes propagate MAs based on the best MAs they receive from their neighbors. A MA with a lower core weight is considered better than a MA with a higher core weight; in the case of a tie, the higher core ID is considered better than a lower core ID.

The core (re-)computes its weight before sending its MA every *multicast announcement interval*. Every *centering interval* (e.g., 15 s), every member of the multicast group (re-) computes its weight and compares it to the weight of the core. If its weight is smaller than the weight of the core by the *minimum threshold* (e.g., 1 hop or 10%), it elects itself as the new core, and broadcasts it to its neighbors. It is worth noting that our simulations (Section 4) did show that setting the *centering interval* equal to or less than the *multicast announcement interval* resulted in an increase in control packet overhead without significant improvements. The simulations performed best overall using 15 s as the *centering interval*.

Table VI. MA fields: initial values.

Field	Value
Core ID	Self
Core weight	Invalid weight
Group ID	Current multicast group
Sequence number	1 (and incrementing by 1 each time)
Distance to core	0
Parent	Invalid address
Distance to mesh	0

A node, that receives a MA with a core ID and core weight that are better than the values it currently holds in its group connectivity list, updates its values and broadcasts a MA immediately. Eventually, every node will receive a MA with the best core ID and core weight for that multicast group. If a receiver does not hear a MA for a period of time three times the MA interval (e.g., 9 s), it elects itself as the core of that multicast group and begins transmitting MAs.

3.5. Mesh establishment and maintenance

Receivers set their mesh distance to zero in their MAs to indicate they are mesh members. A non-receiver becomes a member if its connectivity list contains a fresh entry with at least one mesh member with a bigger hop count to the core than itself. An entry is considered fresh if it has been received within two times the MA interval (e.g., 6 s). This allows all shortest paths from the receivers to the core to be included in the multicast mesh. Nodes transmit an immediate new MA whenever their mesh distance changes to or from zero. A node outside of the mesh sets its parent to the neighbor in the connectivity list with the shortest distance to the mesh and sets its distance to the mesh as 1 plus the value of its parent's distance to the mesh. In the case where more than one neighbor has the same distance to the mesh, the connectivity list entry, that is received first, is chosen.

3.6. Forwarding multicast data packets

The neighbors in the connectivity list with a smaller distance to the mesh are the potential next hops to the multicast group. A node that is not a member of the mesh forwards a multicast data packet if it is the *next hop* of the node that sent the data packet. Multicast data packets are forwarded hop by hop until they reach the nearest mesh member at which point they are flooded within the mesh. The packet ID cache allows nodes to drops duplicates.

When a node that is not a mesh member transmits a packet, it expects its next hop to forward it. When the next hop forwards the packet, the node that originally sent the packet also hears the forwarded packet. This mechanism serves as an implicit acknowledgement that the packet was received by the next hop. The connectivity list is updated and the next hop is removed if a node does not receive an implicit acknowledgement of the data packet transmission within the *acknowledgement period* (e.g., 1 s).

In PUMA, all mesh members forward packets, and all receivers are mesh members. In CPUMA, a receiver forwards packets only if it has mesh children or receives a packet from outside the mesh (i.e., from a non-member neighbor node that considers the receiver as the parent

node). A receiver that is a parent to a mesh member is within the mesh; it is a hop in the path from another receiver to the core. A receiver that is not a parent to a mesh member is in the periphery. There is no need for receivers on the periphery of the mesh to rebroadcast data packets received from within the mesh, since no node outside of the mesh needs to receive the packet. Table VII presents the pseudo code for of the key functions of CPUMA.

4. SIMULATIONS

In this section, we present the simulation results comparing CPUMA and PUMA. We do not compare CPUMA with MAODV or ODMRP since PUMA has already been shown to perform better than those protocols [8]. PUMA concentrates mesh redundancy in the region of the receiver chosen as the core. CPUMA concentrates mesh redundancy in the

Table VII. CPUMA pseudo-code.

Compute Node Weight	`SUM(shortest path length to each source) / the number of sources`
Join Group	`send multicast announcement`
Leave Group	`/* do nothing - node will timeout */`

Elect Core	`if self is a Receiver AND`
	`(my.coreId == Unknown) then`
	`my.coreId = self`
	`send a Multicast Announcement (MA)`
	`/*my.x is equal to the value of x of the current node; self is`
	`equal to current node */`
	`end if`
	
	`if self receives MA`
	`if (my.coreId == Unknown OR`
	`ma.coreWeight < getGroupWeight OR`
	`( ma.coreWeight == getGroupWeight AND`
	`ma.coreId > my.coreId)) then`
	`/*ma.x is equal to the value of x included in the MA that is`
	`received */`
	`coreId = ma.coreId`
	`send MA`
	`end if`
	`end if`

Compute Distance to the Core	`if my.coreId == self then`
	`my.distance_to_the_core = 0`
	`else`
	`my.distance_to_the_core = INVALID_DISTANCE`

Table VII. (*Continued*)

	For each *node* in connectivity list do
	if *node*.distance_to_the_core <
	my.distance_to_the_core
	/*node.x is equal to the value of x of *node* in connectivity
	list*/
	then
	my.distance_to_the_core =
	node.distance_to_the_core
	end if
	end For
	end if

Compute Distance to the Mesh	**if** (my.coreId **==** self **OR**
	self is a receiver **OR**
	self is a member of the mesh) **then**
	my.distance_to_the_mesh = 0
	else
	my.distance_to_the_mesh = INVALID_DISTANCE
	For each *node* **in connectivity list do**
	if *node*.distance_to_the_mesh <
	my.distance_to_the_mesh
	then
	my.distance_to_the_mesh =
	node.distance_to_the_mesh
	end if
	end For
	end if

Process the reception of Data Packet	**if** (parent **==** self **OR** (self is a member of the mesh **AND** my.number_of_mesh_children **>** 0)) **/* parent is the value of the field parent included in CPUMA header**

Table VII. (*Continued*)

	that comes with Data packet */ **then** hopcount++ /* add 1 to hop count value in MA header that is included in all transmitted packets */ broadcast data packet with the new value of hop count **end if**

Compute the Number of Mesh Children	**My.number_of_mesh_children = 0**
	For each *node* in connectivity list do
	if *node* is a member of the mesh AND
	***node*.distance_to_the_core >**
	my.distance_to_the_core
	then
	my.number_of mesh_children ++;
	/* the current node is responsible, in terms of forwarding/transmission, for all mesh nodes farther from the core than the current node */
	end if
	end

region of the mesh that is located in the 'center' of the source nodes, and therefore the area where data packets must travel through. We compare both of these algorithms using NS-2 [19]. We thank Sidney Doria for the PUMA code for NS-2.

To illustrate the data packet overhead savings of CPUMA we consider a simple example with one source, three receivers and static nodes (Figures 2 and 3); solid nodes indicate Mesh members while dashed nodes indicate non-members. Sources and Receivers are labeled in both figures. Figure 2 shows the mesh structure after core election in PUMA. The highest receiver ID is elected core and the other receivers connect *via* the shortest paths to it; the number next to each node indicates the number of times a data packet is broadcasted before it reaches that node. In PUMA, the source forwards packets toward the core; once the core receives the packets it forwards them to all receivers. It takes a total of nine broadcasts to reach all receivers: four broadcasts to get from the source to the core (*via* nodes a-b-d) and five broadcasts to reach the receivers (one broadcast by the core and one broadcast by f, g, h, and i each).

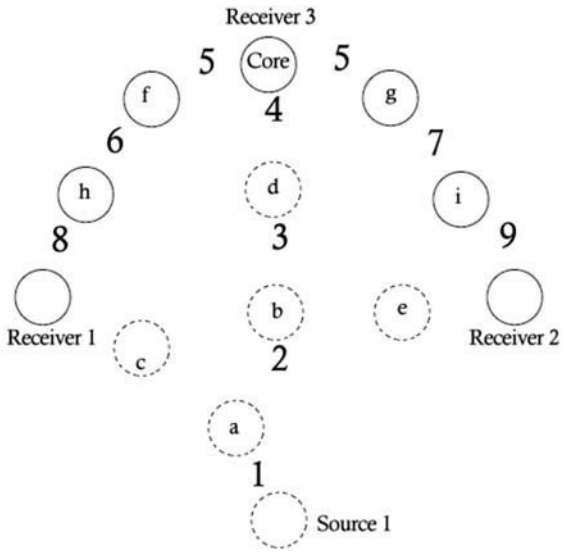

Fig. 2. PUMA: data packet overhead.

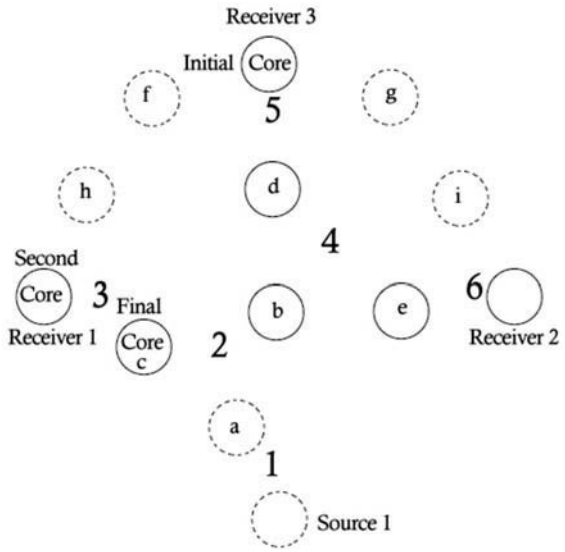

Fig. 3. CPUMA after centering the core.

Starting from the structure shown in Figure 2, we make use of CPUMA. Figure 3 shows the mesh structure after core centering by CPUMA; first, Receiver 1 is selected as the core; then, node c (with a smaller weight) is selected as the core.

Receiver 3 determines a shortest path to node c (the core) *via* nodes d and b and Receiver 2 determines a shortest path to node c *via* nodes e and b. Now, the mesh is optimized for multicasting traffic from the source to the receivers. Indeed, it now only takes six broadcasts to reach all receivers; two from the source to the core, one more from the core to Receiver 1, b broadcasts and reaches both d and e which each broadcasts once more to get to Receiver 2 and Receiver 3.

We simulated the source sending 2 packets/s to the three receivers with zero mobility for 3000 s. Table VIII shows that CPUMA reduces the amount of data packets by almost 50% compared to PUMA (58 686 *vs*. 30 058). The number of control packets is practically the same, for both PUMA and CPUMA, since only two rounds of core re-centering, by CPUMA, are performed. Delivery ratio is slightly improved and latency is improved since the number of hops from source to receivers is reduced.

Table VIII. PUMA *vs.* CPUMA statistics.

	PUMA	CPUMA
Data packets sent	5970	5994
Data packets received	17 587	17 936
Data packets forwarded	58 686	30 058
Delivery ratio	98.20%	99.74%
Control packets sent	13 357	13 053
Latency	0.057	0.035

Table IX. Simulation parameters.

Simulation parameters	
Simulator	NS-2 version 2.33
Simulation time	700 s
Simulation area	1000 m × 1000 m
Node placement	Random
Pause time	0
Mobility model	Random waypoint
MAC protocol	IEEE 802.11-1997
Data packet size	512
All other parameters	NS-2 defaults

4.1. Metrics

The metrics used in our evaluation are packet delivery ratio, control overhead, data packet overhead, latency and traffic. Packet delivery ratio is the number of data packets delivered divided by the number of data packets that should have been delivered. The number of data packets that should have been delivered is the product of the number of transmitted data packets times the number of receivers. Control overhead is the number of control packets that are generated divided by the number of data packets delivered. Data packet overhead is the number of data packets transmitted divided by the number of data packets delivered. Latency is the sum of the delay between sending a packet (from the source) and receiving it (by the receiver) for all data packets divided by the number of data packets received. The data packets overhead is more important than the control overhead since the data packets are several (17 in our simulations) times larger than the control packets (544 compared to 32 bytes). Traffic is the sum of the total Kbytes transmitted. The PUMA and CPUMA headers are equal in size, so no extra overhead is incurred.

4.2. Scenarios

The values of the simulation parameters used in all experiments are shown in Table IX. Five experiments were carried out to compare PUMA with CPUMA.

We used scenarios similar to those found in Reference [8]:

- Experiment 1: 'Mobility' assumes 1, 5, 10, 15, and 20 m/s; Senders = 5; Members = 20; Traffic load = 10 packets/s.
- Experiment 2: 'Senders' assumes 5, 10, 15, and 20; Mobility = 5 m/s; Members = 20; Traffic load = 10 packets/s.
- Experiment 3: 'Members' assumes 5, 10, 20, 30, and 40; Mobility = 5 m/s; Senders = 5; Traffic load = 10 packets/s.
- Experiment 4: Traffic load assumes 10, 20, 30, 40, and 50 packets/s; Mobility = 5; Senders = 5; Members = 20.

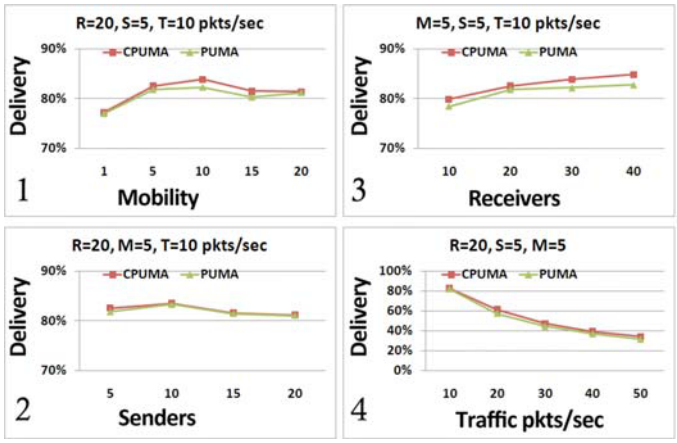

Fig. 4. Packet delivery ratio.

- Experiment 5: Multiple Multicast groups 1, 2, 5, 10; Senders = 5; Members = 20; Mobility = 5 m/s; Traffic load = 10 packets/s.

Senders and Receivers are chosen randomly from among the 50 existing nodes. Traffic load is equally distributed among all senders; a traffic load of 10 packets/s and five senders mean that each sender sends 2 packets/s. R stands for Receivers, S for Senders, M for mobility and T for traffic in the graphs below.

Senders and Receivers are chosen randomly from among the 50 existing nodes. Traffic load is equally distributed among all senders; a traffic load of 10 packets/s and five senders mean that each sender sends 2 packets/s. R stands for Receivers, S for Senders, M for mobility and T for traffic in the graphs below.

4.3. Results

A small improvement in the packet delivery ratio across the board for CPUMA is shown in Figure 4. Indeed, CPUMA delivers 0.3–2% more data packets than PUMA in most scenarios except in Figure 4-4. In Figure 4-4, the network is very congested and the reduced packet forwarding allows CPUMA to outperform PUMA by 2–3.5%. Since data packets travel toward the nearest mesh member instead of the core, data packets benefit from the redundancy of the mesh sooner and are less likely to be lost.

The control overhead of CPUMA and PUMA is shown to be practically equal except in Figure 5-2. In Figure 5-2, the number of senders increases resulting in more values used to calculate node weights (in the case of CPUMA). The weight calculations change faster resulting in the frequent centering of the mesh, and therefore more control packets. The increase is small (2–2.5%) since mesh members only check their weights at 15-s intervals. If the *centering interval* is lowered to 1 s, control packet overhead doubles in our 20 sender scenario without improving the results significantly. This is because the core changes around a few nodes near the current center without affecting the mesh structure. It is prudent to choose a reasonable interval so the core is not constantly changing.

The source centered mesh created and maintained by CPUMA results in shorter paths and mesh nodes on the periphery not forwarding data packets unnecessarily thus

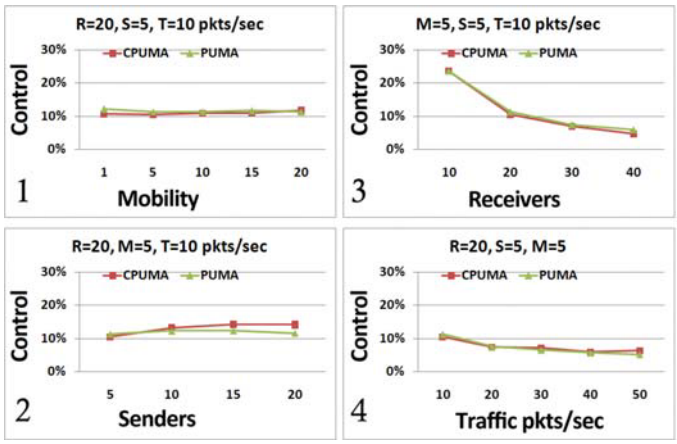

Fig. 5. Control overhead.

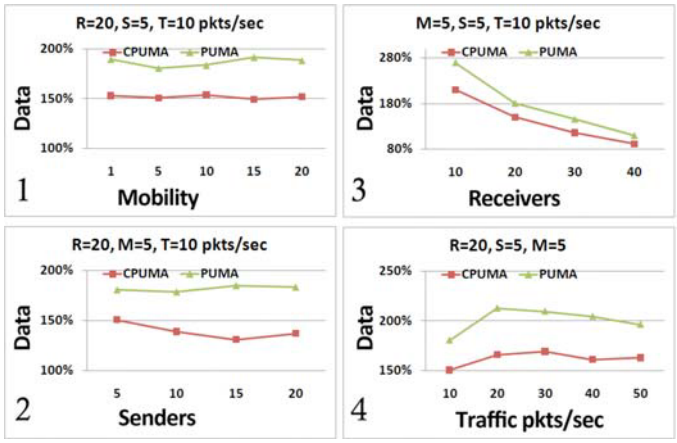

Fig. 6. Data packet overhead.

reducing the amount of data packets being forwarded; this allows CPUMA to handily outperform PUMA in terms of data packet overhead as shown in Figure 6. Indeed, CPUMA achieves an average overhead reduction of 30%; the reduction exceeds 50% in Figure 6-2 with 15 senders. The reduction in data packet overhead is maintained when faced with changes in mobility, the number of senders, the number of receivers and the amount of traffic. A smaller improvement than the others (14–20%) is seen in Figure 6-3 because as the number of receivers approaches 100%, more nodes have to be included in the mesh and PUMA and CPUMA construct similar meshes.

A large difference in the latency of CPUMA compared to PUMA is shown in Figure 7. The latency for CPUMA in Figure 7-1 averages 0.11 s compared to 0.26 s (more than two times bigger) for PUMA. Latency averages 0.09 s for CPUMA compared to 0.29 s for PUMA (more than three times bigger) in Figure 7-2. The latency difference is more pronounced (four times bigger) as the number of receivers increases, as shown in Figure 7-3, and (six times bigger) as traffic increases in Figure 7-4. In CPUMA, this improvement is due to nodes forwarding data packets toward the mesh, and having a mesh near the center of all of the senders in the network. In PUMA, a sender node would instead send its data packet toward the non-centered core which may be at the other side of the network. This increases the length of the path data packets must travel before reaching the mesh and results in a longer delay reaching the receivers.

The difference in the traffic generated by CPUMA compared to PUMA is shown in Figure 8. All simulations show that CPUMA produced an average 18.4% less traffic than PUMA. The best results at an average of 21.8% less traffic, shown in Figure 8-2, correlates to the data packet overhead reduction shown in Figure 6-2. CPUMA only has an average of 14.1% less traffic than PUMA in Figure 8-4, but averages 2.3% higher packet delivery ratio.

A traffic ratio, which compensates for the difference in the packet delivery ratio of the two algorithms, is created by dividing the data received by the total traffic sent and forwarded. In all four experiments, CPUMA outperforms PUMA as shown in Figure 9. CPUMA achieves a ratio over 100% in Figure 9-3. The broadcast nature of MANETs allows one packet sent to be received by multiple receivers. In CPUMA, the receivers on the outside edge of the mesh

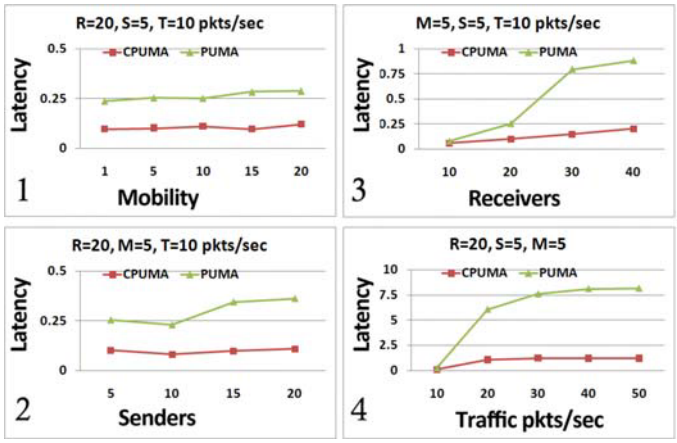

Fig. 7. Latency.

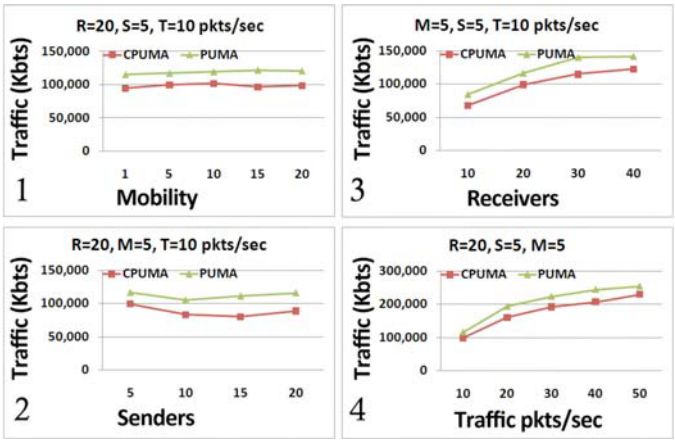

Fig. 8. Traffic (Kbytes).

do not rebroadcast data packets, allowing them to save unnecessary traffic being sent and achieve a traffic ratio over 100%. In the best-case scenario for PUMA (all nodes being receivers), PUMA is limited to just short of 100% due to control packets since all receivers will rebroadcast data packets regardless.

CPUMA outperforms PUMA in the multiple multicast group experiment as shown in Figure 10. CPUMA generates an average of 18% less data packet overhead than PUMA in Figure 10-2. CPUMA generates an average of 17.2% less traffic in Figure 10-4 due to the lower data packet overhead even at the cost of higher control packet overhead. Latency for CPUMA averages 0.12 s compared to 0.19 s for PUMA, as shown in Figure 10-5, a 33% decrease in latency.

Core stability is a feature of PUMA, but not of CPUMA. In Reference [8], the authors report that *frequent* core changes would create additional overhead and significant packet drops because the mesh would always be in a state of reconstruction. To evaluate the impact of core changes on CPUMA, we created a scenario of frequent core changes shown in Figure 11 by extending experiment 1; the re-centering interval of CPUMA assumes 1, 3, 9, 15 and 30 s. Our simulations yielded 15 s to be the best value overall for the re-centering interval. Control overhead is highest when we re-center the mesh every second and the nodes are moving at 20 m/s as seen in Figure 11-2.

When the re-centering interval is equal to or bigger than the multicast announcement interval, control overhead remains stable since MAs transmitted during re-centering reset the MA announcement timer. Since multicast data are forwarded toward the nearest mesh member instead of the core, routes outside of the mesh are not affected by core changes in CPUMA.

To further study the effects of re-centering, we modified the algorithm so that nodes that are neighbors of the current core would not be able to become cores. Simulations were run using the parameters of experiment 1, but limiting the ability to claim core status to nodes 1, 2 and 3 hops away from the current core. The results are shown in Figure 12. For all simulations, packet delivery ratio and latency remained relatively the same, control overhead was lower by only an average of 0.2%, but data overhead increased by an average of 5% since the mesh was not optimized.

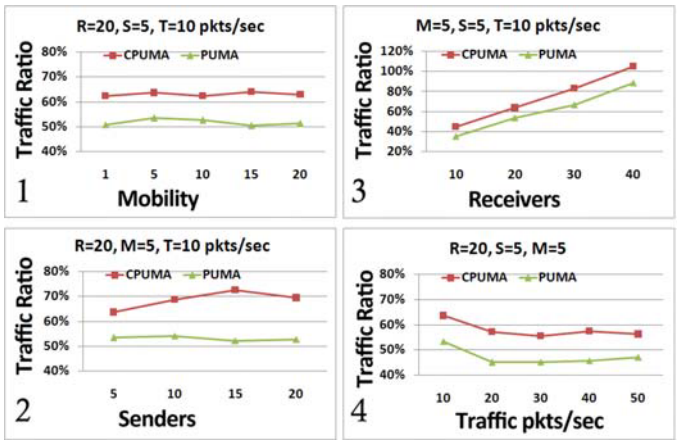

Fig. 9. Traffic ratio.

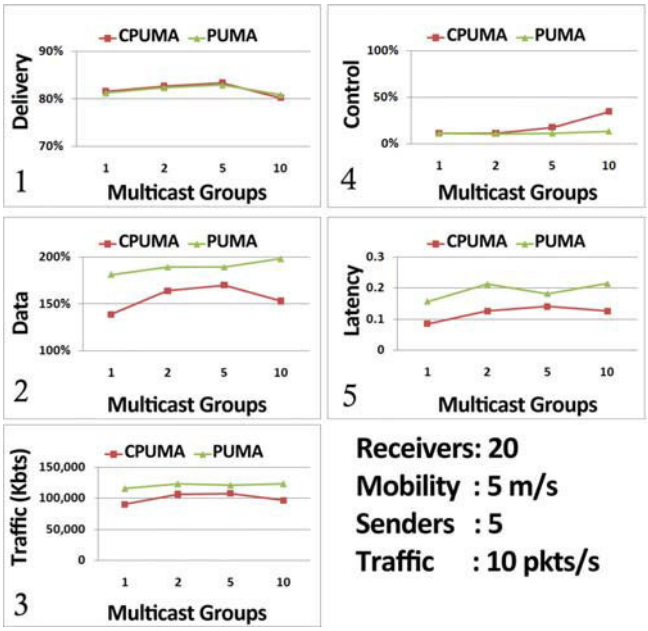

Fig. 10. Multiple multicast groups.

Fig. 11. Re-center intervals.

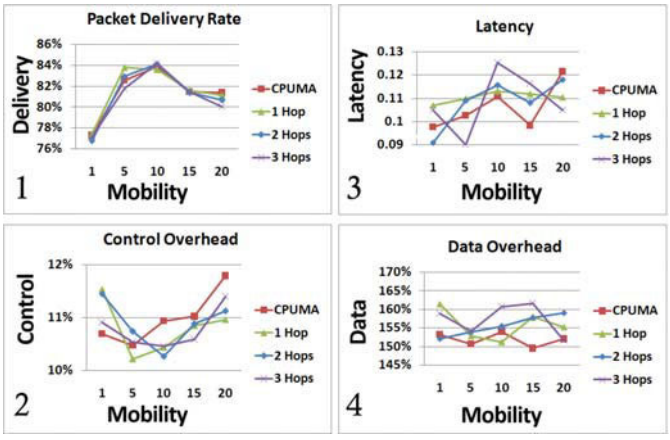

Fig. 12. Re-center away from core.

Table X. Control overhead, data overhead and latency.

Exp no.	1	2	3	4	All
CPUMA control Avg.	10.98	13.05	11.45	7.48	10.57
CPUMA control Std. Dev.	0.5	1.79	8.43	1.78	4.29
PUMA control Avg.	11.55	11.87	12.07	7.25	10.54
PUMA control Std. Dev.	0.36	0.51	7.98	2.44	4.14
CPUMA data Avg.	151.89	139.4	142.3	162.0	149.8
CPUMA data Std. Dev.	1.82	8.21	51.39	7.04	23.96
PUMA data Avg.	186.87	181.6	173.3	200.8	187.2
PUMA data Std. Dev.	4.55	2.83	68.48	12.93	31.0
CPUMA latency Avg.	0.11	0.10	0.13	0.96	0.35
CPUMA latency Std. Dev.	0.01	0.01	0.06	0.48	0.46
PUMA latency Avg.	0.26	0.30	0.50	6.03	1.93
PUMA latency Std. Dev.	0.02	0.07	0.40	3.34	3.08

Table X shows the average control packet overhead, data packet overhead, and latency as well as their standard deviation for both protocols. The experiments listed refer to those described in Section 4.2. Core node changes originate from only a few nodes around the existing core when the weight condition is met and only at specified intervals; therefore, CPUMA does not overwhelm the network with control packets. Standard deviation of control packets for CPUMA is higher in experiment 1 (low to high mobility) and experiment 2 (few to many senders) because high ranges of both may result in more core node changes. In experiment 3 (few to many receivers) and experiment 4 (low to high traffic load) both algorithms have similar control packet characteristics. Eliminating unnecessary data packet transmissions considerably decreases the data packet overhead of CPUMA compared to PUMA in all experiments. Experiment 3 yielded the highest standard deviation for both protocols because the size of the mesh increases with the number of receivers. Even as the size of the mesh increases to encompass all nodes, CPUMA yield savings over PUMA since the nodes on the outer edge of the mesh do not broadcast data packets. Data packet forwarding toward the nearest mesh member instead of the core node lowers latency in all experiments. Networks with lower node densities benefit more from CPUMA. The largest difference is seen in experiment 4 as the traffic load overwhelms the available bandwidth. The mesh constructed by CPUMA not only restricts redundancy to the region containing receivers but also centers it based on the sending nodes and removes data packet forwarding away from the edges of the mesh.

5. CONCLUSION

The CPUMA is based on leveraging data packets to center the core node, forwarding data packets toward the mesh instead of the core to reduce latency. Additionally, in CPUMA receivers selectively forward data packets in an effort to reduce data packet overhead. The mesh constructed in CPUMA is centered with respect to the senders and in the area where data packets must travel to get to the receivers. CPUMA maintains a significantly lower data packet overhead and latency than PUMA while maintaining or improving packet delivery ratio and not significantly increasing control overhead regardless of mobility, traffic, senders or receivers in the network.

REFERENCES

1. Lee SJ, Su W, Hsu J, Gerla M, Bagrodia R. A performance comparison study of ad hoc wireless multicast protocols. *Joint Conference of the IEEE Computer and Communications Societies* 2000; **2**: 565–574.
2. de Morais CordeiroC, Gossain H, Agrawal DP, Multicast over wireless mobile ad hoc networks: present and future directions. *IEEE Network* 2003; **17**: 52–59.
3. Wang Z, Liang Y, Wang L. Multicast in mobile ad hoc networks. *Computer and Computing Technologies in Agriculture* 2008; **1**: 151–1164.
4. Xie J, Talpade R, Mcauley A, Liu M. AMRoute: ad hoc multicast routing protocol. *Mobile Networks and Applications* 2002; **7**: 429–439.
5. Sinha P, Sivakumar R, Bharghavan V. MCEDAR: multicast core-extraction distributed ad-hoc routing. *IEEE Wireless Communications and Networking Conference*, 1999; 1313–1317.
6. Junhai L, Liu X, Danxia Y. Research on multicast routing protocols for mobile ad-hoc networks. *Computer Networks* 2008; **52**: 988–997.
7. Kaliaperumal B, Jeyakumar A. Adaptive core-based scalable multicasting networks. *IEEE India Conference*, 2005; 198–202.
8. Vaishampayan R, Garcia-Luna-Aceves JJ. Protocol for unified multicasting through announcements (PUMA). *IEEE International Conference on Mobile Ad-hoc and Sensor Systems*, 2004.

9. Royer EM, Perkins CE. Multicast ad-hoc on-demand distance vector (MAODV) routing. *Internet-Draft*, draft-ietf-manet-maodv-00.txt, July 2000.
10. Perkins C, Roger E. Ad hoc on demand distance vector (AODV) routing. *IEEE Workshop on Mobile Computing Systems and Applications*, 2002.
11. Vaishampayan R, Garcia-Luna-Aceves JJ. Robust tree-based multicasting in ad hoc networks. *IEEE International Conference on Performance, Computing and Communications*, 2004; 647–652.
12. Ballardie T, Francis P, Crowcoft J. Core based trees (CBT). *ACM Computer Communication Review* 2003; **23**, 85–95.
13. Park V, Corson M. A highly adaptive distributed routing algorithm for mobile wireless networks. *IEEE Conference on Computer Communication*, 1997; 1405.
14. Vaishampayan R, Garcia-Luna-Aceves JJ, Obraczka K. An adaptive redundancy protocol for mesh based multicasting. *Computer Communications* 2007; **30**, 1015–1028.
15. Spohn MA, Garcia-Luna-Aceves JJ. Bounded-distance multi-clusterhead formation in wireless ad hoc networks. *Ad Hoc Networks* 2007; **5**, 504–530.
16. Vaishampayan R, Garcia-Luna-Aceves JJ, Obraczka K. Multicasting On Directional Antennas (MODA). *IEEE International Conference on Mobile Ad-hoc and Sensor Systems*, 2005; 659–664.
17. Karaman A, Hassanein HS. Core-based approach in multicast routing protocols. *International Symposium on Performance Evaluation of Computer and Telecommunication Systems, SPECTS*, 2003; 525–532.
18. Thaler D, Ravishankar C. Distributed center-location algorithms: proposals and comparisons. *IEEE Networking the Next Generation* 1996; **1**, 75–84.
19. Breslau L, Estrin D, Fall K, Floyd S, Heidemann J, Helmy A, Huang P, McCanne S, Varadhan K, Xu Y, Yu H. Advances in network simulation. *IEEE Computer* 2000; **33**: 59–67.

Authors' Biographies

Eric Astier received the B.S from Florida Institute of Technology and M.S. from University of Montreal. His research interests include routing in ad hoc networks.

Abdelhakim Hafid is Professor at département d'Informatique et de recherche opérationnelle de l'Université de Montréal, where he founded the Network Research Lab (NRL) in 2005. Prior to joining U. of Montreal, he was with Telcordia Technologies (formerly Bell Communication Research), NJ, US, faculty at University of Western Ontario, research director at Advance Communication Engineering Center, Canada, researcher at Computer Research Institute of Montreal, Canada, and visiting scientist at GMD-Fokus, Germany. Dr. A. Hafid has extensive academic and industrial research experience in the area of the management of next generation networks including wireless and optical networks, QoS management, distributed multimedia systems, and communication protocols.

Sultan Hamadi Aljahdali, Ph.D. received the B.S from Winona State University, Winona, Minnesota in 1992, and M.S. with honor from Minnesota State University, Mankato, Minnesota, 1996, and Ph.D. Information Technology from George Mason University, Fairfax, Virginia, U.S.A, 2003. He is the dean of the college of computers and information systems at Taif University. His research interest includes software testing, developing software reliability models, soft computing for software engineering, computer security, reverse engineering, and medical imaging. He is also a member of ACM, IEEE, and ISCA.