Centered and Robust Multicast Routing in Mobile Ad Hoc Networks

*Eric Astier, *Abdelhakim Hafid, #Sultan H. Aljahdali
*Network Research Lab, University of Montreal,
Montreal, Canada
{eastier, ahafid}@iro.umontreal.ca
#Computer Science Department, Al-Taif University
Al-Taif, Saudi Arabia
aljahdali@tu.edu.sa

Abstract: **In order for a mesh-based routing protocol, in a mobile ad hoc network, to perform well it must achieve a high level of robustness without excessive overhead. We present the centered protocol for unified multicasting through announcements (CPUMA) for mobile ad hoc networks. 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.**

Keywords-*MANETs; Multicast Routing Protocols; Mobility; Core*

I. I. 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.

The key difference between multicast meshes and multicast trees is that in a multicast mesh data packets are transmitted over more than one path. 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 transmission 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.

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 [3]. The key problem of MAODV is the "continuous" tree reconstruction (because of "real" and "apparent" link failures [4] that are frequent in high mobility and high traffic load scenarios resulting 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 solves the problem of fixing broken links in MAODV by avoiding it altogether and instead reusing the group hellos to periodically reconstruct the group [4]. However, with ROMANT (like other tree-based protocols) broken branches result in packets being lost. The Protocol for Unified Multicast Announcements [5] (PUMA) can operate as a tree or a mesh-based protocol; it evolved from ROMANT. With PUMA, each receiver connects to the core (i.e., the node with the highest receiver id) along all the shortest paths between it and the core forming a mesh with all the nodes along the shortest paths to the core. Once a core is chosen, it remains the core unless the network is partitioned or the core fails. 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. Indeed, the core in PUMA is left to wonder the network and create a non-optimized mesh structure.

Multicasting on Directional Antennas [6] (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. However, GPS is not always available or appropriate in all situations (e.g., underground areas without access to GPS signal). Various distributed center-location algorithms have been proposed to approximate

Digital Object Identifier: 10.4108/ICST.BROADNETS2009.7258
http://dx.doi.org/10.4108/ICST.BROADNETS2009.7258

the minimal-cost tree spanning all members of a multicast group [7]. 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.

In this paper, we propose CPUMA, a mesh-based protocol that provides robustness and reduces overhead (compared to existing protocols), without requiring specific equipment (e.g., GPS) nor hard-to-get information in a mobile environment (e.g., network topology), by (1) periodically centering the core of the mesh; (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; forwarding paths that head directly toward the core instead of the mesh may include nodes not in the mesh where packets have a higher probability of being lost. 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. With CPUMA, 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; mesh members on the outside edge of the mesh do not rebroadcast data packets heard from nodes closer to the core.

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

II. CPUMA

A. *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 (MAs) 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.

B. *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:

- 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 (see Section C) of this multicast group with the smallest distance to the mesh

MAs from multiple multicast groups are aggregated together, eliminating the need for multiple MA broadcasts for each multicast group. The CPUMA Header is included in all packets transmitted. The CPUMA header includes:

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

After the CPUMA Header, the packet may contain 1 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.

C. *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 (distance to core, parent, and distance to mesh) found in the MA received from the neighbor. 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 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 seconds) are not used when calculating node weights.

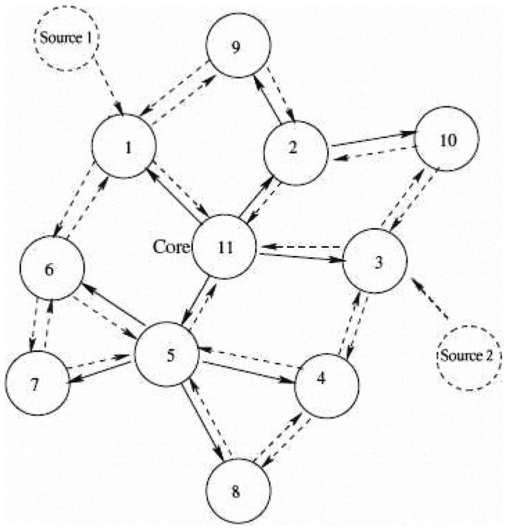

Figure 1. Mesh broadcasting Multicast Announcements

For better understanding, let us consider the example (Figure 1) that shows the broadcasting of MAs initiated by the core. Table 1 shows the MA values for node 6. The core id is 11, the group id is 224.0.0.1, the sequence number is 79 and the core weight is 2. Table 2 shows the source list maintained by node 6. The group id is 224.0.0.1 and the member weight is 3. Node 6 will transmit a MA with the values shown in Table 1.

TABLE I. 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

TABLE II. SOURCE LIST AT MESH MEMBER 6

Source	Hop Count	Packet ID	Time
Source 1	2	309	12190
Source 2	4	204	12250

TABLE III. CONNECTIVITY LIST AT MESH MEMBER 6

Neighbor	Distance to Core	Parent	Distance to Mesh	Time
1	1	11	0	12152
5	1	11	0	12180
7	2	5	0	12260

The core weight is 2 because 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 is received first and is the one selected (Table 3).

D. *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 4):

TABLE IV. 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

Every *multicast announcement interval* (e.g., 3 seconds), 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 seconds), a 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 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 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 seconds as the center interval.

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 will update its values and broadcast 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 receive a MA for a period of time 3 times the MA interval (e.g., 9 seconds) it elects itself as the core of that multicast group and begins transmitting MAs.

E. 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 was received within 2 times MA intervals (e.g., 6 seconds). 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 1 neighbor has the same distance to the mesh, the connectivity list entry, that is received first, is chosen.

F. 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 parent 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 parent to forward it. When the parent forwards the packet, the node that originally sent the packet will also hear the forwarded packet. This mechanism serves as an implicit acknowledgement that the packet was received. The connectivity list is updated and neighbors are removed if a node does not receive an implicit acknowledgement of the data packet transmission within the *acknowledgement period* (e.g., 1 second).

In PUMA all mesh members forward packets, and all receivers are mesh members. In CPUMA receivers forward packets only if they have mesh children or receive a packet from outside the mesh (from a non member neighbor node that considers it 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 is interested in receiving the packet (the packet is not destined for this node). Table 5 presents the pseudo code of the key functions of CPUMA.

TABLE V. CPUMA PSEUDO-CODE

```
Compute   | SUM(smallest hop count to each source)
Node      | / the number of sources
Weight    |
----------+--------------------------------------------
Join      | send multicast announcement
Group     |
----------+--------------------------------------------
Leave     | /* do nothing - node will timeout */
Group     |
----------+--------------------------------------------
Elect     | if self is a Receiver AND
Core      |   (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   | if my.coreId == self then
Distance  |   my.distance_to_the_core = 0
to the    | else
Core      |   my.distance_to_the_core =
          |   INVALID_DISTANCE
          |   For each node in connectivity list
          | do
          |      if node.distance_to_the_core <
          |         my.distance_to_the_core then
          |           /*node.x is equal to the
          |           value of x of node in
          |           connectivity list*/
          |         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)) then
     /* parent is the value of the
    field parent included in CPUMA
    header that comes with Data packet
    */
  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
```

III. 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 [5]. PUMA concentrates mesh redundancy in the region of the receiver chosen as the core. CPUMA concentrates mesh redundancy in the region of the mesh between the source nodes, and therefore the area where data packets travel most through. We compare both of these algorithms using NS-2[8]. We thank Sidney Doria for the PUMA code for NS-2.

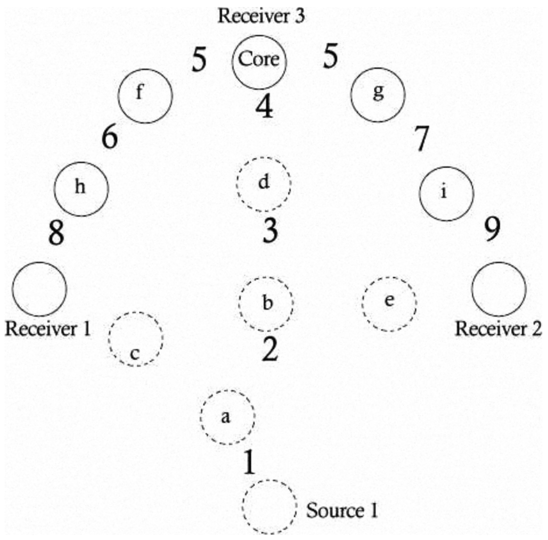

Figure 2. PUMA: Data Packet Overhead

To illustrate the data packet overhead savings of CPUMA we consider a simple example with 1 source, 3 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 both receivers. It takes a total of 9 broadcasts to reach all receivers: 4 broadcasts to get from the source to the Core (via nodes a-b-d) and 5 broadcasts to reach the receivers (1 broadcast by the core and 1 broadcast by f, g, h and i each).

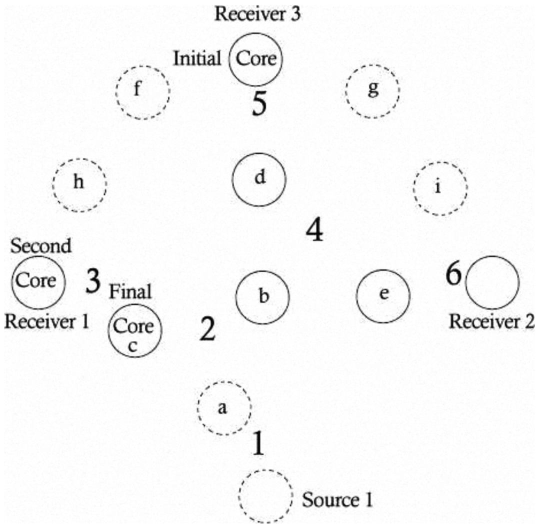

Figure 3. CPUMA after Centering the Core

Digital Object Identifier: 10.4108/ICST.BROADNETS2009.7258
http://dx.doi.org/10.4108/ICST.BROADNETS2009.7258

The mesh structure after the second re-centering of the core in CPUMA is shown in Figure 3. The Source forwards data packets toward the closest mesh member instead of the core, in this case Receiver 1 via nodes a and c. At the first re-centering of the core Receiver 1 becomes the core and the mesh is recreated. Receiver 2 finds the shortest path to Receiver 1 via nodes e-b-c. Receiver 3 finds the shortest path to Receiver 1 via f-h. Node c is selected as the core in the second re-centering of the core. Receiver 3 finds a new shortest path to node c via nodes d and b and the mesh is optimized. It now only takes 6 broadcasts to reach all receivers; 2 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 one source sending 2 packets per second to the three receivers with zero mobility configured as shown above for 3000 seconds. Table 6 shows that CPUMA reduces the amount of data packets by almost 50% compared to PUMA (58686 vs. 30058). The number of control packets is practically the same, for both PUMA and CPUMA, since only two rounds of core re-centering are performed. Delivery ratio is slightly improved and latency is improved since the number of hops from source to receivers is reduced.

TABLE VI. PUMA VS. CPUMA STATISTICS

	PUMA	CPUMA
Data Packets Sent	5970	5994
Data Packets Received	17587	17936
Data Packets Forwarded	58686	30058
Delivery Ratio	98.20%	99.74%
Control Packets Sent	13357	13053
Latency	0.057	0.035

A. 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 data packets that should have been delivered. The number of data packets that should have been delivered is the product of data packets sent 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.

B. Scenarios

The values of the simulation parameters used in all experiments are shown in Table 7. Five experiments were carried out to compare PUMA with CPUMA.

TABLE VII. SIMULATION PARAMETERS

Simulation Parameters	
Simulator	NS-2 version 2.33
Simulation Time	700 seconds
Simulation Area	1000m x 1000m
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

We used scenarios similar to those found in [5]:

- 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
- 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 5 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.

C. Results

A small improvement in the packet delivery ratio across the board for CPUMA is shown in Figure 4. Indeed, CPUMA delivers 0.3 to 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.

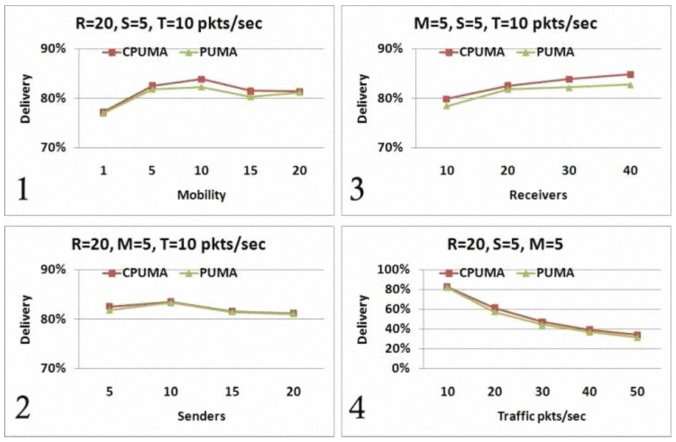

Figure 4. Packet Delivery Ratio

Figure 5 shows that the control overhead of CPUMA and PUMA is 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 second intervals. If the *centering interval* is lowered to 1 second, control packet overhead doubles in our 20 senders' 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.

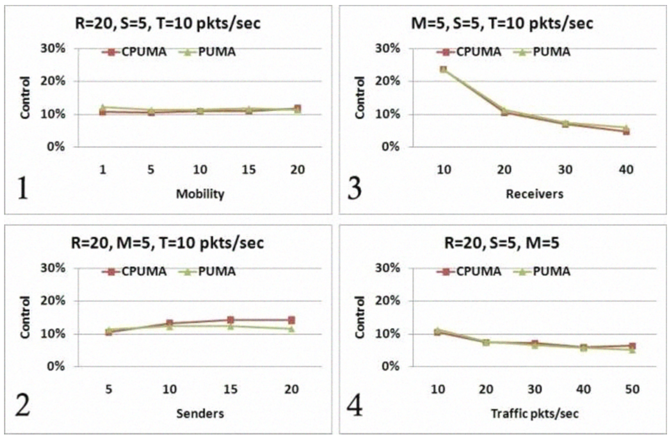

Figure 5. Control Overhead

CPUMA handily outperforms 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.

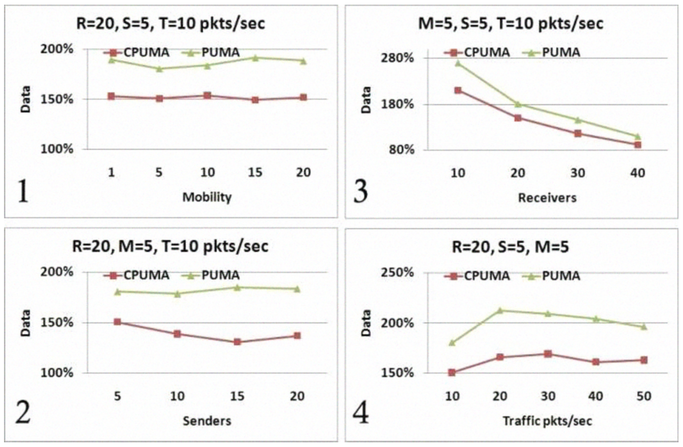

Figure 6. Data Packet Overhead

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 0.11s compared to 0.26s (more than two times bigger) for PUMA. Latency averages 0.09s for CPUMA compared to 0.29s for PUMA (more than 3 times bigger) in Figure 7-2. The latency difference is more pronounced (4 times bigger) as the number of receivers increases, as shown in Figure 7-3, and (6 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 the data packet must travel before reaching the mesh and results in a longer delay reaching the receivers.

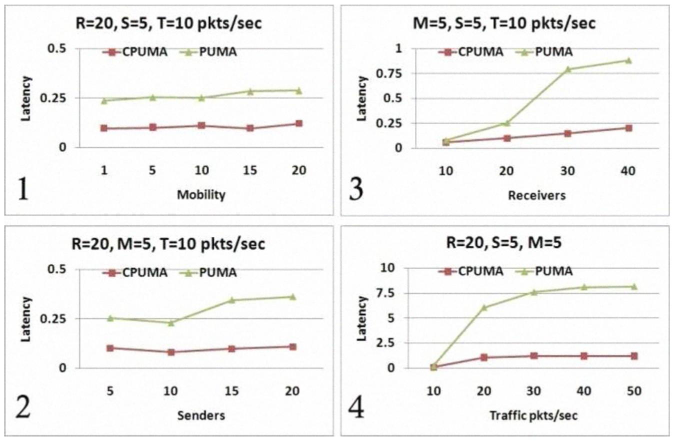

Figure 7. Latency

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 ratios.

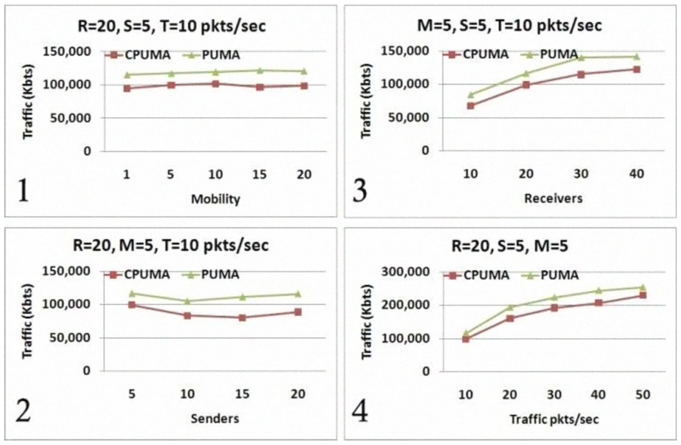

Figure 8. Traffic (Kbytes)

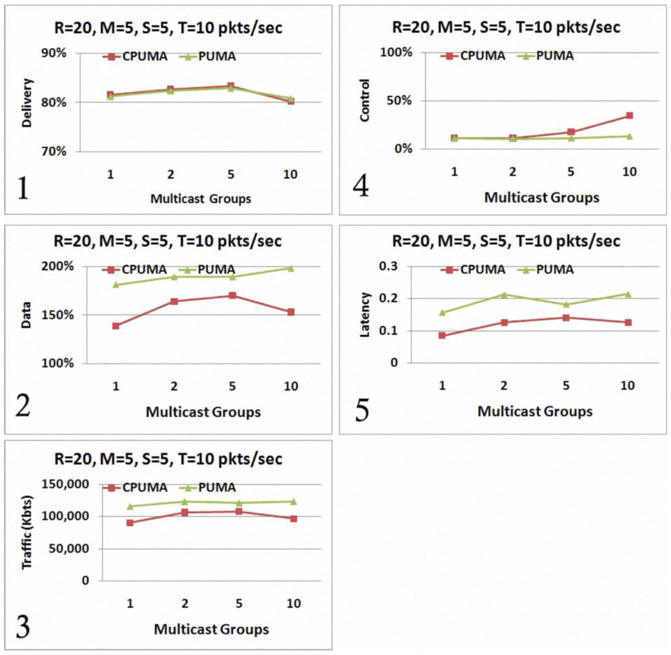

Figure 9. Multiple Multicast Groups

CPUMA outperforming PUMA in the multiple multicast group experiment as shown in Figure 9. CPUMA generates an average of 18% less data packet overhead than PUMA in Figure 9-2. CPUMA generates an average of 17.2% less traffic in Figure 9-4 due to the lower data packet overhead even at the cost of higher control packet overhead. Latency for CPUMA averages 0.12 seconds compared to 0.19 seconds for PUMA as shown in Figure 9-5; this is a 33% decrease in latency.

IV. CONCLUSION

The Centered Protocol for Unified Multicasting through Announcements (CPUMA) is based on leveraging data packets to center the core node, forwarding data packets toward the mesh instead of the core to lower the 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 considerably 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] S.J. Lee, W. Su, J. Hsu, M. Gerla, and R. Bagrodia, "A Performance Comparison Study of Ad Hoc Wireless Multicast Protocols", *Joint Conference of the IEEE Computer and Communications Societies*, 2000, vol. 2, pp. 565-574.

[2] C. de Morais Cordeiro, H. Gossain, DP. Agrawal, "Multicast over Wireless Mobile Ad Hoc Networks: Present and Future Directions", *IEEE Network*, 2003, vol. 17, pp. 52-59.

[3] E.M. Royer and C.E. Perkins, "Multicast Ad-Hoc On-Demand Distance Vector (MAODV) Routing," Internet-Draft, draft-ietf-manet-maodv-00.txt, July 2000.

[4] R. Vaishampayan and J.J. Garcia-Luna-Aceves, "Robust Tree-Based Multicasting in Ad Hoc Networks", *IEEE International Conference on Performance, Computing and Communications*, 2004, pp. 647-652.

[5] R. Vaishampayan and J.J. Garcia-Luna-Aceves, "Protocol for unified multicasting through announcements (PUMA)", *IEEE International Conference on Mobile Ad-hoc and Sensor Systems*, 2004.

[6] R. Vaishampayan, J.J. Garcia-Luna-Aceves and K. Obraczka, "Multicasting On Directional Antennas (MODA)", *IEEE International Conference on Mobile Ad-hoc and Sensor Systems*, 2005, pp. 659-664.

[7] A. Karaman and H.S. Hassanein, "Core-Based Approach in Multicast Routing Protocols," *International Symposium on Performance Evaluation of Computer and Telecommunication Systems*, SPECTS, 2003, pp. 525-532.

[8] Breslau, Estrin, Fall, Floyd, Heidemann, Helmy, Huang, McCanne, Varadhan, Xu, and Yu. "Advances in Network Simulation", *IEEE Computer*, vol. 33, 2000, pp. 59-67.