ADMP: An Adaptive Multicast Routing Protocol for Mobile Ad Hoc Networks*

Rolando Menchaca-Mendez[1], Ricardo Menchaca-Mendez[1], and J. J. Garcia-Luna-Aceves[12]
1 Dept. of Computer Engineering, University of California, Santa Cruz
1156 High Street, Santa Cruz, CA 95064, U.S.A.
{menchaca, rmenchaca, jj}@soe.ucsc.edu
WWW home page: http://www.soe.ucsc.edu/ research/ccrg/home.html
2 Palo Alto Research Center (PARC), 3333 Coyote Hill Road
Palo Alto, CA 94304, U.S.A.
WWW home page: http://www.parc.xerox.com

Abstract. We present ADMP, the adaptive mesh-based multicast routing protocol, in which nodes are able to independently tune the amount of redundancy used to transmit data packets with the goal of improving the overall packet delivery ratio while keeping the retransmission overhead as low as possible. ADMP is based on a novel distributed algorithm for computing connected dominating sets. ADMP uses a single type of control packet, called multicast announcement, which is used to build the meshes of multicast groups, elect the core of each mesh and obtain two-hop neighborhood information. Using detailed simulations for different scenarios, we show that ADMP achieves similar or better reliability than two mesh-based multicast protocols that are very resilient (ODMRP and PUMA) while inducing low packet retransmission overhead.

1 Introduction

Mobile Ad Hoc Networks (or MANETs) are highly dynamic and do not rely on a fixed infrastructure. MANETs are well suited to applications where rapid deployment and dynamic reconfiguration are necessary. Examples of such scenarios are: military battlefield, emergency search and rescue, conference and conventions. The objective of a multicast routing protocol for MANETs is to enable

* This work was supported in part by the Mexican National Council for Science and Technology (CONACyT), by the Mexican National Polytechnic Institute (IPN), by the National Science Foundation under Grant CNS-0435522, and by the Baskin Chair of Computer Engineering at the University of California, Santa Cruz.

communication between one or more senders and a group of receivers in a network where nodes are mobile and may not be within direct wireless transmission range of each other. These protocols must use the available bandwidth and nodes' energy very efficiently, given that they are scarce resources in MANETs and do so when nodes may be highly mobile.

Several MANET multicast protocols have been proposed recently (e.g. [1-8]). In general, the approaches taken up to date can be classified by the way they support the routing structure they maintain; namely tree-based and mesh-based protocols.

A tree-based multicast routing protocol constructs and maintains either a shared multicast routing tree or multiple multicast trees. Recent examples of tree-based multicast routing protocols are the Multicast Ad hoc On-demand Distance Vector Protocol (MAODV) [1] and the Adaptive Demand-driven Multicast Routing Protocol (ADMR) [2]. The tree-based approach has adequate performance in wired networks [9]; however, establishing and maintaining a tree or a set of trees in MANETs incurs substantial communication overhead as the branches break due to node mobility, which has a negative impact in the overall performance of the protocol [3].

On the other hand, a mesh-based multicast routing protocol maintains a mesh for each multicast group consisting of a connected sub-graph of the network that includes all receivers of a particular group and the relays needed to maintain connectivity with all the receivers in the group. Maintaining a connected component is far less complicated than maintaining a tree and hence mesh-based protocols tend to be simpler and more robust. However, as we will see in Section 4, in situations with high mobility or high channel-contention, mesh-based multicast protocols can also have poor performance when too many redundant relays are used to forward multicast traffic. Two well-know representatives of mesh-based protocols are the Core Assisted Mesh Protocol (CAMP) [4] and the On-Demand Multicast Routing Protocol (ODMRP) [5].

In this paper we present the Adaptive Dominant Multicast Protocol (ADMP), a protocol that further improves the reliability and efficiency of its direct predecessors PUMA [6] and DPUMA [3]. ADMP makes use of a novel distributed algorithm that computes connected dominating sets to provide high delivery ratios under high node-mobility and high channel-contention. The main idea that ADMP borrows from PUMA is that a single control packet (a multicast announcement) is flooded periodically to build the mesh for one or multiple multicast groups, elect the core of each mesh and collect two-hop neighborhood information. When forwarding a packet, ADMP dynamically computes the connected dominating set of the current mesh using a utility function that takes into account relative mobility of nodes and channel contention. Depending on the local node conditions, a node adjusts the amount of redundancy used to cover these two-hop neighbors that are also mesh members of a given multicast group.

The remaining of this paper is organized as follows. Section 2 summarizes related work on multicast routing protocols for MANETs and the distributed computation of connected dominating sets. Section 3 presents ADMP and the General Augmented Greedy Set Cover (GAGSC) algorithm. As we will explain in more detail in Sub-section 3.2, GAGSC is able to compute connected dominating

sets taking into account two-hop information regarding channel contention and nodes' mobility in order to compute a dominating set whose size reflects the amount of redundancy used to forward a multicast data packet. In section 4 we show a series of performance comparisons among ODMRP, PUMA, DPUMA, and ADMP over different scenarios. Finally, in Section 5 we present concluding remarks and current work.

2 Related Work

2.1 Multicast Routing Protocols

ODMRP is a representative of the state of the art in mesh-based multicast routing protocols. In order to establish the mesh, ODMRP requires cooperation of nodes wishing to send data to a multicast group. Senders periodically flood a Join Query packet throughout the network. These periodic transmissions are used to update the routes. Each multicast group member after receiving a Join Query, broadcasts a Join Table to all its neighbors in order to establish a forwarding group. Senders broadcast data packets to all its neighbors. Members of the forwarding group forward the packet. Using ODMRP, multiple routes from a sender to a multicast receiver may exits due to the mesh structure created by the forwarding group members. The limitations of ODMRP are the need for network-wide packet floods and the sender initiated construction of the mesh. This method of mesh construction results in a mesh that includes many more nodes that there are needed in a multicast routing tree, as well as numerous unnecessary transmissions of data packets compared to a receiver initiated approach. DCMP [7] is an extension to ODMRP that designates certain senders as cores and reduces the number of senders performing flooding. NSMP [8] is another extension to ODMRP aiming to restrict the flood of control packets to a subset of the entire network. However, DCMP and NSMP fail to eliminate entirely ODMRP's drawback of multiple control packet floods per group.

CAMP avoids the need for network-wide floods from each source to maintain multicast meshes by using one or more cores per multicast group. A receiver-initiated approach is used for receivers to join a multicast group by sending unicast join requests towards a core of the desired group. The drawbacks of CAMP are that it needs the pre-assignment of cores to groups and a unicast routing protocol to maintain routing information about the cores. This later characteristic may induce considerable overhead in a large ad hoc network.

PUMA supports the IP multicast service model of allowing any source to send multicast packets addressed to a given multicast group, without having to know the constituency of the group. Furthermore, sources need not join a multicast group in order to send data packets to the group. Like CAMP, PUMA uses a receiver initiated approach in which receivers join a multicast group using the address of a special node (core in CAMP), without the need for network-wide flooding of control or data packets from all the sources of a group. PUMA implements a distributed algorithm to elect one of the receivers of a group as the core of the group, and to inform each router in the network of at least one next-hop to the elected core of each group (mesh establishment). The election algorithm used in PUMA is essentially the same as the

spanning tree algorithm introduced by Perlman for internetworks of transparent bridges [10]. Within a finite time proportional to the time needed to reach the router farthest away from the eventual core of a group, each router has one or multiple paths to the elected core.

Hence a receiver can connect to the elected core along all shortest paths between the receiver and the core. All nodes on shortest paths between any receiver and the core collectively form the mesh of the multicast group. This is the case given that all nodes in the network receive multicast announcements for every active multicast group stating the core of the group. Hence a sender node can send packets to the multicast group by encapsulating them in unicast packets to the core along any of the paths to the core. PUMA uses a single control packet for all its functions, the multicast announcement. Each multicast announcement specifies a sequence number, the address of the group, the address of the core, the distance to the core, a mesh member flag that is set when the sending node belongs to the mesh, a parent field that states the preferred neighbor to reach the core, and a list of neighbors who are mesh members. With the information contained in such announcements, nodes elect cores, determine routes for sources outside a multicast group to unicast multicast data packets towards the group, notify others about joining or leaving a group's mesh, maintain the mesh and get two-hop information of nodes belonging to each multicast group.

In the basic PUMA protocol, once a multicast message reaches a mesh member, it is flooded across the whole mesh. This can lead to unnecessary overhead because a given node can be covered by more than one neighbor and hence receive a multicast message more than once. In order to reduce this overhead, DPUMA incorporates the concept of connected dominating sets to dynamically determine a subset of one-hop nodes such that if these nodes broadcast the packet, it will be received by all mesh members in a two-hop neighborhood and eventually by all members in the mesh.

2.2 Distributed Computation of Connected Dominating Sets

For the distributed computation of connected dominating sets, we use a simple graph $G = (V,E)$ to represent an ad hoc wireless network, where V represents a set of wireless mobile nodes and E a communication link between two nodes. An edge (u,v) indicates that both nodes u and v are within each other's transmission range. Such graph is also called *unit disk graph* [11]. It is easy to see that the topology of this type of graphs vary over time due to node mobility.

For a given undirected graph $G = (V, E)$, a connected dominating set (CDS) in the graph is any set of connected vertices $V' \subseteq V$ such that each $v \in V - V'$ is adjacent to some vertex in V'. The problem of determining the minimum connected dominating set (MCDS) is known to be NP-complete. Therefore, only distributed approximated algorithms running in polynomial-time are practical for MANETs.

If we compute a connected dominating set V' of a given network, only those nodes belonging to V' have to broadcast a packet in order to reach every node in the network, with the corresponding savings of $V - V'$ messages. It is important to note that distributed approximations that run in polynomial-time do not compute the minimum dominating set; however, in the context of MANETs, computing a larger

dominating set is actually desirable to augment the reliability with which a packet is delivered.

Lim and Kim [12] showed that the *minimum connected dominating set* (MCDS) problem can be reduced to the problem of building a *minimum cost flooding tree* (MCFT) and they proposed a set of heuristics for flooding trees that lead to two algorithms: *self-pruning* and *dominant pruning* (DP). They also showed that both algorithms perform better than *blind flooding*, in which each node broadcast a packet to its neighbors whenever it receives the packet along the shortest path from the source node, and that DP outperforms self-pruning. Since then, many other approaches have been purposed to compute CDS and to improve communication protocols applying CDS. For example, enhancements to dominant pruning have been reported by Lou and Wu [11] who describe the *total dominant pruning* (TDP) algorithm and the *partial dominant pruning* (PDP) algorithm, and by Spohn and Garcia-Luna-Aceves [13] who presented the *enhanced dominant pruning* (EDP) algorithm which improves DP's performance. All these algorithms utilize two-hop neighborhood information.

In this work we propose a generalization to the approach used by Lim and Kim in their dominating pruning algorithm [12]**.** Their approach uses a greedy set cover (GSC) strategy in order to compute the dominating set of each two-hop neighborhood of the nodes involved in the diffusion of a packet.

3 Adaptive Dominant Multicast Protocol (ADMP)

In [3] we demonstrate how DPUMA effectively increase the delivery ratio of PUMA while incurring far less retransmission overhead. However, as it is shown in Figure 4, this is not longer true for scenarios where nodes have high mobility. The reason for this behavior is that, in general, the performance of protocols that rely on the freshness of topological information is strongly impacted by the relative mobility among nodes. It would be desirable to have an approach capable of delivering the reliability achieved by PUMA under light loads or high node mobility, and the one achieved by DPUMA under high loads with low mobility.

The first step towards such a protocol is to get an accurate view of the instantaneous levels of relative mobility and contention, so that; nodes were able to select the operation mode that performs best under each condition. The next section describes two simple mechanisms to detect the degree of relative mobility and the degree of local contention.

3.1 Detection of Relative Mobility and Contention Levels

To compute the level of relative mobility, each node keeps track of how its one-hop neighborhood has changed between two consecutive sampling periods, then, nodes compute an exponential weighted moving average to avoid reacting too fast to changes in their perceived relative mobility.

We define instantaneous relative mobility m as $[d/(r+d)]/s_p$, where s_p is the length of the sampling period, d is the number of new or missing one-hop neighbors detected in the current sampling period with respect to the neighbors detected in the

previous sampling period, and r is the number of neighbors that did not change from the previous sampling period with respect to the current sampling period. The degree of relative mobility υ_n during sampling period n is:

$$v_n = (\alpha - 1)v_{n-1} + \alpha m \quad (1)$$

Where α is a constant used to assign weight to the previous (υ_{n-1}) and newly calculated values (m) of the degree of relative mobility.

Another aspect that has a strong influence over the performance of protocols that use contention-based medium access control (MAC) protocols is traffic load. To measure one-hop contention we propose a simple and very intuitive metric that is based on the ratio between the number of received signals with errors and the total number of received signals during a fixed period of time. This ratio tries to approximate the current probability of a successful transmission. Then, as in the previous case, we use an exponential weighted moving average to cope with sudden and short term variations. We define the instantaneous contention level c as $(e/t)/s_p$, where e is the number of signals with errors received during the sampling period, and t is the total number of signals received during the sampling period. The *degree of contention* γ during sampling period n is defined as:

$$\gamma_n = (\beta - 1)\gamma_{n-1} + \beta c \quad (2)$$

Analogously to the previous case, β is a constant used to assign weight to the previous (γ_{n-1}) and newly calculated values (c) of the degree of contention.

The current default value for α and β is 0.2. However, our results show that the performance of ADMP is not very sensitive to these parameters.

3.2 General Augmented Greedy Set Cover (GAGSC)

In the case of DPUMA, since nodes already interchange one-hop topology information, they can, almost for free, gather information about contention and mobility of the nodes that belong to their two-hop neighborhood. Here we present a novel algorithm that takes advantage of this information, and that makes more fine-grained decisions when selecting the amount of redundancy used to relay a packet.

Our algorithm, which we have called General Augmented Greedy Set Cover (GAGSC), has two main phases. In the first phase, based on their local contention and relative mobility degree, two-hop neighbors are assigned with a coverage value which reflects the amount of redundancy that will be used to cover that node, or in other words, the number of one-hop neighbors that the algorithm will try to use to cover (or to dominate) that particular two-hop neighbor. In the second phase, the algorithm uses a greedy strategy by selecting one-hop neighbors with the highest value in a utility function f_u. $f_u(\cdot)$ of a given one-hop neighbor b is proportional to number of two-hop neighbors (nn) which are covered by b, and inversely proportional to the exponential of b's contention (cd) and relative mobility degrees (md). By using this utility function, GAGSC tend to favor one-hop neighbors with lower local contention and mobility degrees over nodes which might cover more two-hop neighbors but that have larger values for these metrics.

In particular, our current implementation of ADMP uses the following utility function.

$$f_u = nn \cdot e^{-1.5cd-1.5md} \tag{3}$$

Figure 1 shows a partial plot of the utility function that only considers the contention and relative mobility degrees. From the plot of the function, it is easy to see how nodes with low contention and mobility degrees will tend to be selected first than nodes with larger values in these metrics.

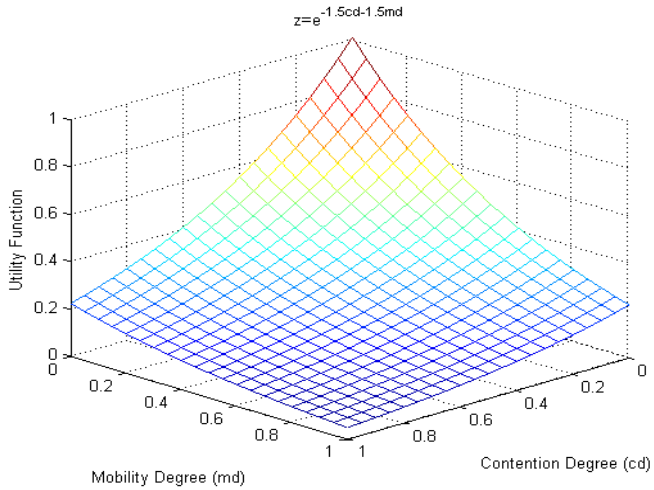

Fig. 1. Plot of the utility function used to select one-hop neighbors

Now, we are ready to make a more formal description of the GAGSC algorithm. As in [12] we use N(u) to represent the neighbor set of a given node u (including u) and N(N(u)) to represent the neighbor set of N(u) (i.e., the set of nodes that are within two hops from u). When a mesh member v receives a data packet from u, it selects a number of forwarding nodes that can cover (with the adequate redundancy) all the nodes in N(N(v)). u is the previous relaying node, hence nodes in N(u) have already received the packet, and nodes in N(v) will receive the packet after v rebroadcast it. Therefore, v just needs to determine its forwarding list $F(u,v)$ from $B(u,v) = \text{N}(v) - \text{N}(u)$ to cover all nodes in $U(u,v) = \text{N}(\text{N}(v)) - \text{N}(u) - \text{N}(v)$.

The GAST algorithm works as follows.

Phase 1: For all two-hop neighbor node $x \in U(u,v)$, let *coverage*(x) be its corresponding coverage value, i.e., the number of one-hop neighbors that the algorithm will try to use to cover (or dominate) that particular node. Now, using rules like "*if v's contention is low and v's mobility is low and x's contention is low and x's mobility is low then set coverage(x) equal to 3*" or "*if v's contention is high and v's mobility is low and x's contention is high and x's mobility is low then set coverage(x) equal to 1*". Nodes use two threshold values to decide whether their current contention and relative mobility degrees are low or high. GAGSC employs 16 rules that correspond to all possible combinations of high and low contention and

mobility degrees for the two-hop node x under consideration and the node v which is computing its forwarding list.

The intuition behind these rules is as follows. When nodes detect low contention degree in the channel, they are safe to use high redundancy during the dissemination of data, or in other words, they can assign a high value to *coverage*(x), so that nodes will try to cover (or dominate) their two-hop neighbors with as many one-hop neighbors as possible. This mode of operation is similar to PUMA where data packets are flooded within the mesh. On the other hand, when high degrees of contention are detected, nodes will try to compute a CDS which is as small as possible; hence they assign low values to *coverage*(x). In this last situation, nodes operate in a mode similar to DPUMA, so that they are able to reduce the redundancy used when disseminating data packets. Analogously, when nodes detect low relative mobility degree, they are safe to rely on their current topology information and compute CDS which are as small as possible. Finally, when nodes perceive a high mobility degree, it is better to flood the mesh which is robust to topological changes because it does not make any assumption about the presence of a given node in the one-hop neighborhood, so nodes will assign high values to *coverage*(x).

It is important to remark that even if a given two-hop neighbor, say x, is assigned with a coverage value $C > 1$, there is no guarantee that at the end of Phase 2, x is going to be covered by C one-hop nodes. This situation is apparent on Figure 2, where node x will be covered by at most one one-hop neighbor (a) no matter what is the value assigned to *coverage*(x).

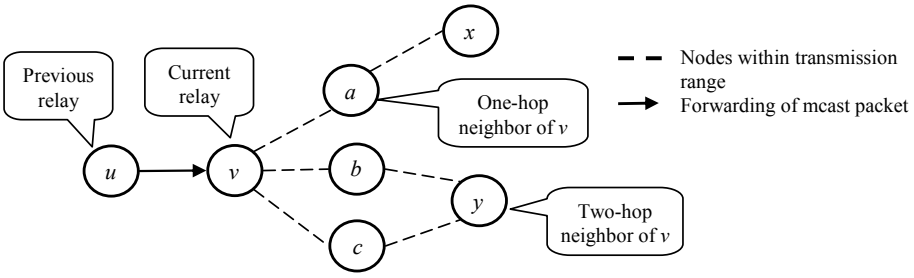

Fig. 2. Two-hop neighborhood of node v that has not been covered so far when receiving a multicast data packet from node u

Phase 2: Compute: $F(u,v)$.

Let $F(u,v)$ be the forwarding list to be computed, Z be the set of nodes that have been covered up to the current iteration which is initially empty, $S_i = N(v_i) \cap U(u,v)$ be the set of two-hop nodes that can be covered by node v_i, $K = \bigcup S_i$ be the set of nodes that have to be covered at the end of the execution of the algorithm, and $mob(v_i)$ be the relative mobility degree of v_i, and $con(v_i)$ be the contention degree of v_i for all $v_i \in B(u,v)$.

1. Find the 1-hop node v_m whose $f_u(|S_m|, con(v_m), mob(v_m))$ is a maximum
2. For all node $x \in S_m$ make $coverage(x) = coverage(x) - 1$ in all S_i
3. For all S_i, make
   - $E_i = \{x \mid x \in S_i \text{ and } coverage(x) = 0\}$,
   - $S_i = S_i - E_i$, $Z = Z \cup E_i$, $S_m = \phi$ and

- $F(u,v) = F(u,v) \cup \{v_m\}$

4. Stop if $Z=K$ or if $F(u,v) = B(u,v)$; otherwise, goto step 1
5. If $F(u,v) = B(u,v)$ then $F(u,v) = \varphi$ where φ is an special marker used to denote that every one-hop neighbor (if it has not done that before) has to retransmit the data packet

It is important to note that, in order to mimic the behavior of PUMA and achieve its resilience to continuous topological changes it is not enough to designate all one-hop neighbors as forwarders. We also need a way to specify that any mesh member, that happens to be in the one-hop neighborhood, has to retransmit a packet, even if the current forwarding node has not perceived it yet. In the specification of our algorithm, the φ marker introduced in Step 5 plays this role.

3.3 Forwarding Data Packets Within The Mesh

Finally, to complete the description of the GAGSC algorithm, we present how it is used in the context of disseminating a data packet within a mesh.

- When a mesh member *v* receives a multicast data packet from its transport layer, it determines its forwarding list *F(-,v)* using GAGSC, then the node piggybacks *F(-,v)* in the data packet and transmits it.
- When *v* receives a multicast data packet from a mesh member *u*
  - If $v \in F(*,u)$ or *F(*,u)* equals φ, it uses GAGSC to determine its forwarding list *F(u,v)*, piggybacks *F(u,v)* in the data packet and retransmit it.
  - If $v \notin F(*,u)$, it just accepts the packet
- When *v* receives a multicast data packet from a non-mesh member *u*, *v* computes the forwarding list *F(-,v),* piggybacks *F(-,v)* in the data packet and retransmit it. In this case, *u* can be a sender or a next hop in the path from the sender to the core.

4 Experimental Results

We compared the performance of ADMP against the performance of PUMA, DPUMA and ODMRP. We used the discrete event simulator Qualnet version 3.5. The distribution of Qualnet itself has the ODMRP code. Each simulation was run for five different seed values with the exception of the mobility scenario which uses 20 seeds. This is necessary to obtain representative results because a given seed can generate very different mobility traces for different protocols. To have meaningful comparisons, all timer values (i.e., interval for sending JOIN requests and JOIN tables in ODMRP and the interval for sending multicast announcements in the PUMA family) were set to 3 seconds. Unless other values are specified, Table 1 lists the details about the simulation environment. The metrics used are packet delivery ratio and average number of data packets relayed.

In our first experiment we varied the packet size from 64 to 1024 bytes. There are two senders and one group composed of 30 nodes. Only one sender belongs to the multicast group. From Figure 3 we can observe how PUMA performs very well

for small packet sizes but as the packet size increases, its packet delivery ratio drops dramatically.

Table 1. Simulation environment

Simulation Environment			
Total Nodes	50	Simulation time	100s
Node Placement	Random	Simulation area	1300×1300m
Mobility Model	Random Waypoint	Channel Capacity	2000000 bps
Pause Time	10s	MAC protocol	IEEE 802.11
Min-Max Vel.	0 – 10 m/s	Data Source	MCBR
Transmission Power	15 dbm		
Number of packets sent per source			1000

This experiment shows how for medium to large packet sizes those protocols that compute dominating sets (DPUMA, and ADMP) achieve higher delivery ratios than protocols like PUMA that use much more redundancy. From Figure 3 we can also observe how ADMP performs very close to the best case of the base protocols, namely, close to PUMA for small packets and close to DPUMA for large packets. This is a strong indication that ADMP nodes effectively detect their current conditions on the channel and select the appropriate mode of operation.

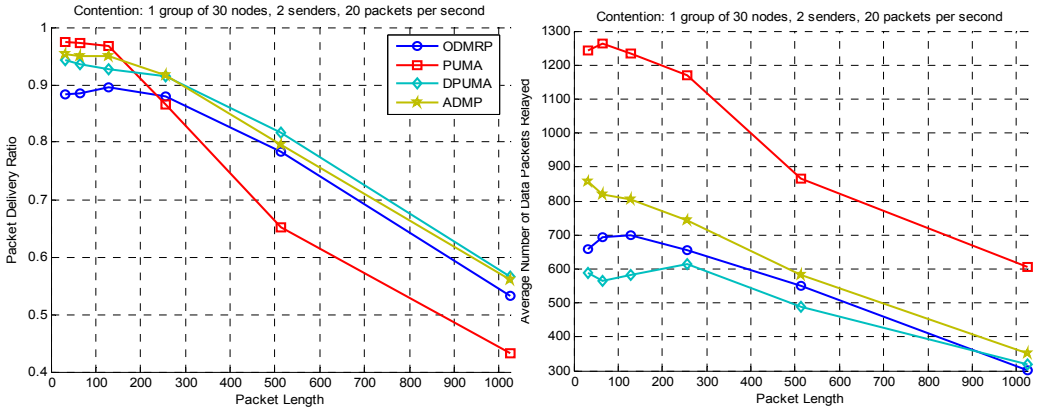

Fig. 3. Packet delivery ratio and average number of retransmission when varying the packet size

In our second experiment we varied from 1 to 50m/s the nodes' speed (with a pause time of 0 seconds). In this experiment we show how effective is the proposed metric to detect relative mobility, and how this information is used by the nodes to autonomously decide which mode of operation has to be used. As it can be seen on Figure 4, ADMP performs similar or better than the best of the base options of the family of PUMA protocols (PUMA, DPUMA). Again, this is a strong indication that ADMP nodes effectively detect the current mobility condition and select the appropriate mode of operation.

A very interesting situation is that for speeds between 10m/s and 30m/s, ADMP performs even better than the base protocols. The reason is that nodes can

independently select the current "best strategy", so nodes which are in different regions of the MANET can use the operation mode that best fit that particular region.

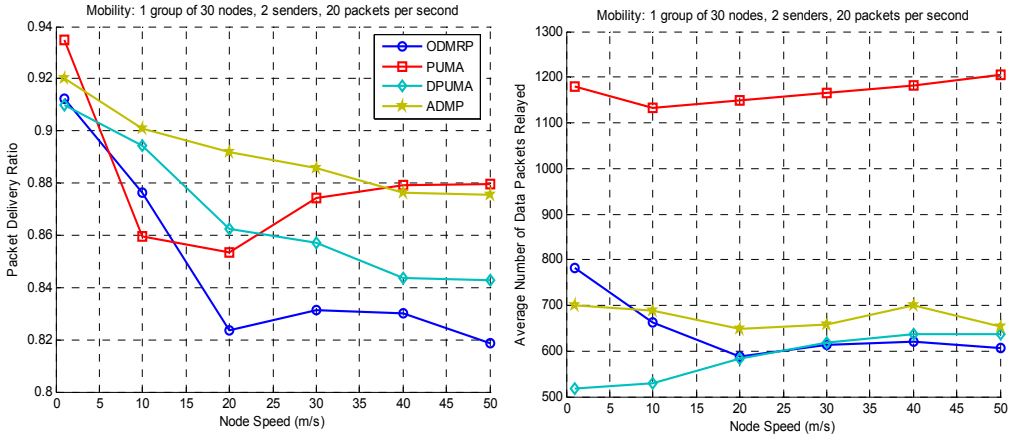

Fig. 4. Packet delivery ratio and average number of retransmission when varying the speed of the nodes

Finally, Figure 5 shows the behavior of the degree of relative mobility of a given node for different values of the speed of the nodes. From the figure we can observe that as the speed of the nodes increases, the height of the peaks in the graphs also increases. The peaks in the graphs correspond to the time when a multicast announcement is flooded across the MANET. This is also the time when the all the topology information is updated.

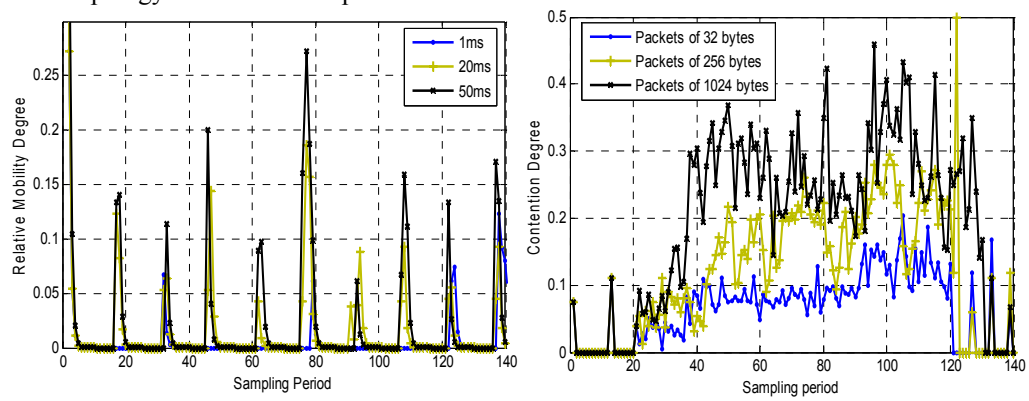

Fig. 5. Values taken by the Relative Mobility and Contention Degree metrics for different values of the speed of the nodes and packet length

Figure 5 also shows the behavior of the contention degree perceived by a given node for different values of the length of the data packets. As well as in the case of the relative mobility degree, we can observe a good correlation between the proposed metrics and the conditions in the network. As we saw in the previous paragraphs, our protocols take advantage of this information to tune the amount of redundancy used to transmit data packets.

5 Conclusions

In this paper we presented ADMP, a mesh-based multicast protocol that carries out its three basic tasks (electing a core, establishment of the mesh and getting 2-hop neighborhood information) by flooding a single control packet per each multicast group. When diffusing a data packet over the mesh, nodes in ADMP use GAGSC to compute a dominating set of the mesh taking into account the nodes' contention and relative mobility degrees. The size of the dominating set reflects the amount of redundancy that is used to diffuse a packet across the mesh. Our results show that for all the scenarios ADMP performs similar or better than PUMA, APUMA and consistently better than ODMRP. Our current research focuses on core election protocols and the way in which core placement affects the topology of the mesh as well as the delay and delivery ratio of mesh-based multicast protocols.

References

1. E. Royer and C. Perkins, "Multicast operation of the ad hoc on-demand distance vector routing protocol," in Proceedings of Mobicom, August 1999
2. L. Ji and M. S. Corson, "A lightweight adaptive multicast algorithm", in Proceedings of IEEE GLOBECOM 1998, December 1998, pp. 1036–1042
3. R. Menchaca-Mendez, R. Vaishampayan, J. J. Garcia-Luna-Aceves, K. Obraczka, "DPUMA: A Highly Efficient Multicast Routing Protocol for Mobile Ad Hoc Networks," ADHOC-NOW 2005: 178-191
4. J. J. Garcia-Luna-Aceves and E.L. Madruga, "The core assisted mesh protocol," IEEE Journal on Selected Areas in Communications, Special Issue on Ad-Hoc Networks, vol. 17, no. 8, pp. 1380–1394, August 1999
5. S. J. Lee, M. Gerla, and Chian, "On-demand multicast routing protocol," in Proceedings of WCNC, September 1999
6. R. Vaishampayan and J.J. Garcia-Luna-Aceves, Efficient and Robust Multicast Routing in Mobile Ad Hoc Networks , Proc. IEEE MASS 2004: The 1st IEEE International Conf. on Mobile Ad-hoc and Sensor Systems, Fort Lauderdale, Florida, October 25-27, 2004
7. S. K. Das, B. S. Manoj, and C. S. Ram Murthy, "A dynamic core based multicast routing protocol for ad hoc wireless networks," in Proceedings of the ACM MobiHoc, June 2002
8. S. Lee and C. Kim, "Neighbor supporting ad hoc multicast routing protocol," in Proceedings of the ACM MobiHoc, August 2000
9. Deering S. E., et-al, "The PIM Architecture for Wide-Area Multicast Routing", IEEE/ACM Transactions on Networking, Vol.4, No.2, April 1996
10. R. Perlman, "An algorithm for distributed computation of a spanning tree in an extended lan," in ACM Special Interest Group on Data Com. (SIGCOMM), 1985, pp. 44–53
11. Wei Lou, Jie Wu, "On Reducing Broadcast Redundancy in Ad Hoc Wireless Networks," IEEE Transactions on Mobile Computing, Vol. 1, Issue 2 (April 2002), Pages: 111 – 123
12. H. Lim and C. Kim, "Flooding in wireless ad hoc networks," Computer Communications, vol. 24, February 2001
13. M.A. Spohn and J.J. Garcia-Luna-Aceves, "Enhanced Dominant Pruning Applied to The Route Discovery Process of On-demand Routing Protocols," Proc. IEEE IC3N 03: Twelfth Int. Conf. on Computer Com. and Networks, Dallas, Texas, October 20 - 22, 2003