Chapter 2

Scalable Multicast Routing in Mobile Ad Hoc Networks

Rolando Menchaca-Mendez* and J.J. Garcia-Luna-Aceves†

*Computer Engineering Department
University of California, Santa Cruz
Santa Cruz, CA 95064, USA
menchaca@soe.ucsc.edu

†Palo Alto Research Center, 3333 Coyote Hill Road
Palo Alto, CA 94304, USA
jj@soe.ucsc.edu

In the context of mobile and ubiquitous systems there is an increasing number of applications where data has to be transmitted, not to a single node or person, but to dynamic groups of nodes or people. This call for efficient multicast routing protocols capable of efficiently use the available bandwidth as well as the usually restricted hardware resources such memory and power. Moreover, given the current and expected sizes of this type of networks, multicast protocols have to be designed to scale up to hundreds of nodes. Here, we describe Hydra, the first multicast routing protocol for MANETs that establishes a multicast routing structure approximating the set of source-rooted shortest-path trees from multicast sources to receivers, without requiring the dissemination of control packets from each source of a multicast group. Hydra accomplishes this by (a) dynamically electing a core for the mesh of a multicast group among the sources of the group, so that at most one control packet is disseminated in the network to announce the existence of the group and (b) aggregating multicast routing state in the nodes participating in multicast meshes, so that redundant control packets are not disseminated towards the receivers of a group. We also present an improved version of PUMA which is a receiver-initiated multicast protocol that for each multicast group periodically floods a single control packet which is used to elect a core for the group and to build and maintain the multicast mesh. We present simulations results for WiFi and TDMA MAC protocols illustrating that Hydra and PUMA attain comparable or higher delivery ratios than ODMRP, but with considerably lower end-to-end delays and, in the case of Hydra, far less data overhead.

Keywords: Multicast routing; mesh-based multicast protocols; scalable multicast; state aggregation; ad hoc networks.

2.1. Introduction

The objective of a multicast routing protocol for mobile ad hoc networks (MANET) is to enable communication between a sender and a group of receivers in a network where nodes are mobile and may not be within direct wireless transmission range of each other. Any MANET node can act as traffic originator, destination, or forwarder. Hence, 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, as well as many new emerging ubiquitous applications such as those envisioned by Mark Weiser.[13] In both civilian and military scenarios, we can find a wide range of possible applications to the multicast communication pattern, for instance, group coordination (rescue team leaders deliver instructions to a given subset of their team members), event notification (attendants of a conference receive updates regarding previously defined interests or a group of medical experts are notified of an emergency in their expertise area). Due to the nature of the underlying hardware, these multicast protocols have to be designed to efficiently use the available bandwidth as well as nodes' energy.

We can classify multicast routing protocols for MANETs by the type of routing structure they construct and 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 (one per each sender) to deliver packets from sources to receivers. Several tree-based multicast routing protocols have been reported (e.g. Refs. 6 and 9). These approaches have proven to deliver adequate performance in wired networks. However, in the context of MANETs, establishing and maintaining a tree or a set of trees in the presence of frequent topology changes incur substantial exchange of control messages, which has a negative impact in the overall performance of the protocol.

On the other hand, a mesh-based multicast routing protocol maintains a mesh consisting of a connected sub-graph of the network containing all receivers of a particular group and the relays needed to maintain connectivity. Maintaining a connected component is far simpler than maintaining a tree and hence mesh-based protocols tend to be simpler and more robust. Three representatives of this kind of protocols are the Core Assisted Mesh Protocol (CAMP),[3] the On-Demand Multicast Routing Protocol (ODMRP),[8] and PUMA.[12] A potential concern in mesh-based schemes is that, under high channel contention, these protocols may have poor performance if too many redundant relays are involved in the forwarding of multicast traffic.

Whether multicast routing protocols for MANETs build multicast trees or meshes, all of them are based on network-wide dissemination of control packets to inform the rest of the nodes about the existence of multicast groups. In core-based or receiver-initiated schemes, only one node originates the dissemination of information about a multicast group reaching all other nodes, and receivers send explicit requests towards the core to join the group. In contrast, source-based or

sender-initiated schemes have each multicast source originate the dissemination of state information that reaches all nodes in the network. Given that the sender-initiated protocols proposed to date use per source flooding, they do not scale well as the number of groups and sources increases. However, they can provide shortest paths from sources to destinations and avoid hot spots. On the other hand, core-based protocols incur far less overhead, but they do not establish shortest paths from sources to destinations, which leads to higher delays than the ideal shortest paths from sources to receivers.

The work presented here is motivated by the desirability of providing the best features from the two alternatives in the existing design space summarized above, and which we discuss in more detail in Section 2.2. Section 2.3 discuses Hydra, a multicast routing protocol that creates a multicast mesh formed by a mixture of source-specific and shared sub-trees (or sub-meshes) using as few control packets as receiver initiated schemes do. The key ideas behind Hydra are: restricting the dissemination of control packets to those regions of the network where other dynamically designated sender has previously discovered receivers, aggregation of control messages from non-core senders, and electing a sender as the core in non-destructive manner. Section 2.4 discuses PUMA,[12] a receiver-initiated mesh-based multicast routing protocol in which receivers join a multicast group using the address of a special node (core), without the need for network-wide dissemination 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. Within a finite time, each router has one or multiple paths to the elected core. All nodes on shortest paths between any receiver and the core collectively form the mesh of the multicast group.

Section 2.5 describes the results of simulation experiments used to study Hydra's and PUMA's performance with that of ODMRP by considering different numbers of sources, group sizes, network density and the use of 802.11 or TDMA as the underlying MAC protocol. The results illustrate the performance benefits that should be expected from the approaches implemented in Hydra and PUMA. Both protocols provide substantial performance improvements over ODMRP even in scenarios involving relatively small networks with few multicast sources. Hydra and PUMA attain the same or better delivery ratios than ODMRP, and incurring end-to-end delays that are close to an order of magnitude smaller than in ODMRP. The simulation experiments also compare different versions of the aggregation algorithms implemented in Hydra.

2.2. Related Work

The multicast ad hoc on-demand distance vector protocol (MAODV)[10] maintains a shared tree for each multicast group consisting of receivers and relays. Sources

acquire routes to the group on demand in a way similar to the ad hoc on demand distance vector protocol (AODV).[9] Each multicast group has a group leader who is the first node joining the group. The group leader is responsible for maintaining the group's sequence number, which is used to ensure freshness of routing information. The group leader periodically transmits a group hello packet to become aware of reconnections. Receivers join the shared tree by means of a special route request (RREQ) packet. Any node belonging to the multicast tree can answer to the RREQ with a route reply (RREP). A sender joins the group through the node reporting the freshest route in a RREP with the minimum hop count to the tree. Data are delivered along the tree edges maintained by MAODV. If a node that does not belong to the multicast group wishes to multicast a packet, it has to send a non-join RREQ, which is treated similar to RREQ for joining the group. As a result, the sender finds a route to a multicast group member. Once data is delivered to a group member, the remaining members receive the data along the multicast tree.

The adaptive demand-driven multicast routing protocol (ADMR)[5] maintains a source-based multicast tree for each sender of a multicast group. A new receiver performs a network-wide flood of a multicast solicitation packet when it needs to join the multicast group. Each source replies to the solicitation and the receiver sends a receiver join packet to each source that answered the solicitation. Each source-based tree is maintained by periodic keep-alive packets from the source, which allow intermediate nodes to detect link breaks in the tree by the absence of data or keep-alive packets. A new sender also sends a network-wide flood to allow existing group receivers to send receiver joins to the source. MZR[2] like ADMR, maintains source-based trees. MZR performs zonal routing; and hence the dissemination of control packets is less expensive.

In ODMRP,[8] group membership and multicast routes are established and updated by the sources. Each multicast source broadcasts *Join Query* (*JQ*) packets periodically, and these are disseminated to the entire network to establish and refresh group membership information. When a *JQ* packet reaches a multicast receiver, it creates and broadcasts a *Join Reply* (*JR*) to its neighbors stating a list of one or more forwarding nodes. Nodes receiving *JR* listing them as part of forwarding groups forward the replies with its own list of forwarding nodes. A *JR* is propagated by each forwarding group member until it reaches a multicast source via the selected paths. This process constructs (or updates) the routes from sources to receivers and builds a mesh of nodes, the forwarding group. A source can multicast data packets to multicast receivers via selected routes and forwarding groups. DCMP[1] is an extension to ODMRP that designates certain senders as cores and reduces the number of senders performing flooding. NSMP[7] 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 use of multiple nodes flooding control packets for each group.

CAMP[3] avoids the need for network-wide disseminations 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.

2.3. Hydra

2.3.1. *Overview*

As it is the case in ODMPR, multicast sources in Hydra periodically broadcast *Join Query* (*JQ*) packets to establish a partial ordering of the nodes in the network. In the case of ODMRP and Hydra, the ordering is based on the nodes' distances in hops to the sources. This ordering is further used to route *Join Reply* (*JR*) packets from receivers to sources, forcing intermediate nodes to join either a mesh or a tree. However, Hydra uses three mechanisms to build a routing structure as close as possible to a set of source-rooted breadth-first trees (or meshes composed of the union of breadth-first trees) spanning all the receivers while incurring as few control overhead as possible.

In contrast to ODMRP, Hydra uses an elected source as the core of the group, and this is the only source whose *JQ*s reach the entire network. Non-core sources take advantage of the routing state established by the core to identify connected sub-graphs containing one ore more non-core sources and receivers of the group. This way, the scope of the dissemination of *JQ*s from non-core sources is restricted to these connected regions, and other parts of the network are not flooded with unnecessary control information.

In addition, Hydra identifies regions of the network where two or more sources share common sub-graphs (meshes or trees) and performs routing-state aggregation, so that nodes located inside of those common regions only keep routing state regarding one of the aggregated sources and receive *JQ*s and *JR*s only from that source. To detect the boundaries of a common sub-graph, Hydra compares the orderings established by previous sources with the ordering that is being established by the current *JQ* from a non-core source. If the ordering induced by the *JQ* is equivalent to the ordering established by a prior *JQ* from another source, then the current *JQ* is not forwarded any further and the two sources that have equivalent orderings are considered as aggregated. Two partial orderings over a graph are *equivalent* if the gradient vectors among neighbors obtained from the two orderings are the same. As the number of senders increases, the likelihood of finding equivalent regions also increases, because nothing prevents a source to share different sub-graphs with different sources, or a given sub-graph to be shared by more than two sources. This property helps the scalability of Hydra with respect to the number of sources. We

also note that, while Hydra takes advantage of having a core, it is not necessary for its correct operation.

2.3.2. *Control signaling*

Hydra opportunistically groups control messages of different sources and groups into a single control packet. However, in the rest of our description, we focus on the signaling intended for a specific group.

2.3.2.1. *Join queries*

The first active multicast source for a given group considers itself to be the core for that group and states so in the *Join Query* (*JQ*) packets it broadcasts every *join query period*. *JQ*s inform other nodes of the existence of the multicast group and its current core, and create a partial ordering of the network based on the distance in hops from each node to the current core. If two or more sources become active concurrently in the same partition, a distributed election is held. The details of the election algorithm are presented in Section 2.3.3.

Sources other than the core in the same multicast group are considered regular sources or *non-core senders*. Non-core senders transmit Non-Core Join Query (*JQnC*) packets to build their own trees or meshes. A *JQ* is composed of a packet type identifier, the address of the group, the address of the core, a TTL, the distance to the core and a sequence number. In addition, a *JQnC* contains the address of the non-core sender, the distance to the non-core sender, and the address of the parent towards the core that is used to route *JQnC*s towards the mesh of the core.

Because non-core nodes benefit from the routing structure created by the core, the transmission of *JQnC*s is roughly synchronized with the reception of *JQ*s. Upon receiving a *JQ* with a larger sequence number, non-core senders wait for a random period of time which is much smaller than the join query period. However, it is also long enough to allow the establishment of the routing structure of the core before transmitting their next *JQnC* that refreshes the routing information for that source. *JQnC*s are also sent by non-core senders when they have data to sent but no route is known to the receivers and when the sender has not received a *JQ* from the core in the last two consecutive join query periods.

The objective of the combined use of *JQ*s and *JQnC* for a given multicast group is to order all nodes with respect to the core of the group, and to make the multicast routing structure (mesh or tree) as close as possible to the aggregation of the source trees of all the multicast sources in the group. *JQ*s must be sent to all nodes; however, the overhead due to the dissemination of *JQnC*s is reduced using two mechanisms.

The first way of reducing the overhead incurred with *JQnC*s consist of disseminating them only to a subset of the network composed of nodes that are part of the mesh or tree established by the core, nodes that lay in the path from the

non-core sender to the core, and nodes located at most k hops away from them. The set of nodes that forward $JQnC$s for a given non-core sender is called the source's *k-restricted region of interest* or simply *k-restricted region*. This way, the dissemination of $JQnC$s is carried out only among nodes that are likely to be close to receivers, and other regions of the network do not receive irrelevant control information.

The optimal value of k for a k-restricted region depends on the topology of the network as well as on the mobility of the nodes and on the length of the join query period. In our experiments, a sensitivity analysis showed that 1 is a reasonable value for k. In general, as the value of k grows, more redundancy is introduced, which helps coping with mobility. In the worst case, the k-restricted regions of interest cover the entire network, and the scheme degenerates to the case of flooding the network with control packets per sender per group as in ODMRP.

Figure 2.1 illustrates the above concepts. The figure shows two multicast groups, G_1 and G_2 with their respective cores, S_c and S_j. Each group has a non-core source; S_m for G_1 and S_i for G_2. The mesh of the core S_c of G_1 is composed of nodes labeled m_{G1} and the mesh of the core S_j of G_2 is composed of nodes labeled m_{G2}. In the figure, the 1-restricted region of interest of S_m is delimited by a dotted line. We observe that it contains the mesh constructed by the core of the group S_c, which is delimited by a solid line, as well as the nodes located one hop way from the mesh or from the path from the non-core sender to the mesh. $JQnC$s generated by S_m are forwarded only by such nodes as node x or node y, which are located inside of

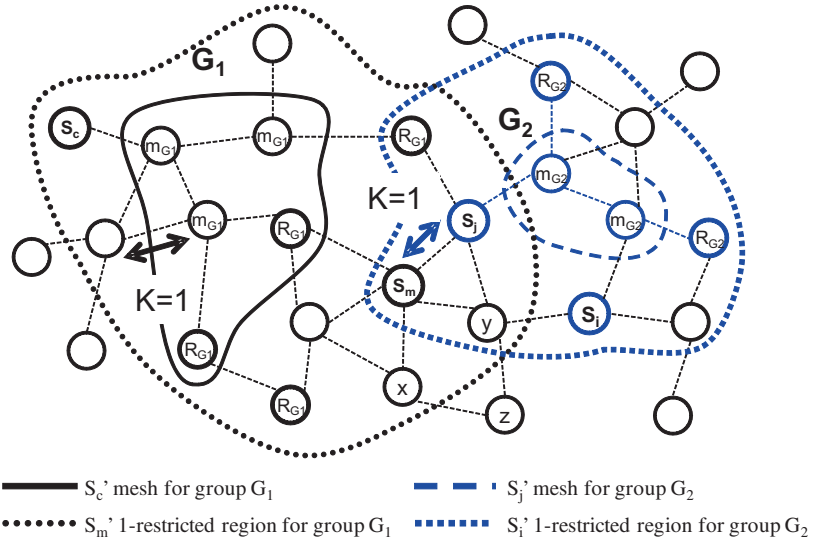

Fig. 2.1. k-restricted region. A k-restricted region of a non-core sender is a set of nodes that forward Join Queries generated by that source.

the 1-restricted region of interest, and nodes located outside of this region, such as node z, may receive the packet but does not forward it. The figure also presents a similar situation for group G_2.

The second way to reduce the number of $JQnC$s sent to the network and the state kept at nodes is to find common sub-graphs and perform multicast state aggregation on these particular regions of the network. Nodes located in a common sub-graph only receive and forward JQs (or $JQnC$s) of one of the sources that share that sub-graph and keep state about the source whose join queries are forwarded. Figure 2.2 shows an example of a network in which two sources, S_c and S_2 share a common sub-graph. With this goal, we propose the *Dissemination of Multicast Aggregated-State* (*DIMAS*) algorithm. From the standpoint of message complexity, DIMAS behaves as simple flooding in the worst case. However, depending on the perceived current topology of the network, DIMAS stops disseminating control packets of non-core senders before covering the whole k-restricted region.

To do so, nodes determine if they are located in the boundary of a region that would likely be ordered by a JQ of a given source (say S_i) in an equivalent way as it was already ordered by a previous dissemination of JQs generated by a different source (say S_j). If this is the case, then nodes stop the dissemination of control packets from S_i and mark that source as aggregated with S_j. Beyond this point, data packets generated by S_i are forwarded as if they were data packets from S_j. DIMAS (i, snd) is executed at node i when it is about to relay a $JQnC$ from sender snd to decide whether the message is sent or the sender snd is aggregated at node i. The three rules used to decide when to aggregate are described below.

Rule 1: Upon reception of a $JQnC$ with a larger sequence number, nodes wait a period of time equal to FWD_DLY to collect packets forwarded by other neighbors. Based on the distances stated in these $JQnC$, nodes compute their own distance to

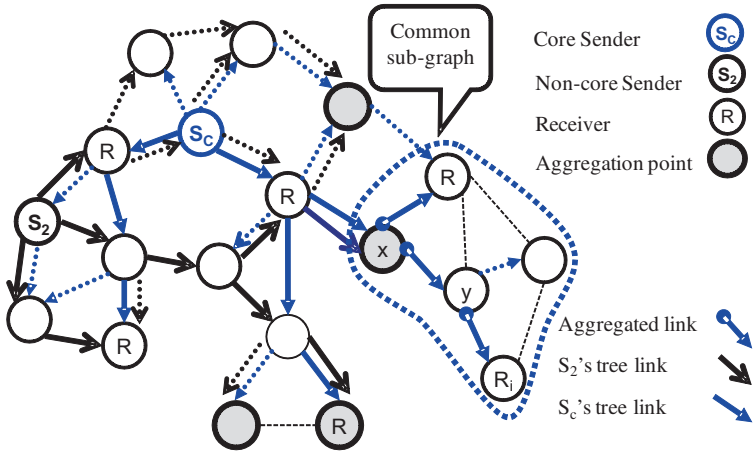

Fig. 2.2. Aggregated sub-graphs.

the source and a set of pairs $(neighbor, gradient)$ where the gradient is computed as the node's distance minus the distance reported by each neighbor. Then, nodes check if they have recently received JQs or $JQnC$s from other source, such that the set $\{(neighbor, gradient) : gradient \geq 0\}$ matches with the one computed for the current source. If that is the case, nodes do not forward the $JQnC$ and mark the senders as aggregated. If there is no match, nodes forward their own $JQnC$ (with their computed distance).

Rule 2: If a node receives $JQnC$s generated by different sources at roughly the same time (within a FWD_DLY period) and there is a match between the sets of gradient pairs, the node forwards the control packet corresponding to the source with the *largest identifier* and stops the control packets corresponding to the other sources.

Rule 3: The core source cannot be aggregated to any non-core source, because the JQs generated by the core must cover the whole network. Aggregation is allowed only either among non-core sources, or with the core aggregating non-core sources.

DIMAS has two different variants. The first is called Total-DIMAS and is implemented by Hydra-TA which allows the aggregation of a non-core source with the core or any other source. The second variant is called Core-DIMAS and is implemented by Hydra-CA. This latter version only allows the aggregation of non-core senders with the core but not with other non-core senders.

2.3.2.1.1. Disseminating Aggregation Maps.

JQs and $JQnCs$ can be augmented with an *aggregation map* containing pairs of node identifiers of the form $(aggregated, aggregatedWith)$, where $aggregated$ is the identifier of a source that is aggregated to other source with identifier $aggregatedWith$. The aggregation maps are stored at downstrem nodes and can be used to decide which source's routing information has to be used when forwarding a data packet of an aggregated source. Aggregation maps permit the forwarding of multicast data packets from sources whose state has been aggregated. In effect, they are routing-table extensions, as we discuss in Section 2.3.4.

2.3.2.2. *Join replies*

After receiving a JQ or a $JQnC$ with a fresher sequence number, a receiver generates a *Join Reply* (JR). A JR contains a packet type identifier, the address of the group, the address of the sender (either core or non-core), the distance to the sender, the address of the selected parent and a sequence number.

A JR is routed back to the source following the reverse direction of the gradient of the distances established by the partial ordering obtained from the diffusion of JQs. JRs travel hop-by-hop forcing intermediate notes to join the multicast routing structure, until the JRs reach the first node that is already part of the multicast routing structure or a multicast source. If a JR is generated inside of a shared

sub-graph it travels along the shared region establishing routing state only about a single source. However, as soon as the JR reaches a node in which one or more $JQnC$ were stopped by the DIMAS algorithm, a new JR is generated for each $JQnC$ that was stopped in that particular node. Then, these JRs follow their own ways towards the different sources.

From the example shown in Figure 2.2, we observe that a JR generated by receiver R_i travels to node y, and establishes routing state regarding only the core S_c. However, when the JR reaches the aggregation point x, two independent JRs are forwarded towards S_c and S_1. As it is also shown in the figure, these two JRs can follow independent paths towards their respective sources.

When nodes forward JRs, they can select one or more parents to reach a source. In the first case the resulting routing structure is a tree, while in the latter case it is a mesh. In Hydra, we chose to keep state per source because we want to avoid forwarding data packets to places where they are not needed, e.g. toward other senders. This last design decision implies an increase in the state kept at the nodes. However, as our simulations show, bandwidth is a much more stringent bottleneck than memory, and spending extra memory in order to save bandwidth is a good tradeoff. Furthermore, keeping state per sender is necessary if the protocol has to support a source-specific multicast service model.[4]

2.3.3. *Non-destructive core election*

If a source has data to send to a multicast group, it first determines whether it has received a JQ from the core of that group. If the node has, it adopts the core specified in the JQ it has received, and it transmits a $JQnC$ that also advertises the same core for the group. Otherwise, it considers itself the core of the group and starts transmitting JQs periodically to its neighbors, stating itself as the core of the group and a 0 distance to itself. Nodes propagate JQs based on the best JQ they receive from their neighbors. A JQ with a higher core ID is considered better than a JQ with a lower core ID. Eventually, each connected component has only one core. If a sender becomes active for a group before other senders, then it becomes the core of the group. If several senders become active concurrently, then the one with the highest ID is elected the core of the group.

A core election is also held if the network is partitioned. The election is held in the connected component of the partition that does not have the old core. A node detects a partition if it does not receive a fresh JQ from the core for three consecutive join query intervals. Once a sender detects a partition and it has data to send, it promotes itself to the rank of core and participates in the core election. JQs from nodes with lower IDs are not discarded and the routing information regarding those senders is not destructed. Instead, JQs are just demoted to $JQnC$ and they are forwarded using the k-restricted scheme and become susceptible of being stopped by the DIMAS algorithm. This scheme contrasts with the destructive schemes used

in the past core-based multicast routing protocols (e.g. PUMA) in which the partial routing structure built by the dissemination of JQs of senders that contended and loose an election is eliminated when JQ from senders with larger IDs are received.

The way in which two partitions are merged depends on the type of join queries that traverse from one partition to the other. If a JQ reaches a new partition with a "better" core, then it is demoted to a $JQnC$ and disseminated accordingly in that region of the network. The node or nodes that received the JQ from the core with a smaller ID check if they have recently forwarded a JQ for the current core (for instance, within the last 100 mS). If not, then they send JQs that merge the partitions. While traversing the region of the network with the smaller core, JQs force nodes to change to the new core but do not destroy their current routing information regarding the previously known sources. When the JQ is received by the senders located in the previously different partition with the smaller core, they generate new $JQnC$s with larger sequence numbers if at least one-third of the join query period has elapsed since the last transmission of a join query.

For the case of non-core join queries we have the following options. If a $JQnC$ reaches a previously different partition with a better core, then the behavior is analogous to the one just described with the only difference that there is no need of demoting the message because it is already a $JQnC$. On the other hand, when a $JQnC$ arrives to a previously different partition with a smaller core, nodes that first receive the message aggregate the core stated at the arriving $JQnC$ with the sender that originated it, and relay the $JQnC$ but now stating as a core the core with the smaller id. Nodes located at the border of the region of the network with the core with smaller ID are allowed to perform aggregation because (1) nodes in that region have not received a JQ from the core with larger ID (otherwise that sender would be the core), and (2) it is certain that their links are cut links between the core with the larger ID and the receivers that may be located at the region with the core with smaller ID. If the two regions remain connected long enough, then the next JQ generated by the core with larger ID will force all nodes in the network to have a single core.

2.3.4. *Forwarding multicast data packets*

When a source has data to send, it first checks whether it has received at least one JR with the same sequence number as the last transmitted JQ or $JQnC$. If it is the case, the source considers the node from which it received the JR a child and transmits the data packet. If the source does not have any child, then it checks if has elapsed $ALLOW\_NEXT\_JQ$ time since the last time it sent either a JQ or a $JQnC$. If so, it piggybacks the data packet in a JQ (or $JQnC$) with a newer sequence number and transmits it. Otherwise, the packet is silently dropped.

A multicast data packet received from a sender s_i is discarded by a node if a hit is found in the packet cache at the node based on the packet's sender and

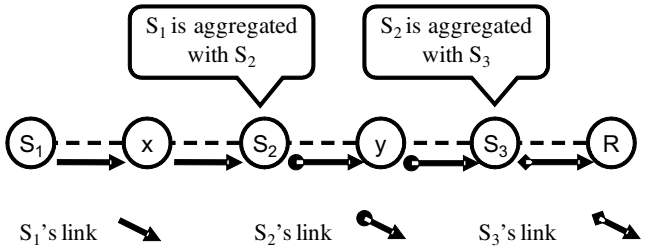

Fig. 2.3. Aggregated path. The $S_1 \rightsquigarrow R$ path is an aggregated path composed of the $S_1 \rightsquigarrow S_2$, $S_2 \rightsquigarrow S_3$ and $S_3 \rightsquigarrow R$ sub-paths.

its sequence number. Otherwise, the receiving node inserts the (*sender's address, sequence number*) pair in its packet cache and determines the *address of the effective source* (*es*) of the packet, which is the one used to decide whether the packet has to be forwarded or not. The address of the effective source for a given packet is the original source s_i, or s_j if s_i was aggregated with s_j at the current node. Once the node determines the value of es, it forwards the data packet if it has at least one child for es. The effective source is obtained from the aggregation map maintained by each node.

Figure 2.3 shows a simple example of a path composed of three concatenated paths, each of which corresponds to a path established for an aggregated source. In the figure, data packets generated by S_1 are routed by x using S_1 as the address of the effective source (es). When a S_1's packet reaches S_2, it determines from its aggregation map that the effective source for S_1 is itself (S_2) and forwards the packet accordingly using its routing table. Nodes along the subpath from S_2 to S_3 similarly determine that S_2 is the effective source for S_1. At node S_3, the effective source becomes S_3 itself, and the same is true for the relays in the subpath from S_3 to R.

If aggregation maps are not communicated among neighbors as part of $JQnCs$ and are maintained only locally, the original multicast data packet can be encapsulated in another multicast data packet with the aggregating source as the sender and the same group as the destination. At each hop, a relay decapsulates and encapsulates the packet before forwarding it.

2.4. PUMA

As well as Hydra, 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. And sources need not join a multicast group in order to send data packets to the group. Like CAMP and MAODV, PUMA uses a receiver-initiated approach in which receivers join a multicast group using the address of a special node (core in CAMP or group leader in MAODV), 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.

Every receiver can be connected to the elected core along all shortest paths between the receiver and the core. Nodes on shortest paths between any receiver and the core are candidate mesh members. The actual number of shortest paths used to establish the mesh depends on the number of next hops to the core that are forced to join the mesh. A sender sends a data packet to the group along any of the shortest paths between the sender and the core. When the data packet reaches a mesh member, it is flooded within the mesh. Nodes maintain a packet ID cache to drop duplicate data packets.

PUMA uses a single control message for all its functions, the multicast announcement. Each multicast announcement specifies a sequence number, the address of the group (group ID), the address of the core (core ID), the distance to the core, a mesh member flag that is set when the sending node belongs to the mesh, and a parent that states the preferred neighbor to reach the core. With the information contained in such announcements, nodes elect cores, determine the routes for sources outside a multicast group to unicast multicast data packets towards the group, notify others about joining or leaving the mesh of a group, and maintain the mesh of the group.

2.4.1. *Connectivity lists and propagation of multicast announcements: mesh establishment and maintenance*

A node that believes itself to be the core of a group transmits multicast announcements periodically for that group. As the multicast announcement travels through the network, it establishes a data structure known as connectivity list at every node in the network. A node stores the data from all the multicast announcements it receives from its neighbors in the connectivity list. Fresher multicast announcements from a neighbor (i.e. one with a higher sequence number) overwrite entries with lower sequence numbers for the same group. Each entry in the connectivity list, in addition to storing the multicast announcement, also stores the time when it was received, and the neighbor from which it was received. The node then generates its own multicast announcement based on the best entry in the connectivity list: for the same core ID, only multicast announcements with the highest sequence number are considered valid. For the same core ID and sequence number, multicast announcements with smaller distances to the core are considered better. When all those fields are the same, the multicast announcement that arrived earlier is considered better. The distance to the core of the current node is one plus the distance to core reported in the best multicast announcement. The parent field is filled with the address of the neighbor from which the best multicast announcement was received. The mesh membership flag indicates if the current node is a mesh member. In general a node

is a mesh member if it has recently heard about at least one neighbor with a larger distance to the core and that is either a receiver or a mesh member.

After receiving a multicast announcement with a fresh sequence number, nodes wait for a short period (e.g. 100 ms) to collect multicast announcements from multiple neighbors before generating their own multicast announcement. When multiple groups exist, nodes group all the fresh multicast announcements they receive into a single control packet, and broadcast them periodically every multicast announcement interval. However, multicast announcements representing groups being heard for the first time, resulting in a new core, or resulting in changes in mesh membership status are forwarded with a much smaller delay. This is to avoid large delays in critical operations, like core elections and mesh establishment which could lead to a delay in establishing the correct mesh, and could lead to packet drops as well as unnecessary transmissions of data packets. However, announcements of this type are not sent immediately, this, to avoid oscillations in the mesh establishment process and to allow opportunistic grouping of two or more announcements in a single control packet.

A node also generate new multicast announcement when it detects a change in its mesh member status. This could occur when a node detects a mesh child for the first time, or when a node that previously had a mesh child detects that it has no mesh children. This way, only nodes lying in shortest paths from receivers to the core are forced to become mesh members.

2.4.2. *Core election*

When a receiver needs to join a multicast group, and similar to Hydra, it first determines whether it has received a multicast announcement for that group. If the node has, it adopts the core specified in the announcement it has received, and it starts transmitting multicast announcements that specify the same core for the group. Otherwise it considers itself the core of the group and starts transmitting multicast announcements periodically to its neighbors stating itself as the core of the group and a 0 distance to itself. Nodes propagate multicast announcements based on the best multicast announcements they receive from their neighbors. A multicast announcement with higher core ID is considered better than a multicast announcement with a lower core ID. Eventually, each connected component has only one core. If one receiver joins the group before other receivers, then it becomes the core of the group. If several receivers join the group concurrently, then the one with the highest ID becomes the core of the group.

A core election is also held if the network is partitioned. The election is held in the partition which does not have the old core. A node detects a partition if it does not receive a fresh core announcement for $3 \times$ *multicast announcement interval*. Once a receiver detects a partition, it behaves in exactly the same way it would upon joining the group, and participates in the core election.

2.4.3. *Forwarding multicast data packets*

The parent field of the connectivity list entry for a particular neighbor corresponds to the node from which the neighbor received its best multicast announcement. This field allows nodes that are non-members to forward multicast packets towards the mesh of a group. A node forwards a multicast data packet it receives from its neighbor if the parent for the neighbor is the node itself. Hence, with no need of packet encapsulation, multicast data packets move hop-by-hop, until they reach mesh members. The packets are then flooded within the mesh, and group members use a packet ID cache to detect and discard packet duplicates.

2.5. Performance Results

In this section we present simulation results in which we compare PUMA and three different variants of Hydra against ODMRP. The three variants of Hydra consist of using the Total-Aggregation algorithm (Hydra-TA), the Core-Aggregation Algorithm (Hydra-CA) or no aggregation (Hydra-NA). We chose ODMRP for our comparison because it is a representative of the state of the art in multicast protocols for MANETs, and because it constructs its forwarding mesh as the union of the source-specific trees of the senders of a particular group. Given that Hydra also builds a structure that is close to a set of source-specific trees or meshes, comparing Hydra against ODMRP allows us to highlight the benefits obtained by the signaling used in Hydra. On the other hand, the results obtained by PUMA allow us to identify the tradeoffs of further reducing the control overhead but at the expense of increased data overhead caused by having meshes that are not composed of shortest paths from senders to receivers.

We use packet delivery ratio, average end-to-end delay and average number of packets relayed per packet received at receivers as our performance metrics. The latter metric can be seen as the cost in terms of bandwidth that the protocol pays to achieve a given delivery ratio.

The multicast protocols are tested with IEEE 802.11 and TDMA as the underlying MAC protocols. The former is the most commonly used MAC in the literature of multicast protocols for MANETs and the latter allows us to isolate the multicast signaling and construction of routing structures from the effects of collisions at the MAC layer. We used the discrete event simulator Qualnet[11] version 3.9. The software distribution of Qualnet itself has the ODMRP code, which was used for the ODMRP simulations. Each simulation was run for ten different seed values. To have meaningful comparisons, all the protocols use the same join query intervals (or multicast announcement interval in the case of PUMA) of 3 seconds for 802.11 and 60 seconds for TDMA. For ODMRP, the forwarding group timeout was set to three times the value of the join query interval, as advised by its designers. Figure 2.4 lists the details of the simulation environment. For Hydra, we set the value of k of

Simulation Environment					
Total Nodes	50	Node Placement	Random	Data Source	MCBR
Transmission Power	15 dbm	Mobility Model	Random Waypoint	Pkts. sent per src.	1000
Channel Capacity	2000000 bps	Simulation Area	1400x1400m		
MAC Protocol 802.11					
Simulation Time	150s	Pause Time	10s	Min-Max Vel.	1-20m/s
MAC Protocol TDMA (1 slot per node in round-robin)					
Simulation Time	30min	Pause Time	50s	Min-Max Vel.	1-2m/s

Fig. 2.4. Simulation environment.

the *k-restricted* region to 1. Hence, only nodes that are at most one hop away from the routing structure of the core disseminate $JQnC$s. For PUMA, each node selects two parents (if availabe) to reach the core of the group.

2.5.1. *Results using 802.11 MAC*

We first focus our attention on an experiment in which the number of concurrent active senders changes. Each sender transmits 20 packets of 256 bytes per second and the group is composed of 20 nodes. We observe from Figure 2.5 that for up to 6 concurrent sources, all the versions of Hydra (TA, CA and NA) consistently outperform ODMRP and PUMA, but, starting from 9 sources, PUMA performs

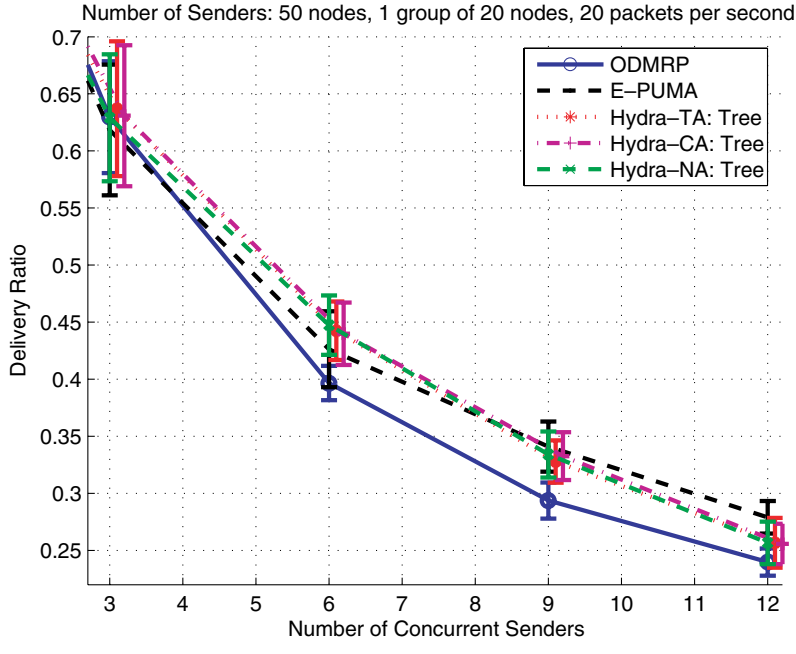

Fig. 2.5. Delivery ratio when varying the number of concurrent active sources with 802.11 MAC.

similar or better than all the other protocols. These results show that the reduced amount of control overhead induced by PUMA allows it to scale better than the other protocols that induce control overhead per source per group. A detailed analysis of the simulations reveals that ODMRP and Hydra reach first the point where the combined overhead induced by control and data traffic does not allow the correct establishment of the routing structures. On the other hand, Hydra performs better than PUMA for up to 6 senders because the routing structures established by Hydra which are close to a set of source specific breath-search trees are more efficient than the shared-trees rooted at an arbitrary receiver built by PUMA.

Among the Hydra variants we can observe that the versions that use aggregation are capable of attaining delivery ratios equivalent to the ones attained by the versions that do not use aggregation, but incurring less control overhead. For these experiments, the tree and mesh versions of Hydra-TA transmitted an average of 77.89% and 78.73% of the *Join Queries* ($JQ + JQnC$) transmitted by the tree and mesh versions of Hydra-NA, respectively.

Figure 2.6 shows the average number of data packets relayed per packet received by receivers. We can observe that in general, PUMA employs more transmissions to cover the receivers of the group. This is due to the fact that the PUMA's mesh-establishment process does not take into account the location of the sources. Among

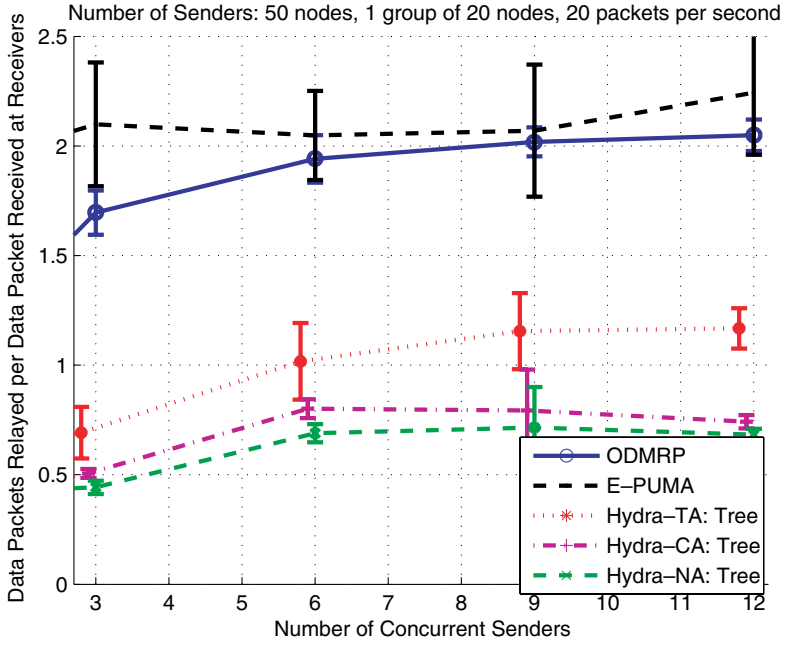

Fig. 2.6. Data packets relayed per data packet received at receivers when varying the number of concurrent active sources with 802.11 MAC.

the protocols that use source-rooted trees, we can see that ODMRP incurs considerably more redundancy than the Hydra variants. As we would expect, the variant of Hydra that uses less redundancy is the tree version with no aggregation, while the one that uses more redundancy is the mesh version with total aggregation. In general, the versions that use total aggregation generate more redundancy than the core aggregation versions, which in turn generate more redundancy than the versions that do not use aggregation. This is due to the fact that the no-aggregation versions try to establish source-specific shortest-path trees or meshes, while the structures created by the aggregated versions are not necessarily composed of shortest paths. If we analyze these two metrics, we can notice how the versions of Hydra are able to attain higher delivery ratios at a much smaller cost than ODMRP. This is a strong indication that the routing structures built by Hydra are more efficient than the ODMRP's mesh that is composed of the union of the source-rooted trees of the sources of the group. We can also observe that even when PUMA has more data overhead its reduced control overhead pays off as the number of senders increases.

Figure 2.7 shows the average end-to-end delay attained by the protocols. We observe a situation similar to the delivery ratio, namely, that the versions of Hydra perform similar or better than the other protocols for up to 6 sources and that PUMA attains the lowest end-to-end delay for more than 6 sources. There are two important factors behind this behavior. The first one is the queueing delay, which

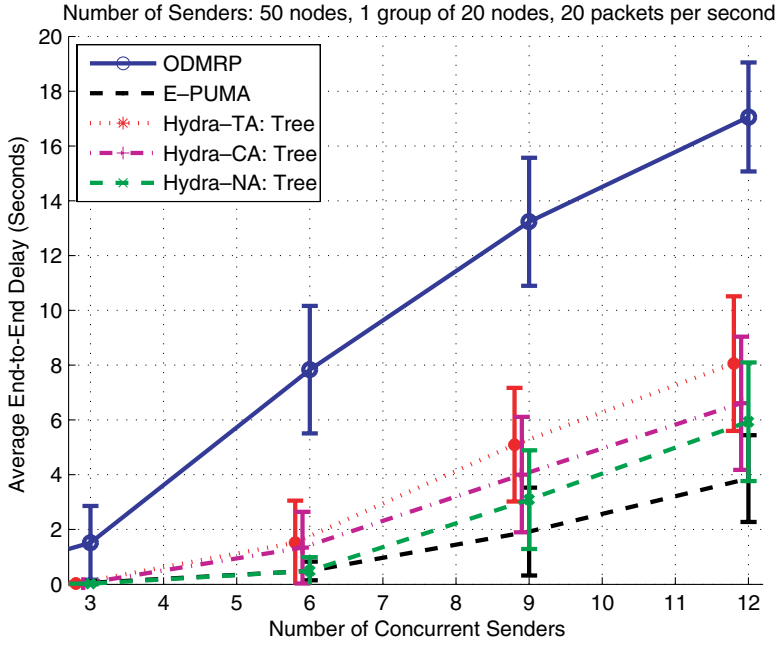

Fig. 2.7. Averge end-to-end delay when varying the number of concurrent active sources with 802.11 MAC.

is strongly related to the amount of control overhead and redundancy used by the protocol. Here is also important to point out that in the context of IP, the queue used for control traffic has higher priority than the data queue, hence the control overhead tends to have a considerable effect in the end-to-end delay. Therefore, protocols that incur less control overhead and redundancy when forwarding data packets tend to attain lower end-to-end delays than the ones with increased level of control overhead or redundancy. The other factor is the length of the paths followed by the data packets.

Figure 2.8 summarizes a number of simulation results for three concurrent sources in different scenarios: node mobility, node density and group size. The results clearly show that Hydra and PUMA render similar delivery ratios and with close to an order of magnitude improvement in delay. Hydra also has the advantage of incurring far less data overhead (about half).

2.5.2. *Results using TDMA MAC*

The results shown in Figures 2.9–2.11 present a picture of the behavior of the protocols when the transmission of control information (by assigning control packets to a higher priority queue) is isolated from the effects of the data traffic. For these experiments, each sender transmits 1 packet of 256 bytes per second and the group is composed of 20 nodes. TDMA allocates 1 slot of 10 mS per node in round-robin fashion by address. For one sender, all the protocols have equivalent performance in terms of delivery ratio and end-to-end delay, as should be expected. As in the case of 802.11 MAC, PUMA incurs more transmission overhead because its mesh is not rooted at the sender. However, as the number of sources increases, the performance of ODMRP drops sharply, which is due to the per source per group flooding strategy it uses. For six or more sources, the Hydra family and PUMA provide an improvement of around 30% in delivery ratio and up to 40% in end-to-end delay. The decreased amount of overhead shown by ODMRP for 12 sources is due to the fact that much more data packets are being dropped early. For 12 sources, the reduced control overhead of PUMA allows it to scale better than the Hydra family showing an improvement of around 5% in delivery ratio and a more considerable improvement of close to 30% in end-to-end delay. The reason of the increased improvement in delay is that PUMA generates much less high-priority control traffic that is sent before the low-priority data traffic.

2.6. Conclusions

We described Hydra and an improved version of PUMA which are two mesh-based multicast routing protocols for MANETs. Hydra is the first multicast routing protocol that establishes routing structures that approximate those built with sender-initiated approaches, but incurring the communication overhead of a

Mobility	0S		5S		10S		15S		20S	
(pause time)	Avg	SD	Avg	SD	Avg	SD	Avg	SD	Avg	SD
	Delivery Ratio									
ODMRP	0.632	0.037	0.639	0.055	0.630	0.049	0.629	0.045	0.630	0.050
PUMA	0.635	0.066	0.619	0.063	0.618	0.057	0.62	0.055	0.62	0.057
Hydra-TA	0.656	0.057	0.646	0.048	0.644	0.057	**0.646**	0.056	0.635	0.054
Hydra-CA	0.649	0.053	**0.651**	0.060	0.637	0.049	0.641	0.057	**0.643**	0.059
Hydra-NA	**0.657**	0.046	0.649	0.057	**0.653**	0.057	0.645	0.060	**0.643**	0.059
	Number of Data Packets Relayed per Data Packet Received at Receivers									
ODMRP	1.714	0.125	1.699	0.090	1.696	0.101	1.727	0.113	1.695	0.120
PUMA	1.918	0.229	1.976	0.251	2.099	0.282	1.999	0.255	2.064	0.276
Hydra-TA	0.986	0.140	1.064	0.171	1.020	0.146	1.001	0.184	1.034	0.162
Hydra-CA	1.005	0.182	0.948	0.113	0.958	0.158	0.940	0.176	0.923	0.167
Hydra-NA	**0.864**	0.155	**0.866**	0.122	**0.822**	0.084	**0.889**	0.146	**0.848**	0.085
	End-to-End Delay (S)									
ODMRP	1.691	1.374	1.570	1.308	1.513	1.349	1.594	1.490	1.291	1.232
PUMA	**0.038**	0.021	**0.046**	0.029	0.056	0.053	**0.063**	0.084	0.070	0.077
Hydra-TA	0.166	0.360	0.180	0.288	0.101	0.164	0.172	0.427	0.134	0.247
Hydra-CA	0.179	0.311	0.049	0.048	0.107	0.174	0.078	0.146	0.068	0.115
Hydra-NA	0.092	0.143	0.076	0.107	**0.026**	0.004	0.082	0.114	**0.036**	0.022
Terrain	1200x1200m		1300x1300m		1400x1400m		1500x1500m		1600x1600m	
Dimensions	Delivery Ratio									
ODMRP	0.659	0.029	0.646	0.033	0.630	0.049	**0.619**	0.049	**0.597**	0.070
PUMA	0.669	0.053	0.648	0.063	0.618	0.057	0.594	0.059	0.564	0.065
Hydra-TA	0.685	0.048	0.670	0.052	0.644	0.057	0.618	0.057	0.586	0.062
Hydra-CA	0.703	0.052	0.672	0.049	0.637	0.049	0.618	0.062	0.587	0.059
Hydra-NA	**0.709**	0.053	**0.680**	0.052	**0.653**	0.057	0.617	0.064	0.589	0.062
	Number of Data Packets Relayed per Data Packet Received at Receivers									
ODMRP	1.660	0.119	1.699	0.128	1.696	0.101	1.663	0.137	1.690	0.093
PUMA	1.771	0.342	1.763	0.359	2.099	0.282	2.149	0.165	2.172	0.146
Hydra-TA	1.050	0.258	1.017	0.194	1.020	0.146	0.994	0.132	1.092	0.082
Hydra-CA	0.888	0.225	0.967	0.246	0.958	0.158	0.982	0.096	1.010	0.112
Hydra-NA	**0.802**	0.231	**0.846**	0.187	**0.822**	0.084	**0.851**	0.087	**0.917**	0.094
	End-to-End Delay (S)									
ODMRP	2.875	1.531	2.098	1.575	1.513	1.349	0.854	0.972	0.464	0.602
PUMA	**0.123**	0.163	**0.039**	0.022	**0.056**	0.053	0.064	0.084	0.036	0.014
Hydra-TA	0.667	0.765	0.267	0.382	0.101	0.164	0.058	0.071	0.038	0.016
Hydra-CA	0.384	0.598	0.313	0.542	0.107	0.174	0.033	0.009	0.031	0.007
Hydra-NA	0.345	0.645	0.175	0.304	0.026	0.004	**0.027**	0.005	**0.027**	0.004
Group size	10		15		20		25		30	
(receivers)	Delivery Ratio									
ODMRP	**0.656**	0.062	0.636	0.052	0.630	0.049	0.625	0.034	0.620	0.045
PUMA	0.589	0.065	0.61	0.071	0.601	0.062	0.618	0.065	0.612	0.069
Hydra-TA	0.650	0.063	0.651	0.055	0.644	0.057	0.639	0.058	0.634	0.050
Hydra-CA	**0.656**	0.055	**0.654**	0.048	0.637	0.049	**0.649**	0.048	0.645	0.054
Hydra-NA	0.647	0.059	**0.654**	0.055	**0.653**	0.057	**0.649**	0.058	**0.653**	0.061
	Number of Data Packets Relayed per Data Packet Received at Receivers									
ODMRP	2.857	0.248	2.145	0.158	1.696	0.101	1.368	0.095	1.200	0.067
PUMA	2.459	0.854	1.979	0.571	1.756	0.264	1.412	0.205	1.213	0.160
Hydra-TA	1.464	0.215	1.158	0.206	1.020	0.146	0.855	0.182	0.745	0.094
Hydra-CA	1.304	0.207	1.080	0.124	0.958	0.158	0.829	0.115	0.741	0.094
Hydra-NA	**1.250**	0.140	**0.972**	0.122	**0.822**	0.084	**0.734**	0.116	**0.647**	0.126
	End-to-End Delay (S)									
ODMRP	0.723	0.75	1.298	1.274	1.513	1.349	1.53	1.381	1.79	1.411
PUMA	0.031	0.038	0.036	0.042	0.027	0.007	**0.03**	0.018	**0.03**	0.021
Hydra-TA	0.033	0.027	0.086	0.17	0.101	0.164	0.271	0.488	0.286	0.402
Hydra-CA	0.022	0.003	0.034	0.021	0.107	0.174	0.185	0.252	0.219	0.361
Hydra-NA	**0.02**	0.002	**0.02**	0.004	**0.03**	0.004	0.097	0.152	0.14	0.286

Fig. 2.8. Mobility, node density, and group size with 802.11 MAC.

receiver-initiated approach. This is accomplished by limiting the dissemination of control information from non-core sources to small regions of the network, and aggregating routing information by establishing common sub-trees (or sub-meshes) that are shared by two or more senders. Hydra can work in either mesh or tree mode

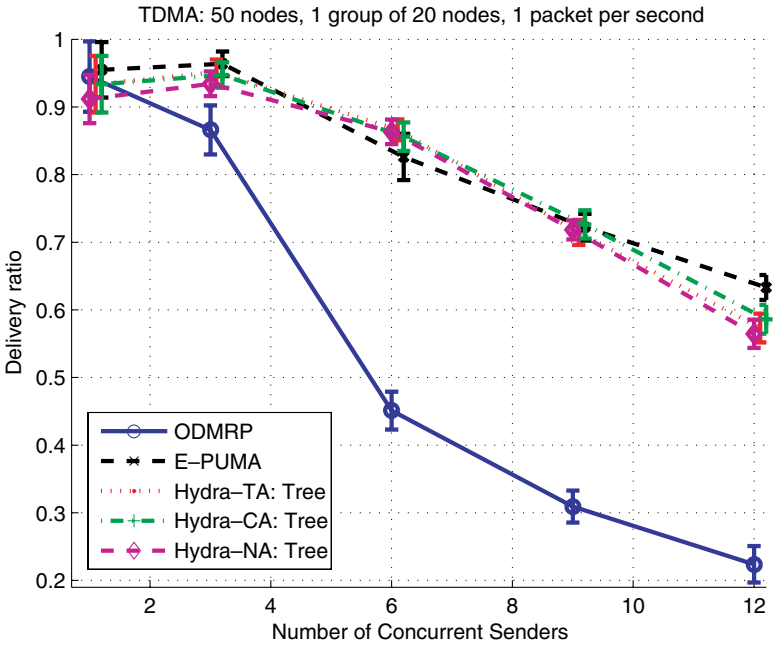

Fig. 2.9. Delivery ratio when varying the number of concurrent active sources with TDMA MAC.

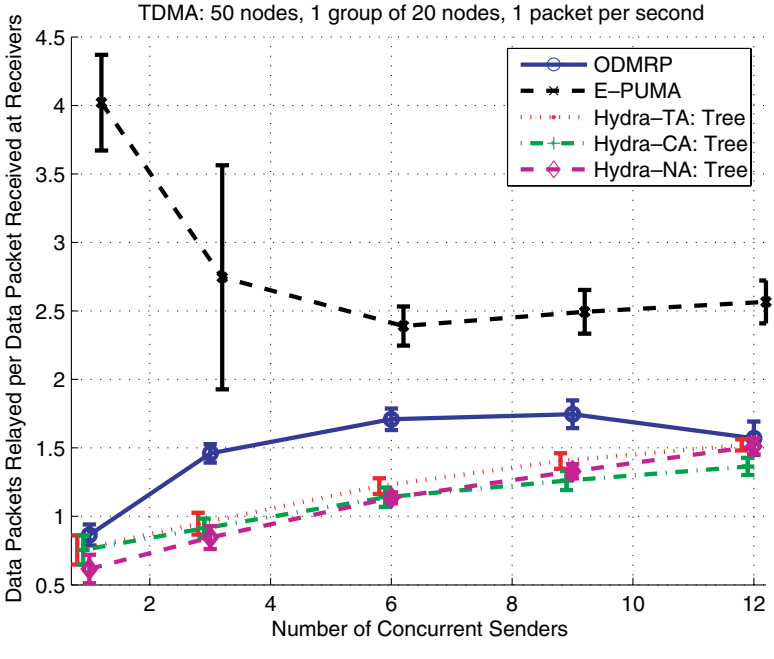

Fig. 2.10. Average number of packets relayed per packet received at receivers when varying the number of concurrent active sources with TDMA MAC.

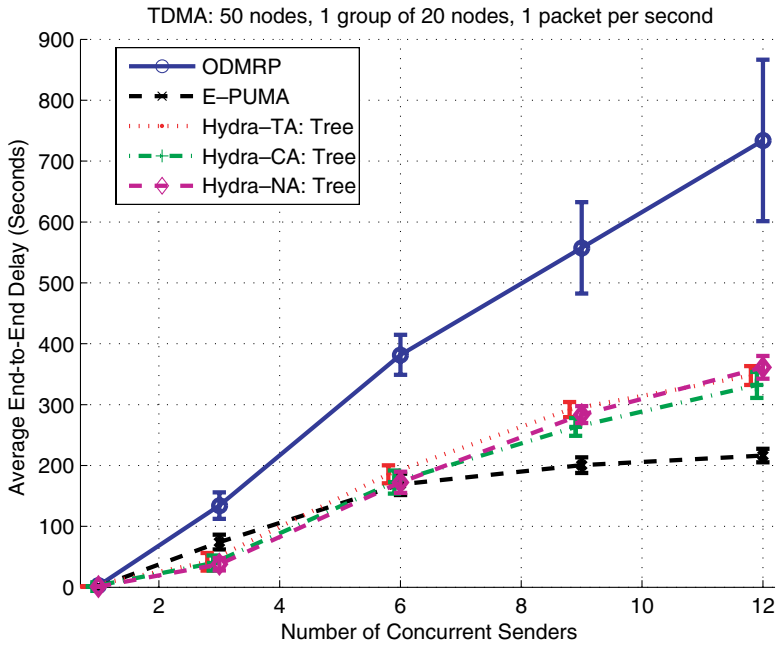

Fig. 2.11. Average end-to-end delay when varying the number of concurrent active sources with TDMA MAC.

by restricting the number of parents that are forced to join to the routing structure. Hydra takes advantage of the work carried out by a core but a core is not necessary for correctness. Cores are elected using a non-destructive core election protocol that does not destroy recently established trees (or meshes). To save bandwidth and take advantage of the broadcast nature of the wireless medium, Hydra opportunistically combines control messages from different groups and sources into a single control packet. PUMA uses a receiver-initiated approach in which receivers join a multicast group using the address of a special node (core), without the need for network-wide dissemination 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. Within a finite time, each router has one or multiple paths to the elected core. All nodes on shortest paths between any receiver and the core collectively form the mesh of the multicast group. 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. The version of PUMA presented here opportunistically combines multicast announcement of different multicast group in a control packet which enables it to react faster than the original PUMA protocol to changes in the topology of the network. We also presented the results of a series of simulation experiments using 802.11 and TDMA MAC protocols illustrating that

Hydra and PUMA attain higher delivery ratios and considerably lower end-to-end delays than ODMRP, and in the case of Hydra, inducing far less data retransmission overhead, even in relatively small networks with few multicast sources. Even though PUMA and Hydra scale better than ODMRP, we are far from having a really scalable multicast protocol for MANETs, hence further research is needed.

Acknowledgments

This work was partially sponsored by the US Army Research Office (ARO) under grants W911NF-04-1-0224 and W911NF-05-1-0246, by the National Science Foundation (NSF) under grant NS-0435522, by the Defense Advanced Research Projects Agency (DARPA) through Air Force Research Laboratory (AFRL) Contract FA8750-07-C-0169, by the Baskin Chair of Computer Engineering, by the CONACyT-UC MEXUS program, and by the Mexican National Polytechnic Institute. The views and conclusions contained in this document are those of the authors and should not be interpreted as representing the official policies, either expressed or implied, of the Defense Advanced Research Projects.

Problems

(1) Explain the mechanism used by ODMRP, PUMA and Hydra to discard duplicate packets. Why is this necessary?
(2) Mention some of the advantages of selecting a core or leader dynamically instead of having it predetermined.
(3) Mention some of the advantages and disadvantages of mesh-based protocols against their tree-based counterparts?
(4) Why are core-based multicast routing protocols more prone to create contention hot-spots than their source-specific counterparts?
(5) Why does the control overhead induced by source-specific multicast routing protocols such as ODMRP grow faster than the overhead induced by core-based protocols such as PUMA?
(6) Describe how the average end-to-end delay attained by multicast routing protocols that run on top of contention-based MAC protocols is related with the control and data overhead they incur.
(7) In the case of ODMRP, PUMA and Hydra, what is the function of the sequence numbers included in the control packets?
(8) Describe some of the differences between the core election protocol implemented in PUMA and the one implemented in Hydra.
(9) Describe the mechanisms implemented in Hydra to reduce the number of control packets used to establish and maintain the multicast mesh.

Bibliography

1. S. K. Das, B. S. B. S. Manoj and C. S. R. Murthy, A dynamic core based multicast routing protocol for ad hoc wireless networks, *MobiHoc '02: Proc. 3rd ACM Int. Symp. Mob. Ad Hoc Net. Comput. (ACM)*, ISBN 1-58113-501-7, pp. 24–35, doi: http://doi.acm.org/10.1145/513800.513804 (2002).
2. V. Devarapalli and D. Sidhu, Mzr: a multicast protocol for mobile ad hoc networks, *Proc. IEEE Int. Conf. Commun. ICC 2001.* **3** (2001) 886–891, vol. 3, doi: 10.1109/ICC.2001.937365.
3. J. J. Garcia-Luna-Aceves and E. L. Madruga, The core-assisted mesh protocol, *IEEE J. Selected Areas Commun.* **17**(8) (1999) 1380–1394.
4. H. Holbrook and B. Cain, Source-specific multicast for ip. internet-draft (2000).
5. J. G. Jetcheva and D. B. Johnson, Adaptive demand-driven multicast routing in multihop wireless ad hoc networks, *MobiHoc '01: Proc. 2nd ACM Int. Symp. Mob. Ad Hoc Net. Comput. (ACM)*, ISBN 1-58113-428-2 (2001) 33–44.
6. L. Ji and M. S. Corson, Differential destination multicast-a manet multicast routing protocol for small groups, *Proc. INFOCOM 2001. Twentieth Ann. Joint Conf. IEEE Comput. Commun. Soc.* **2** (2001) 1192–1201.
7. S. Lee and C. Kim, Neighbor supporting ad hoc multicast routing protocol, *MobiHoc '00: Proc. 1st ACM Int. Symp. Mob. Ad Hoc Net. Comput. (ACM)*, ISBN 0-7803-6534-8 (2000) 37–44.
8. S.-J. Lee, M. Gerla and C.-C. Chiang, On-demand multicast routing protocol, *Proc. IEEE Wireless Commun. Net. Conf. (WCNC, 1999)* **3** (1999) 1298–1302.
9. C. E. Perkins and E. M. Royer, Ad-hoc on-demand distance vector routing, *Proc. Second IEEE Workshop Mob. Comput. Syst. Appl. (WMCSA '99)*, doi: 10.1109/MCSA.1999.749281 (1999), 90–100.
10. E. M. Royer and C. E. Perkins, Multicast operation of the ad-hoc on-demand distance vector routing protocol, *MobiCom '99: Proc. 5th Ann. ACM/IEEE Int. Conf. Mob. Comput. Net. (ACM)*, ISBN 1-58113-142-9, pp. 207–218, doi: http://doi.acm.org/10.1145/313451.313538 (1999).
11. SNT Qualnet 3.9, scalable network technologies: http://www.scalablenetworks.com (2005).
12. R. Vaishampayan and J. J. Garcia-Luna-Aceves, Efficient and robust multicast routing in mobile ad hoc networks, *Proc. IEEE Conf. Mob. Ad Hoc Sensor Syst.,* (2004) 304–313.
13. M. Weiser, The computer of the 21st century, *Sci. Am.* **265**(3) (1991) 66–75.