A Topology-Aware Overlay Multicast Approach for Mobile Ad-Hoc Networks

Mohamed Ali Kaafar, Cyrine Mrabet, and Thierry Turletti

INRIA Sophia Antipolis
2004 route des lucioles, 06902 Sophia Antipolis, France
{mkaafar, cmrabet, turletti}@sophia.inria.fr

Abstract. AOMP (Ad-hoc Overlay Multicast Protocol) is a novel approach for application-layer multicast in ad-hoc networks. We introduce in this paper a new algorithm that exploits a few properties of IP-routing to extract underlying topology information. The basic idea is to match nodes' path to the source in order to detect near neighbors in the physical topology. Then, in a dynamic and decentralized way, we construct a minimum cost mobility-aware delivery tree, connecting nodes that are close to each other. We design a tree improvement algorithm in order to enhance the global performance of AOMP during data distribution. Our simulations results show that, compared to previously proposed application-layer multicast structures, AOMP yields trees with lower cost and traffic redundancy. In addition, it performs well in terms of packet losses, especially in case of node mobility.

1 Introduction

Using mobile and wireless devices is becoming ubiquitous. In recent years, the study and developments of wireless networks have been very popular, leading to flexible and efficient wireless devices. In particular, *Mobile Ad-hoc NETworks* (MANETs) are dynamically reconfigurable wireless networks with no fixed infrastructure, where nodes act as hosts as well as routers. MANETs are deployed in applications such as disaster recovery, distributed collaborative computing, vehicular communication, data and information sharing in difficult terrain, extension of the infrastructure-based networks and video-conferences.

Multicasting provides a mean for group communication by enabling applications to seemingly communicate with a set of nodes. Traditionally a well suited tool for collaborative applications, multicasting is especially useful in ad-hoc networks where tasks may be carried out by groups of nodes. Due to scarcity of bandwidth, varying network connectivity and frequent topology changes caused by node mobility and transient availability, routing algorithms tailored for wired networks will not operate well if directly transposed to MANET. All the more so with multicasting, which adds to the difficulties of unicast routing the complexity of maintaining and handling dynamic multicast group membership changes. Multicast routing protocols have been proposed for MANET. These protocols assume however, that even non member nodes actively participate in maintaining multicast state information and replicating multicast packets. If some non member nodes are fast moving or refusing multicast cooperation, they affect all the involved multicast sessions.

K. Cho and P. Jacquet (Eds.): AINTEC 2006, LNCS 4311, pp. 31–47, 2006.

Application Layer Multicasting has been proposed as the possible solution for this scenario. In this case, multicast group members organize themselves to form an overlay topology; multicast communication between end systems is implemented by forwarding messages through the overlay links over unicast IP. By moving the multicast functionality to the end systems, we solve the problems associated with fast moving intermediate nodes in maintaining multicast state information. But we pay the penalty of increase in end-to-end latency due to duplication of packets flowing over underlying network, and significant bandwidth consumption by proximity measurements overhead or application-level topology adaptation, etc.

Existing studies focus then on the differentiation between application layer multicast protocols and routing (network layer) protocols. This differentiation stands in the traditional Internet, because overlays are built to circumvent the fact that router-assisted approaches are not feasible, and thus message routing is done at the application layer. In contrast, because nodes in MANETs are end hosts as well as routers, all nodes in MANETs are effectively involved in supporting P2P overlay abstractions, and thus P2P overlay abstractions in MANETs have the option of being implemented either at the network layer or above, that is, at the application layer. However, built separately, these two options would cumulate each other disadvantages and would induce overhead, due to inefficient communication between both layers. What is implemented in network layer would be designed and run in a redundant way, by application layer, and vice versa.

In this paper, we propose a new scheme, named AOMP(Ad-hoc Overlay Multicast Protocol), to construct an efficient topology-aware overlay multicast without inducing measurements overhead. In our proposal, while building an overlay multicast structure, we rely first on information provided by the network layer to construct a virtual topology closer to the actual underlying network topology. Adaptation to nodes' mobility is also detected and triggered by the network layer. Routing information needed by AOMP, means "shortest" IP paths that mobile nodes maintain. Actually, "shortest" depends on the particular routing protocol employed, but typically denotes shortest in terms of delay or number of hops. AOMP relies thus on reactive routing protocols, that maintain route paths from a source to a destination.

In a first stage (section 4), we introduce a novel algorithm that connects newcomers to the underlying topology-aware overlay, namely the path matching algorithm. The algorithm is similar to the concept of car pooling. Suppose that your friend is, more or less, on your way home, so giving him/her a ride will not excessively delay you, and you can reduce overall traffic by car pooling. If he/she is out of your way, however, you decide to drive separately. The major strengths of the path matching algorithm are that it exploits path information provided by already-run reactive routing protocols, to construct a topology-aware overlay. It ensures that no specific (costly) route discovery mechanism is deployed, and no end-to-end measurements are exchanged, and thus avoids channel overhead and improves scalability. Moreover, the constructed overlay network has a low delay penalty and avoids "useless" duplication of packets sent on the same link.

In a second stage (section 5), we construct the multicast spanning tree. We propose runtime adaptation mechanisms that allows to enhance efficiency during data

distribution. These are non-greedy link adjustments designed to optimize the overall overlay performance, and not a few particular nodes. Moreover, AOMP adapts to nodes mobility (section 5.4). AOMP does not track nodes mobility at the application level, since this issue is totally handled by the underlying unicast protocol. A mobility adaptation procedure is then triggered by a link status change raised by the routing protocol. This procedure connects nodes to the "next" closest overlay member within a local predefined search scope. It allows to adapt smoothly to frequent and continuous topology changes in ad-hoc networks. The main advantages of our procedure are to maintain virtual-physical topology mapping, while avoiding much overhead resultant from classical 'leave/join' operations used by most of application layer multicast protocols.

Taken into consideration overlay messages control overhead, mobility management and difference that may exist between reactive routing protocols, we evaluated our proposed scheme using extensive simulations, comparing it to previously proposed overlay and network-based multicast protocols. Our findings can be summarized in the following points:

- AOMP outperforms the previous best-performing application layer multicast in terms of delay, packets duplication and reliability.
- AOMP performs favorably even when compared with network layer multicast protocols, more specifically ODMRP [LEE 02]. For ad-hoc groups where 20% to 40% of nodes are part of the multicast group, AOMP exhibits better results in terms of packet delivery ratio.
- AOMP achieves promising delivery ratio in different mobility scenarios, outperforming other proposed protocols.

In the following section, we outline the related literature. We describe the general architecture of AOMP in section 3. The path matching algorithm used in the initial connection of nodes to the AOMP overlay is described in section 4. The second phase of AOMP is presented in section 5. We introduce mechanisms to construct and manage the delivery tree, then we discuss how AOMP adapts to nodes mobility. In section 6, we demonstrate and study the performance of AOMP, through extensive simulations, by providing comparison with several previous approaches. Section 7 concludes the paper.

2 Multicasting in MANETs

In this section, we mention existing ad-hoc multicast protocols and summarize their basic operations. Research efforts can be classified whether they adopt multicasting techniques on the networking layer with multicast routing protocols or on the application layer with overlay multicast schemes.

2.1 Multicast Routing Protocols

As with unicast routing, multicast routing comes in *proactive*, *reactive*, or a combination of the two flavors (*hybrid*). Reactive algorithms represented by MAODV [ROG 99], ADMR [JET 01], OLAM [BAS 00] and ODMRP [LEE 02] present reduced maintenance overhead by maintaining state information only when a multicast session is active. The drawback is decreased responsiveness. Proactive algorithms such as CAMP

[GAR 99] and FGMP [CHI 98] react faster since multicast routing information is readily available, but at the price of introducing high overhead for maintaining multicast group structure even when no multicast session is active. The hybrid approach represented by MZR [DEV 01] aims at obtaining a satisfactory balance among the characteristics of both methods by limiting the scope of the proactive procedures to the local neighborhood of nodes and implementing reactive procedures for longer distances.

State management is one of the most important issues of these multicast protocols. State management involves timely updating of the multicast routing tables at all the nodes (including nodes that are not involved in the multicast session) to maintain the correctness of the multicast routing structure, tree or mesh, according to the current network topology. Even under moderate node mobility and multicast member size, state management incurs considerable amount of control traffic. To address the scalability issues, we need to reduce the protocol states and constrain their distribution, or even use methods that do not need to have protocol state. A number of research efforts have adopted this method, which leads to overlay multicasting.

2.2 Overlay Multicast Protocols

A recent shift towards stateless multicasting is represented by DDM [JI 01], LGT [CHE 02] and RDG [PAT 03]. In overlay multicast, a virtual infrastructure is built to form an overlay network on top of the physical network. Each link in the virtual infrastructure is a unicast tunnel in the physical network. IP layer implements a best-effort unicast datagram service, while the overlay network implements multicast functionalities such as dynamic membership maintenance, packet duplication and multicast routing. All these protocols do not require maintenance of any routing structure at the forwarding nodes. These protocols use different techniques to achieve stateless multicasting. LGT builds an overlay packet delivery tree on top of the underlying unicast routing protocol, using geometric distances between member nodes. Multicast packets are encapsulated in a unicast envelop and unicasted between the group members. When an overlay node receives a data packet from its parent node, it gets the identities of its children from the information included in the header of the packet. For RDG, a probabilistically controlled flooding technique, termed as gossiping, is used to deliver packets to all the group members. In DDM, a source encapsulates a list of destination addresses in the header of each data packet it sends out. When an intermediate node receives the packet, its DDM agent queries the unicast routing protocol about which next-hop node to forward the packet towards each destination in the packet header. DDM is intended for small groups. When group size is large, placing the addresses of all members into the packet headers will not be efficient. The protocol has a caching mode, so that only the difference from the previous states is actually placed in the headers. However, as the forwarding set at the on-route nodes inevitably grow large, each intermediate node needs to keep routes for a large set of destinations. This poses a heavy burden on the supporting unicast protocol even under moderate mobility.

PAST-DM (Progressively Adapted Sub-Tree in Dynamic Mesh) [GUI 03] and ALMA (Application Layer Multicast Algorithm) [GE 04], are two recent overlay multicast approaches. With PAST-DM, each node implements an expanded ring search algorithm [PER 99] to become aware of neighboring member nodes. Nodes periodically

exchange the link-state table with their neighbors in a non flooding manner such that, after several exchanges, a given nodes link state reaches distant nodes. Thus, by looking at each nodes link state, a node can view the entire network. PAST-DM suffers from scalability issues, considering the important overhead caused by control and measurement messages during neighbors discovery and the exchange of link state tables.

ALMA constructs a multicast tree in a decentralized and incremental way. This approach is based on RTT (Round Trip Time) measurements in order to detect and manage nodes' mobility. When periodic RTT measurements towards its parent exceed a threshold, a node has to perform a reconfiguration procedure of its delivery tree. ALMA is also based on an 'expanded ring search' technique limited by a maximum hop count, to detect neighbors. This makes ALMA running over costly positioning systems, that incur considerable amount of control traffic, and thus is more likely to contribute to the overall congestion in the network.

3 AOMP: General Description

In the following, we describe the basic model of AOMP. We have designed a topology-aware overlay multicast architecture to provide a scalable and efficient multicast distribution service to mobile ad-hoc end users. Basically, the AOMP overlay construction is divided into two processes: (1) initial connection to a *backbone Tree*, and (2) delivery tree construction and management.

The initial connection process consists in finding the closest neighbor of each newcomer. It constructs gradually the backbone tree, which is a low cost spanning tree rooted at the source node, and connecting nodes that are topologically close together. The process is based on a *path matching* algorithm that consists in matching the overlay path of a newcomer to the source, with those of other existing overlay members. Each new member of a multicast session first extracts the path (route) from the root (source as a primary sender) of the session to itself. The overlap among this path and other paths from the root is used to partially traverse the overlay data delivery tree, and determine the best parent and children for the new member. We denote the newcomer's parent in the backbone tree, the principal parent. The heuristic used in the path matching algorithm is subject to both capacity or node's fan-out (referring back to the car pooling example, space in your car) and delay (e.g., car pooling will not make the journey excessively long) constraints.

As a spanning tree uses the minimal number of links, additional links can be included in the overlay to improve the delay properties of the low cost backbone tree. The resultant mesh is degree-bounded based on each individual nodes capacity constraint (fan-out). The second process aims thus, in a first step, to derive the delivery tree from the mesh, while respecting the degree constraints of each overlay node. Figure 1 shows a sample AOMP overlay. In the figure, s is the data source and the rest of the nodes are receivers. The dashed lines define the mesh links, and the backbone and delivery tree links are shown as respectively arc and solid lines connecting a parent node to its child.

In a second step, the delivery tree management process aims to periodically refine the delivery tree links. Basically, this is done by adding/deleting links to/from the overlay using a set of local rules running at each node. The rules prioritize the minimization of

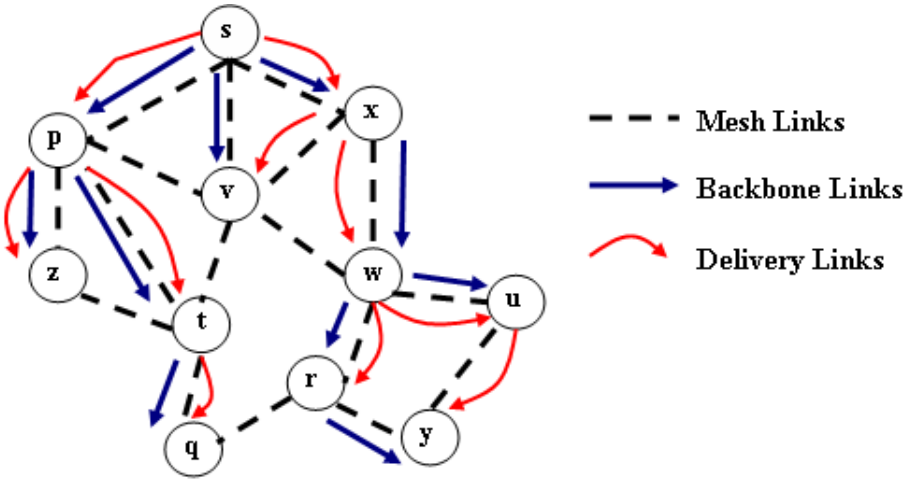


Fig. 1. Example of an overlay tree

each node subtree delay, weighted by the number of its descendants, over a greedy delay minimization of a unique node. Each add/delete link operation involves only the end-points of the link, and requires no global coordination with other nodes in the overlay.

4 AOMP: Connection Process

The AOMP connection protocol is based on the path matching algorithm. Typically, it uses the following heuristic: a newcomer selects, as a parent in the backbone tree, the node whose shortest path from the source has maximal overlap with its own path from the source. This minimizes the increase in number of hops (and hence delay [SIK 04]) over the unicast path, and interestingly decreases the number of duplicated packets on the same physical link.

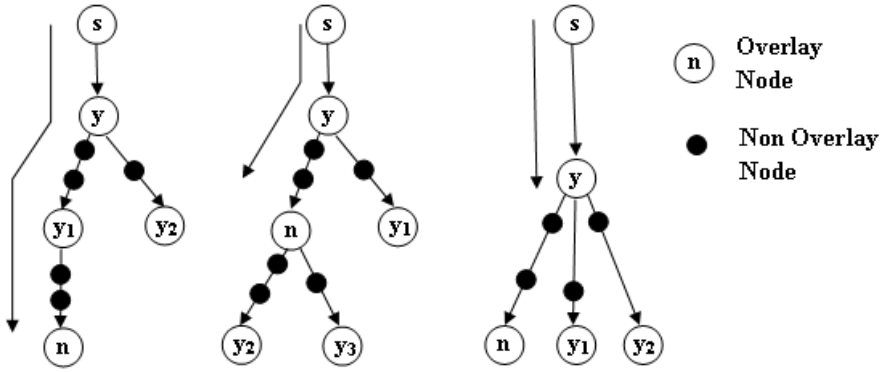
The path matching algorithm is initiated by every newcomer joining the overlay. The algorithm traverses the overlay data delivery tree to determine the best parent for a new node. The main goal is to build a tree connecting the neighboring nodes (proximity in the physical topology), i.e. the *backbone Tree* (see figure 1). In the following, we present terminology and notation to be used throughout this paper. We then describe the path matching algorithm process.

We denote the path between two nodes x and y by $P(x, y)$. It is the sequence of nodes comprising the shortest path from node x to node y according to the reactive underlying routing protocol. $|P(x, y)|$ is the number of hops in $P(x, y)$. A path $P(x, y)$ is a **prefix** of $P(x, y')$ if and only if $P(x, y)$ is included in $P(x, y')$. This property is denoted as: $P(x, y) \hookrightarrow P(x', y')$. In figure 1, $P(s, t)$ is a prefix of $P(s, q)$.

The path matching algorithm is a decentralized process that determines the overlay node, y that shares the longest path prefix (from the source) with the newcomer, x. In other words, the algorithm has to search for y satisfying:

$$(P(s,y) \hookrightarrow P(s,x)) \; and \nexists \; z \; , P(s,z) \hookrightarrow P(s,x), suchthat \; |P(s,z)| > |P(s,y)| \tag{1}$$

Let n be a new member wishing to join a multicast session. n sends then a "Joining_Request" to s. Upon being requested, the source node s extracts its path to the newcomer, and executes the path algorithm. If s estimates itself as the principal parent of n, then the process is terminated and s answers the newcomer accordingly. Otherwise, the request, is forwarded to the child of s of which the path satisfies equation 1. The path of the newcomer to the source is piggybacked in the forwarded request. The algorithm is then processed by the overlay node that has been transmitted the (propagated) "Joining_Request", say y. The algorithm considers three mutually exclusive conditions, as depicted in figure 2.



(a) case 1. (b) case 2. (c) case 3.

Fig. 2. The three cases for the path matching

If possible, node y selects, one of its children y_i, such as $P(y, y_i)$ is the longest prefix of $P(y, n)$. If such a child exists, the algorithm proceeds to traverse the sub-tree rooted by y_i (Case 1 in figure 2(a)). Otherwise, if there are children y_i of y such that the path of n is a prefix of those of y_i for some i, n becomes a child of y with y_i as its children (Case 2 in figure 2(b)). In case no child of y satisfying the first or second conditions exists, n becomes a child of y (Case 3 in figure 2(c)). It is important to note that no more than one child of y satisfying the first condition can exist, since the algorithm searches for the node corresponding for the longest path matching. The "Joining_Request" is then propagated on a unique subtree rooted by a unique child.

Next, we describe the multicast delivery tree management protocol. We specifically discuss the following issues with regards to AOMP:

- Creating and maintaining the multicast tree, i.e. member joins and data distribution.
- Improving efficiency by run-time refinements in mobile scenarios or when members experienced poor performances.
- Ensuring packet level reliability during reconfigurations and mobility.

5 AOMP: Tree Management Process

After the connection process terminates, a newcomer connects to its principal parent in the backbone tree. It is important to notice that at this stage, connection to the backbone tree does not consider any capacity constraints. This allows to define for each newcomer its closest node in the backbone tree, and delay fan-out constraints to its process of connection to the delivery tree. This process starts by constructing a mesh structure from which the delivery tree would be derived.

Maintaining a mesh has several advantages over maintaining only the delivery tree structure. First, a mesh topology consists of multiple paths to the data source, and hence is more robust than a tree structure which can be partitioned even with a single node failure. The multiple paths property is also useful for the overlay optimization. The routing protocol also automatically handles the potential looping problem in distributed tree maintenance[1]. Next, we provide a description of the AOMP mesh structure, and how the newcomer joins the mesh.

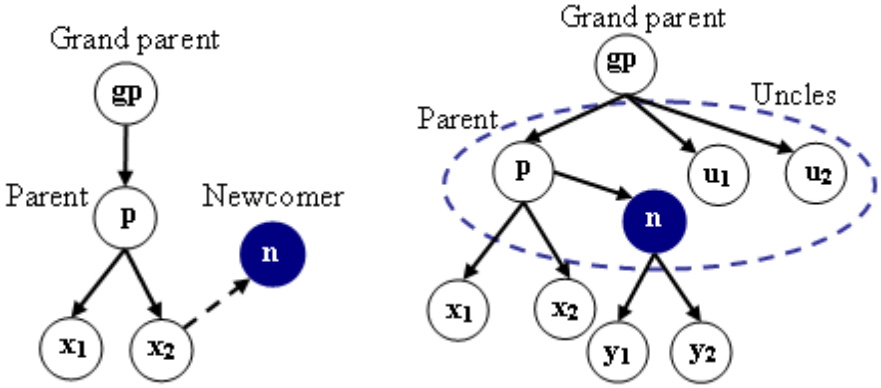
5.1 The Mesh Structure

The backbone tree, constructed during the connection process, constitutes the main skeleton of the mesh. This structure is ever since added with additional links to evolute towards a mesh connecting each newcomer to not only its principal parent (or one of its descendants as described in section 5.2), but also to a few specific nodes that may improve the delay properties of the delivery tree. Adding such links is established as follows: as soon as a newcomer connects to its principal parent, it is informed of the addresses of its grand parent as well as those of its uncles. The newcomer then establishes connections with theses considered nodes, constituting the mesh links. Specifically, two nodes are said to have a neighboring (or peering) relationship when the overlay link between them exists in the constructed mesh. The link may appear in one or both or none of the backbone and delivery trees.

5.2 Joining the Mesh

The connection process terminates when the path matching algorithm is executed by the principal parent of a newcomer. The principal parent sends then a response message to the "Joining_Request" of the newcomer. The response includes position of the newcomer according to the three path matching cases described above. It also contains the principal parent acceptance or not in the delivery tree, as a child. If it is not accepted, the list of the parent's children in the delivery tree is transmitted to the newcomer, "Join_DeliveryTree" messages will be transmitted to each descendant recursively until it can be attached to the delivery tree. Thus, despite the path matching-based connection process that allows the newcomer to detect the closest node in the underlying topology, that one can yield its position in case its principal parent has not the capacity to connect it to the delivery tree (fan-out constraints). Indeed, all the children of a unique principal parent are worth in terms of the metric number of hops. A newcomer will then concede

[1] In [FRA 00], Francis et al. provide a detailed discussion on the looping problem in an overlay tree.



(a) Fan-out constraints prevent full adaptability to underlying topology: non adapted delivery tree.

(b) Local search scope for node n.

Fig. 3. Run-time refinements example

its position in the delivery tree to other overlay nodes that already exist. Figure 3(a) illustrates the case where the newcomer does not connect in the delivery tree to its principal parent, but to the child x_2. The refinement procedure will allow it thereafter to go up in the tree, if ever it acquires a superior weight in the delivery tree, as and when few nodes connect to it or if its performance impose that, as described in the next paragraph.

5.3 Delivery Tree Refinement

We described above how refinements are necessary for nodes that conceded their positions in the delivery tree, due to capacity constraints. The refinement procedure is also important to reorganize the overlay due to the changes in the overlay memberships (when members join, leave or fail) and in the underlying network conditions.

State at an AOMP node. During data transmission, each node is able to maintain its current latency, denoted L_i induced by the overlay routing from the delivery tree source, s. It is computed as the difference between transmission and reception time stamps of data packets. Based on the latency time, each node i estimates its weight w_i in the delivery tree. Typically, a node weight is the sum of latencies of all its descendants to their parents. We notice that a high value of a node's weight implies that the subtree rooted by this node is poorly served in the delivery tree. In other words, this indicates that the subtree members suffer from a significant increase of transmission delay. The weight is computed as:

$$w_i = L_i + \sum_{j \in Children(i)} w_j \tag{2}$$

where w_l, denoting the weight of a leaf node l is equal to its latency L_f. Equation 2 implies that the computation of node's weight is decentralized and recursively transmitted from the children to their respective parents. Moreover, the weight information

is shared between each parent and their children, through "Keep Alive" messages used for the overlay maintenance. In this paper, we will not focus on the mechanisms of maintenance well studied in previous works.

Which node is concerned by the refinement, and when? Refinement procedures are periodically executed by all overlay nodes, except the source. However, the setting of the refinement interval represents a trade-off between the overhead and the accuracy of the delivery tree. To be conservative, by default a node executes a refinement procedure only once every 1200 received data packets (approximately every 5 minutes). However, we use a more aggressive strategy for nodes that concede their position, returned by the connection process, in the delivery tree. Actually, this type of nodes execute the refinement procedure once per 60 seconds, during the 5 first minutes. The refinement procedure is triggered for further reasons, such as drastic change in the network topology, nodes failures, etc. Each node that is concerned by the refinement, estimates its performance while substituting itself to one randomly selected node in its search scope. The latter is composed of the node's parent and uncles, as depicted by figure 3(b).

Refinement decisions. Once a node i selects a node in its local search scope, say j, it computes what would be its potential weight, it connects directly to its grand parent, gp. The potential weight is computed as follows:

$$w_i^{potential} = w_i + (N_i + 1) \cdot [d(gp, i) - d(i, j) - d(j, gp)] \tag{3}$$

where N_i is the number of nodes in the subtree rooted by i. Node i sends then this information to node j, as a "Refinement_Request" message. Upon receiving such request, node j considers henceforth its weight while eliminating the subtree rooted by i, $w'_j = w_j - w_i$. Requested node j computes then its potential weight if ever it yields its position in the delivery tree to the requesting node i:

$$w_j^{potential} = w'_j + (N_j - N_i) \cdot [d(gp, i) + d(i, j) - d(gp, j)] \tag{4}$$

The substitution is processed (and accepted by node j) only if the potential weight of the requesting node i is greater than to the potential weight computed by the requested node j: $w_i^{potential} > w_j^{potential}$.

5.4 Adaptation to Mobility of Nodes

In an ad-hoc environment, it is necessary that the delivery tree adapts to mobility of nodes. A basic adaptation in the AOMP overlay, would result in periodic operations to check if paths to the source have changed or not. An initial connection process would be imposed to the concerned nodes. Nevertheless, such a mechanism may incur high overhead, as well as eventual overload of the source.

We propose a mechanism that allows intermediary overlay nodes (in the path of each node to the source), to take part in the detection of mobility and reconnection to the backbone tree and thus to the delivery tree. First, this would induce less overhead and less sollicitation of the source. Second, the adaptation process would be faster, while relying on the first overlay node that is aware of the topology change due to mobility.

Our mechanism exploits yet another time path information provided by the underlying reactive routing protocol. In fact, we use the route maintenance, automatically (and continuously) processed by the routing protocol, that detects any route changes, caused by nodes mobility. The route to the source change in the routing layer triggers a procedure of adaptation to mobility at the application level. The basic idea is to proceed locally with a connection to the backbone tree in a first step, then reconfigure links of the delivery tree. Let us take the example of the figure 1, and suppose that node w moves towards node v in a way that $P(s, w)$ changes. This change will be noticed at the level of not only the routing layer of w, but also of its parent x, since the latter maintains IP addresses as well as routes to its children, during data transmission. First, x verifies the impact of such movement. Typically, it checks in the new path generated by its child movement, whether it contains at least an overlay node in its neighborhood or not. If no such nodes exist, x reconfigures only its IP path to its child w without imposing any change in overlay structure. Otherwise, connections to the backbone tree are carried out by both w and its children. The request to reconnect to the backbone tree is imposed by x to w, which forwards the request to its children. Reconnection to the backbone differs from the initial connection, in the fact that it is local and is initiated at the first overlay node met in the new path (v in our example for a reconnection of w). Like the initial connection, the reconnection to the backbone applies the path matching algorithm, with v as a source, rather than s. It is finally important to note that the delivery tree is modified only when all connections of w and its children are established in the backbone tree.

6 Performance Evaluation of AOMP

We evaluate the performance of AOMP by carrying out various simulation studies. AOMP is based on both DSR [JOH 03] and AODV [PER 99], as underlying routing protocols, and is denoted respectively AOMP-DSR and AOMP-AODV. For AODV, we extract from the routing tables, the source route accumulation (feature of DSR) to run the path matching algorithm.

We performed simulations to provide quantitative performance analysis according to group members in terms of packet delivery ratio, control overhead as well as average end-to-end delay. We also observe the behavior of AOMP in case of mobility of nodes.

We compared AOMP to both ALMA [GE 04] and PAST-DM [GUI 03] as overlay multicast approaches. For reference purpose, we also compare it to an IP-layer multicast protocol: the On-Demand Multicast Routing Protocol, namely ODMRP [LEE 02]. The detailed parameters on each protocol are described in Table 1.

6.1 Simulation Model and Performance Metrics

The simulation model was built around the NS-2.28 [CAN 04] simulator. Our simulation models a network of 140 mobile nodes placed randomly within a 1000 $\times$ 1000 meters square area. By varying the group size, we vary the percentage of mobile nodes that are involved in the multicast session. The simulation duration is 900 seconds. Each node has a transmission range of 200 meters and channel capacity is 2Mbit/sec. The mobility model follows random waypoint model which has 50 seconds as pause time.

Table 1. Simulation parameters over each protocol

Protocol	Parameter	value
PAST-DM	The period of virtual link exchange	15s
ALMA	Tree reconfiguration period	20s
ODMRP	Interval between join query floods	3s
	Duration of group forwarding state	10s

The minimum speed is 0 m/s and the maximum speed is set to 20 m/s. For the experiments for which no group size is specified a default size of 50 group members is used. A default node's speed of 2 m/s is set as a default mobility parameter. Traffic is generated as constant bit rate (4 packets/second) and packet size is set to 512 kbytes. Nodes fan-out is uniformly distributed in [2..20]. Finally, we use IEEE 802.11 DCF as MAC protocol. The following metrics are studied for comparing protocol performances:

1. **Delivery Tree cost:** The total number of the physical links that make up the logical links in the multicast delivery tree. This metric represents the "goodness" of the structure created by the overlay multicast.
2. **Stress:** The stress of a physical link is the number of identical copies of a multicast packet that needs to traverse the link. This metric quantifies the efficiency of the overlay multicast scheme.
3. **Average Relative Delay Penalty (ARDP):** The relative delay penalty is the relative increase in delay between the source and an overlay member against unicast delay between the source and the same member. ARDP is then the average ratio between the overlay delay (d') and the shortest path delay in the underlying network (d) from s to all other nodes: $\frac{1}{N-1}\sum_{i=1}^{N-1}\frac{d'(s,i)}{d(s,i)}$, where N is the number of nodes in the overlay. This metric is used to quantify the relative cost of routing on the overlay.
4. **Control Overhead:** The number of control packets for delivering per data packet. It includes in AOMP the number of all control packets generated by path matching during the connection to the backbone tree, establishing connections in the delivery tree and refinements. This metric evaluates the cost of the overlay structure according to the overlay goodput.
5. **Data Delivery Ratio:** The ratio of the number of packets actually delivered to the receivers versus the number of data packets that were expected. This metric is used to quantify the reliability of the multicast protocol.

6.2 Performance Analysis

In the following, we detail our simulation results and provide explanations of the observed behavior.

AOMP creates a less expensive delivery tree than PAST-DM and ALMA. Our proposed protocol constructs a multicast delivery tree with a lower cost in terms of physical

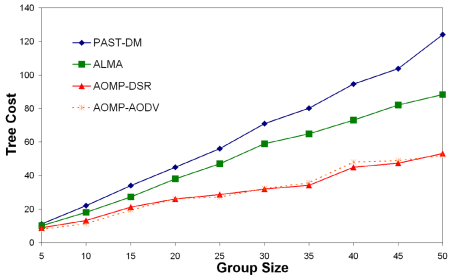


Fig. 4. Tree Cost versus the Group Size

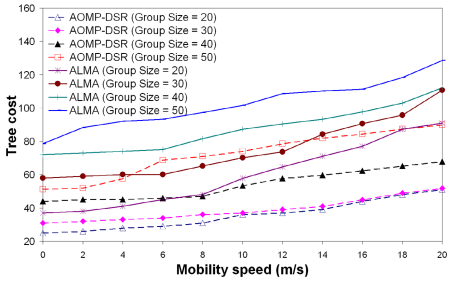


Fig. 5. Tree cost var according to mobility

hop counts than PAST-DM and ALMA. In figure 4 , we plot the tree cost versus the size of the group. We observe that AOMP (AOMP-DSR and AOMP-AODV) achieves delivery trees with an average cost of 42.8 for a group size of 50, i.e. 1.5 to 2.5 less than PAST-DM and ALMA. First, we attribute the difference between PAST-DM and both ALMA and AOMP to the "locally-adaptive" nature of these two protocols. Indeed, PAST-DM creates a logical tree in somewhat centralized way; the decisions of any node transmitting data (considered as a source) affect the creation of the tree globally. In both ALMA and AOMP, the reconfigurations are handled by the receivers, and these local decisions turn out to respond more efficiently to the effects of mobility, and topology changes in case of group membership variation. Second, we observe that AOMP scales better than ALMA. In fact, the ALMA tree cost increases drastically to more than 80 physical links making up the overlay links. This demonstrates that ALMA does not scale to tens of overlay members. AOMP has almost a constant tree cost with a maximum of 51 for AOMP supported by the AODV protocol. Topology information is of paramount importance in this observation, as data packets in AOMP are sent through the shortest path defined by the underlying routing protocol. This fact makes the AOMP overlay structure maps the routing (physical) structure that packets would be guided through anyway. Exploiting this information allow then to build the delivery tree at a minimum cost. In figure 5, we observe the tree cost as a function of mobility speed, for both AOMP-DSR and ALMA. While tree cost is expected to increase with mobility speed, this simulation shows how far a protocol could adapt to mobility. For AOMP, we observe that the tree cost increases "smoothly" while this value drastically reaches high values for ALMA. The ALMA tree cost is 91 for a group size of 20 nodes, under high mobility, which oversteps the tree cost of a 50 nodes group in AOMP. The latter is less affected by mobility and continues to construct less expensive trees, as and when nodes are moving due to its reliance on the routing protocol to detect and extract the new path of the mobile node.

The path matching algorithm of AOMP avoids "useless" packet duplications. Figure 6 shows average physical network stress for each of the overlays, namely PAST-DM, ALMA and AOMP supported by DSR and AODV. The average stress in this simulation is tracked 2 minutes after the last node joined the overlay. The average stress observed in a delivery tree constructed by AOMP is much smaller than those with ALMA and

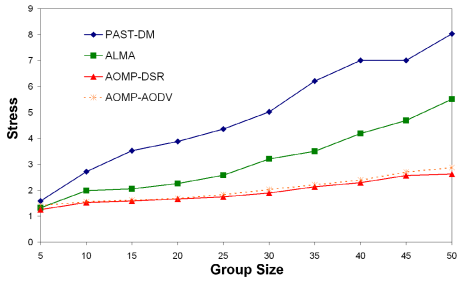
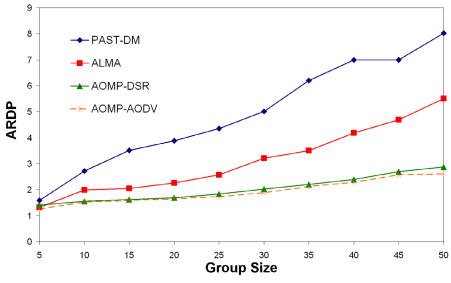


Fig. 6. Average Stress versus the Group Size

PAST-DM. This value stabilizes for AOMP between 2.62 and 2.88 for a group size of 50 members, while it exceeds 5 and 8 for other overlays. Besides the difference that may exist between AOMP-DSR and AOMP-AODV, the results show the efficiency of the path matching algorithm that avoids redundant packets over the same physical links, by binding overlay nodes according to their proximity. Consequently, the possibility of bottlenecks are much lower in AOMP, than in PAST-DM or ALMA. We attribute the slight difference between AOMP-DSR and AOMP-AODV to the fact that while both routing protocols share the on-demand behavior in that they initiate routing activities only in the presence of data packets in need of a route, many of their routing mechanics are very different. In particular, DSR uses source routing, whereas AODV uses a table-driven routing framework and destination sequence numbers. The shortest path extracted by AOMP to be exploited in the path matching algorithm is then different from DSR to AODV. However, the simulation prove that for both cases, AOMP is able to reduce considerably the amount of redundant flows traversing the ad-hoc network, demonstrating the efficiency of the path matching algorithm heuristic.

AOMP achieves a much better $ARDP$ as compared to PAST-DM and ALMA. We characterize the average incurred delay observed by the receivers in a large populated overlay by observing the $ARDP$ variation according to the overlay size in figure 7. In PAST-DM, the $ARDP$ value increases drastically to more than 6 demonstrating that this protocol does not scale to a few number of nodes. We note also that ALMA has lower $ARDP$ than the PAST-DM delivery tree, but suffers relatively poor performance with $ARDP \geq 5.5$ in a 50-nodes overlay. AOMP, for both DSR and AODV as underlying routing protocol, maintains a stable $ARDP$ value while the overlay size is increasing. Thanks to the topology awareness of this protocol, $ARDP$ values are roughly maintained between 1.2 and 2.8.

AOMP incurs low control overhead. We ran simulations to evaluate the control overhead in the overlay and analyse the protocol behavior under dynamic ad-hoc overlay. We assumed a basic header size of 40 bytes per IP-packet and we measured the overall control message traffic sent and received by each node throughout a session. Figure 8 shows the average overhead per node when varying the speed of nodes. Additional messages cost increases with mobility of nodes, particulary with PAST-DM. Recall that



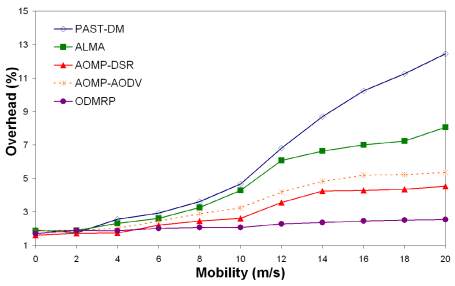


Fig. 7. Average Relative Delay Penalty property versus the Group Size

Fig. 8. Control overhead as function of mobility

PAST-DM creates a logical Steiner tree in a somewhat centralized way; the decisions at the source affect the creation of the tree globally and generate an important cost of control messages. ALMA has a less important additional cost that PAST-DM but more important than AOMP when mobility increases (twice more control overhead messages). The periodic measurements processed by ALMA nodes to know their RTT towards their parent and neighbors make ALMA less efficient than AOMP in terms of control overhead. By relying on the path matching mechanism, AOMP generates lower control trafic. In fact, exploiting path information provided by either DSR or AODV ensures that no costly end-to-end measurements are exchanged and thus alleviates nodes overhead. AOMP-DSR and AOMP-AODV have almost identically shaped curves. However, the absolute overhead required by AOMP-AODV is more important than AOMP-DSR because each of its route discoveries typically propagates to every node in the ad-hoc network. AOMP-DSR sends less overhead, but bigger control packets. In ODMRP, the control overhead remains relatively constant because no updates are triggered by mobility. JOIN QUERY refresh interval was set constant to three seconds and hence no additional overhead is required as mobility increases.

AOMP is reliable. Figure 9 illustrates the packet delivery ratio for different protocols as a function of varying movement speed with static group members. Since ODMRP provides redundant routes with a mesh topology, it shows good performance event in high dynamic situations. On the other hand, AOMP shows similar packets delivery ratio to ODMRP. AOMP is even more reliable in high speed scenarios ($>$ 10m/s), with a ratio slightly decreasing from 0.8% to 0.73%. ALMA and AOMP are very close to each other when nodes are static (lower than 4m/s), but ALMA is much less reliable than AOMP when the speed exceeds 10m/s. ALMA uses the RTT metric to define closeness and to detect topology changes. This may be sufficient when nodes are static, but could lead to many losses in case of high dynamic network. AOMP detects topology changes by exploiting information provided by the routing protocol. It then reacts better. Moreover, if a mobile node moved away from its parent in the delivery tree, and cannot connect anymore to its principal parent, the delivery tree is not modified until a new principal parent has been found.

We vary the number of overlay nodes in figure 10 and observe the data delivery ratio. ODMRP is not affected by the number of multicast members. The data delivery ratio shows slightly better performance (less than 30 overlay nodes) than the case with small group members. As the number of group members increases, more redundant routes may be established, and thus many alternative paths remain available even though the primary path is broken. Similar to ODMRP, the data delivery ratio in AOMP is improved as the number of overlay nodes increases. In particular for groups were 20% to 40% of nodes are part of the multicast session, AOMP exhibits better delivery ratios. In fact, the larger the group size, the greater the probability to detect an overlay node in the path to the source is.

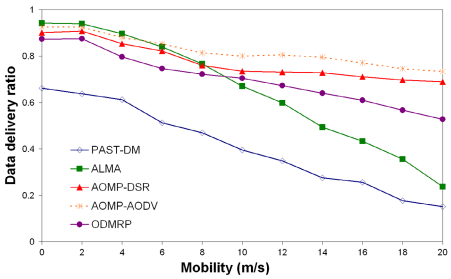


Fig. 9. Packet Delivery Ratio as function of mobility

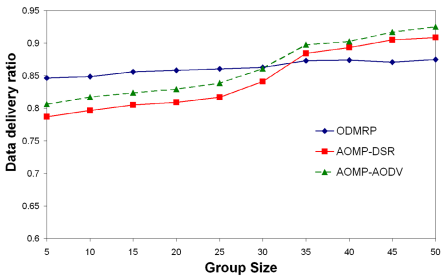


Fig. 10. Packet Delivery Ratio as function of group size

7 Conclusion

In this paper, we proposed a new multicast overlay construction for mobile ad-hoc networks, named AOMP. Based on a path matching algorithm that is underlying routing-aware, the protocol consists in a first step to connect the closest nodes in a backbone tree. An efficient delivery tree is then generated. Run-time refinements are processed during data distribution to adapt to both underlying network and membership changes, and to optimize the overlay performance. The overlay construction process includes also mechanisms to adapt to ad-hoc nodes mobility in a smooth and reliable manner. We carry out simulations to quantify our protocol performance and demonstrate that AOMP outperforms previously proposed overlay schemes. Our main findings prove that exploiting path information at the connection process, allows AOMP to be highly efficient by creating low cost delivery trees and avoiding useless packet duplication without inducing high overhead. Furthermore, AOMP is reliable, achieving promising delivery ratios in case of high mobility scenarios. Our future works consist first in adapting our connection process to different routing protocols, in particular to proactive protocols. The idea is to extract recursively routing information from overlay members tables, exploit it to gradually get a local view of the newcomer and locate it. In a second step, we will focus on designing mechanisms to pro-actively manage nodes mobility in the overlay, using mobility prediction models.

References

[BAS 00] BASAGNI S., ET AL., *On-Demand Location Aware Multicast (OLAM) for Ad Hoc Networks*, Proceedings of IEEE Wireless Communications and Networking Conference (WCNC), Chicago, 2000.

[CAN 04] MCCANNE S., ET AL., *NS network simulator*, http://www.isi.edu/nsnam/ns/, 2004.

[CHE 02] CHEN K., NAHRSTEDT K., *Effective Location - Guided Tree Construction Algorithm for Small Group Multicast in MANET*, Proceedings of IEEE Infocom'02, 2002.

[CHI 98] CHIANG C., GERLA M., ZHANG L., *Forwarding group multicast protocol (FGMP) for multihop mobile wireless networks*, Proceedings of Cluster Computing, 1998.

[DEV 01] DEVARAPALLI V., SIDHU D., *MZR: A multicast protocol for mobile ad hoc networks*, Proceedings of IEEE International Conference on Communications, 2001.

[FRA 00] FRANCIS P., *Yoid Tree Management Protocol (YTMP) Specification*, Technical report, AT&T Center for Internet Research at ICSI (ACIRI), 2000.

[GAR 99] GARCIA-LUNA-ACEVES J.J., ET AL., *The Core-Assisted Mesh Protocol*, Proceedings of IEEE Journal on Selected Areas in Communications, 1999.

[GE 04] GE M., ET AL., *Overlay multicasting for ad hoc networks*, Proceedings of Third Annual Mediterranean Ad Hoc Networking Workshop, 2004.

[GUI 03] GUI C., MOHAPATRA P., *Efficient Overlay Multicast for Mobile ad-hoc Networks*, Proceedings of IEEE WCNC, 2003.

[JET 01] JETCHEVA J., JOHNSON D. B., *Adaptive Demand-Driven Multicast Routing in Multi-Hop Wireless Ad Hoc Networks*, Proceedings of the Second Symposium on Mobile Ad Hoc Networking and Computing, 2001.

[JI 01] JI L., CORSON S., *Differential Destination Multicast–A MANET Multicast Routing Protocol for Small Groups*, Proceedings of IEEE INFOCOM, 2001.

[JOH 03] JOHNSON D.B., ET AL., *The Dynamic Source Routing Protocol for Mobile ad-hoc Networks (DSR)*,draft IETF MQNET 2003.

[LEE99] LEE S.J., GERLA M., CHIANG C.-C., *On Demand Multicast Routing Protocol*, Proceedings of IEEE WCNC99, pp. 1298-1302, September 1999.

[LEE 02] LEE S.J., ET AL., *On-Demand Multicast Routing Protocol in Multihop Wireless Mobile Networks*, ACM/Baltzer Mobile Networks and Applications, Proceedings of Communications in Wireless Mobile Networks, 2002.

[PAT 03] PATRICK J.L., EUGSTER T., *Route driven gossip: Probabilistic reliable multicast in ad hoc networks*, Proceedings of IEEEINFOCOM, 2003.

[PER 99] PERKINS E. AND ROYER E.M., *Ad hoc on-demand distance vector routing*, In Proceedings of the 2nd IEEE Workshop on Mobile Computing Systems and Applications, pages 90100, Feb 1999.

[ROG 99] ROYER E., PERKINS C.E., *Multicast Operations of the Ad-hoc On-Demand Distance Vector Routing Protocol*, Proceedings of ACM/IEEE MOBICOM'99, 1999.

[SIK 04] SIKORA M., ET AL., *On the Optimum Number of Hops in Linear Wireless Networks*, Proceedings of IEEE Information Theory Workshop, San Antonio, 2004.