Overlay, Borůvka-based, Ad-hoc Multicast Protocol: description and performance analysis

Andrea Detti, Nicola Blefari-Melazzi, Claudio Loreti

University of Rome "Tor Vergata", Electronic Engineering Dept., Italy

{andrea.detti, blefari, claudio.loreti }@uniroma2.it

Abstract—**This paper presents a novel MANET multicast protocol, named Overlay Borůvka-based Ad-hoc Multicast Protocol (OBAMP), and evaluates its performance. OBAMP is an overlay protocol: it runs only in the end-systems belonging to the multicast group. User data are distributed over a shared distribution tree formed by a set of non-cyclic UDP tunnels. OBAMP derives the distribution tree by approximating the Borůvka algorithm; the Borůvka algorithm is a classical tool (1926) to find the minimum spanning tree; thus, the distribution tree of OBAMP is an approximation of the minimum spanning tree of the connectivity topology at hand.**

OBAMP shows three distinctive advantages: i) its distribution tree closely resembles the minimum spanning tree; ii) it exploits broadcast communications (with favourable consequences on its efficiency); iii) its design takes into account not only overlay signalling but also network-layer signalling; thus, the protocol succeeds in limiting the overall signalling load, network+overlay. As a consequence, OBAMP has a low-latency and a high delivery ratio, even when the group size increases.

To prove this statement, we analyze the performance of OBAMP with ns-2 and compare it with two state-of-the-art protocols, namely ODMRP (a network-layer protocol) and ALMA (an overlay protocol). Both OBAMP and ALMA are assumed to use AODV as underlying routing protocol.

Also, we stress that we have implemented OBAMP, in Java, and we have tested it on the field, to prove its feasibility.

To allow fellow researchers to reproduce and test our work we published all simulation and implementation codes, in [11].

I. INTRODUCTION

An attractive use of Mobile Ad-hoc NETworks (MANETs) consists in performing cooperative work during occasional team tasks. In such scenarios, multicast traffic plays a more important role, with respect to the one that it would get in the public Internet. As a matter of fact, occasional team tasks are more in need of real time multipoint-to-multipoint services (e.g., push-to-talk, GPS positioning, etc.) than point-to-point ones. MANET multicast protocols can operate as a network layer protocol or as an application layer protocol (known also as overlay protocol). In the first case, all nodes run the protocol and all nodes can be involved in managing a multicast session, including nodes that are not members of the multicast session (e.g., ODMRP [2], MAODV [3], etc.).

On the contrary, an overlay multicast protocol involves only nodes that are members of the multicast session, i.e. the protocol is peer-to-peer (e.g., AMRoute [1], ALMA [7], PAST-DM [8]). User data and signalling information are transferred via transport layer tunnels (e.g. UDP sockets) among member nodes. Therefore, multicast functionality can be embedded within end-user applications, without any multicast support from the underlying network layer. This feature greatly simplifies the deployment of the multicast protocol and justifies the interest of the research community in this topic.

In this paper, we propose an overlay protocol, based on the so-called Borůvka algorithm [5]; we evaluate its performance by means of simulations; we compare it with two promising alternatives, ODMRP [2] (a network-layer protocol) and ALMA [7](an overlay protocol, see Section IV); we implement it to assess its feasibility.

The main aim of OBAMP is to limit the network traffic, both user data and signalling information, so as to achieve a high delivery ratio and a low latency. Also, we would like the protocol to be scalable, i.e., to maintain these properties when the group size increases. These goals are achieved by: i) developing an efficient distribution tree that approximates as much as possible the minimum spanning tree (i.e., the overlay tree with the minimum number of network hops); ii) exploiting radio broadcasting; iii) taking into account and reducing as much as possible not only overlay signalling but also network-layer signalling.

An efficient distribution tree is a quite common objective for designers of overlay protocols. Radio broadcasting is exploited in only one (recent) proposal [8], while, to the best of our knowledge, only OBAMP takes into account also the network-layer signalling to reduce the overall signalling load.

The paper is organized as follows: section II recalls some background on graph theory; section III describes the protocol; section IV focus on performance evaluation; section V presents the conclusions. For space limitations, some details are omitted and can be found in an extended version of this paper available at [12]. This document includes: i) a comparison between the overlay approach and the network-layer approach; ii) the sequence diagrams and messages format of the protocol; iii) a more detailed description of the protocol procedures; iv) a more comprehensive performance evaluation.

Finally, we stress that we have implemented OBAMP, in Java, to prove its feasibility (see implementation codes in [11] and [17]). To the best of our knowledge, this is the first time that an overlay multicast protocol is really deployed in practice. We believe that this is very important, as it often happens that some protocol procedures or mechanisms may seem deceptively easy to implement. Practice brings about unforeseen problems and difficulties.

II. THEORETICAL BACKGROUND

The aim of this section is twofold: a) to recall graph theory results showing which is the cheapest distribution tree in terms of number of hops on a given topology: it comes out that the best tree is the *minimum spanning tree* for the overlay approach, and the *Steiner tree* for the network layer approach; b) to describe the Borůvka algorithm, which is a classical tool to find the minimum spanning tree.

A. Graph theory background

Let us consider a generic multicast session at time t, over a given MANET, without detailing, for the moment, if it is implemented with an overlay or a network-layer approach. We can define:

– $\{N\}$ as the set of network nodes;
– $\{M\}$ as the set of member nodes of the multicast session, i.e., nodes belonging to the multicast group ($M \subseteq N$);
– $G=(V,E)$ as the connection graph, formed by the set $\{V_i\}$ of vertices and the set $\{E_{ij}\}$ of edges, where:
  – the vertex V_i is associated with the i-th network node able to perform multicast routing;
  – the edge E_{ij} is associated with the connection service provided by the underlying layer between vertex V_i and vertex V_j;
– $c(E_{ij})$ as the cost (or weight) of the edge E_{ij} measured in network hops;
– T as the minimum cost tree, i.e. a sub-graph of G that spans all members M and has the minimum cost, measured as the sum of the cost of the involved edges.

We note that the graph G contains all the possible routes that can be used to setup the multicast session in the considered MANET. As a consequence, T is the *minimum* cost tree for the considered MANET.

Now, the graph G and the tree T may vary, depending on if we consider an overlay or a network-layer approach.

In the *overlay case* it turns out that: 1) only member nodes can perform multicast routing; 2) the connection service is supplied by the underlying TCP or UDP layers, which provide all Eij connections among members; these connections are named overlay links. In terms of graph theory these two features translate in: i) the edges of the graph are overlay links (and thus they may consist of more than one network link); ii) $V=M$; iii) G is the fully meshed connection graph connecting all the vertices; iv) the cost $c(Eij)$ is the number of network hops of the overlay link between member i and member j (assuming that the underlying unicast routing protocol minimizes the path length measured in number of hops, in that case the overlay link is also the shortest path between member i and member j); v) T is the sub-graph of G containing all vertices and having the minimum possible cost, namely the *minimum spanning tree*.

In the *network-layer approach* it turns out that: 1) all nodes can perform multicast routing; 2) the connection service is supplied by the underlying Data Link / MAC layers, which provide only connections E_{ij} among adjacent nodes; these connections are named network links. In terms of graph theory these two features translate in: i) the edges of the graph are network links; ii) $V=N$; iii) G is the connection graph connecting only adjacent nodes; iv) $c(Eij)=1$; v) T is the sub-graph of G containing only the vertices of M and having the minimum possible cost, namely the *Steiner tree.*

This said, we can conclude the section with an important consideration. The overlay approach can not exploit non-member nodes for multicast routing; this implies that it is less efficient than the network-layer approach. In particular, it has been found that the ratio between the cost of the minimum spanning tree and that of the Steiner tree is limited to 0.9, in practical ad-hoc network scenarios [6].

B. Borůvka algorithm

The Borůvka algorithm (1926) [5] finds the minimum spanning tree over a given graph. Alternative methods serving the same purpose are the Kruskal algorithm (1956) [18] and the PRIM algorithm (1957) [19]. We selected the Borůvka algorithm because it lends itself more easily to a distributed implementation. It works as follows:

```
1. make a list L of {W} trees, where each
   tree is composed of a single vertex
2. while L has more than one tree
      for each tree in L, find the smallest
      edge connecting the tree to another
      disjoined tree, thus forming a new tree
3. end
```

Let us define *n-level edges* the set of edges of the minimum spanning tree built by the Borůvka algorithm at the n-th iteration (in the while loop). It is easy to see that the first-level edges are the edges that connect nearest vertices. This definition will be useful in the sequel.

III. OBAMP

A. Main protocol features

The goal of OBAMP is to limit the network traffic, both user data and signalling information, so as to achieve a high delivery ratio and a low latency, and to maintain these properties when the group size increases. To this end, OBAMP: i) creates a cheap distribution tree; ii) exploits radio broadcasting; iii) limits the protocol overhead and the "induced" network layer-signalling.

OBAMP is a mesh-first overlay multicast protocol: first it builds an overlay network spanning all members (i.e., a mesh); then it builds the distribution tree by selecting a subset of non-cyclic overlay links belonging to the mesh. Fig. 1 reports an example of mesh and corresponding distribution tree in a 10-members case. Red lines are overlay links of the tree (and of the mesh as well, of course); dashed lines are overlay links of the mesh and not of the tree. Non-member nodes are not drawn, although they participate to the network layer routing.

Both the mesh and the distribution tree are periodically updated, to follow the dynamics of the network links. The procedures that build the mesh and the tree are named *mesh-create* and *tree-create*, respectively.

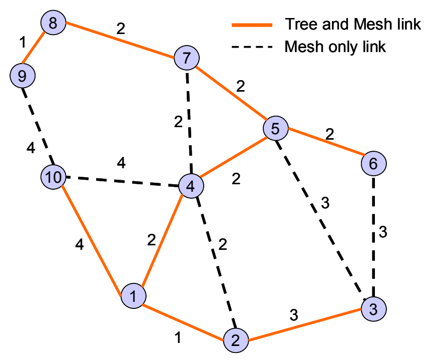

Fig. 1: A mesh with a corresponding tree and related hop distance/cost

1) *Usefulness of the mesh-first approach*

The mesh-first approach was originally proposed by AMRoute [1] and is alternative to the tree-first approach used, for instance, by ALMA [7], which builds directly the multicast tree, without previously forming a mesh.

The mesh-first approach constructs a structure more resilient to overlay link failures; this feature is important in a MANET scenario. As a matter of fact, in a tree-first approach the loss of an overlay link (e.g., due to a member hardware failure) would force a time-consuming process of neighbour discovery to select the "recovery" overlay link. During the discovery process, the tree is partitioned and loss phenomena can occur. On the contrary, in the mesh-first approach, it is often possible to quickly select a recovery overlay link among the mesh links, without the need of discovering available overlay links.

2) *Performance issues driving the design process*

This sub-section discusses some performance issues that explain the rationale that lies behind the definition of OBAMP. Let us define the *efficiency* $\rho(t)$ of a generic distribution tree at time t as the ratio between the cost of the corresponding minimum spanning tree $C_{MST}(t)$ and the cost of the distribution tree $C_{tree}(t)$ itself, i.e. $\rho(t) = C_{MST}(t)/C_{tree}(t)$. As discussed in section II, the best choice for an overlay protocol is the minimum spanning tree and hence $\rho(t) \leq 1$. It is helpful to express $\rho(t)$ as follow:

$$\begin{aligned} \rho(t) &= \frac{C_{MST}(t)}{C_{tree}(t)} = \rho_{mesh}(t) \cdot \rho_{tom}(t) \\ \rho_{mesh}(t) &= \frac{C_{MST}(t)}{C_{meshMST}(t)} \\ \rho_{tom}(t) &= \frac{C_{meshMST}(t)}{C_{tree}(t)} \end{aligned} \tag{1}$$

where:

– $\rho_{mesh}(t)$ is the *mesh efficiency*, defined as the ratio between the cost $C_{MST}(t)$ of the minimum spanning tree and the cost $C_{meshMST}(t)$ of the minimum spanning tree of the given mesh at time t. This parameter measures the ability of the *mesh-create* procedure to include in the mesh overlay links belonging to the minimum spanning tree; as a matter of fact, if all minimum spanning tree links are contained in the mesh then $C_{meshMST}(t) = C_{MST}(t)$ and $\rho_{mesh}(t)$=1. On the other side, the more the mesh is "efficient", in the sense just explained, the more are the opportunities for the *tree-create* procedure to build a cheap tree.

– $\rho_{tom}(t)$ is the *tree-over-the-mesh efficiency* defined as the ratio between $C_{meshMST}(t)$ and the cost $C_{tree}(t)$. This parameter is a measure of the effectiveness of the *tree-create* procedure; in the best case, when ρ_{tom}=1, the distribution tree will be the minimum spanning tree of the mesh.

The overall meaning of Eq. (1) is that the creation of a tree is a two-steps process and thus its efficiency is the product of the efficiency of the component steps. In [12] we report a simulation run showing the behaviour of the costs $C_{tree}(t)$, $C_{meshMST}(t)$, $C_{MST}(t)$, to better highlight this issue.

B. *Protocol procedures*

This section describes the protocol procedures. The sequence diagrams and messages format can be found in [12].

User data are distributed over a distribution tree created by means of protocol procedures. We first describe how user data are distributed and then we introduce the procedures that we need to create the distribution tree.

1) *Data distribution*

Let us define as neighbours two members connected by a mesh link. Data distribution is executed in two different ways, depending on if the hop distance between the forwarding member and its neighbour is greater than one or equal to one:

– in the first case, each member forwards in unicast way the received (or generated) data on all overlay links of the distribution tree to which it is connected, with the exclusion of the receiving one.

– in the second case, data are distributed to members belonging to the mesh by means of only one radio broadcasting transmission, with a value of the IP Time To Live (TTL) equal to 1.

The rationale of the second case is that it may happen that more than one receiving member is within the coverage area of a forwarding member. In that case, it is more efficient to reach receiving members via radio broadcasting, instead that via unicast transmissions over the distribution tree.

Data duplication may occur because of loops in the mesh, loops in the tree, due to the dynamics of the MANET, and to other reasons. To limit such duplication, we resort to two mechanisms: i) each data message contains a field (named `JustForwardedMemberCode`) that codes the list of the members to which those data have already being sent. Members will not forward a message to members contained in this list; ii) data duplication is further limited by means of temporal data caches, as done in [7].

Thanks to these mechanisms, only one transmission is sufficient to reach neighbours within the same radio coverage, independently from the tree topology. This feature maintains good performance even when the group size increases.

This said, we can now focus on how to create the mesh and the distribution tree.

2) *Mesh-create*

From Eq. (1), it is easy to see that the *mesh-create* procedure should return a mesh that contains the greatest possible number of overlay links belonging to the minimum spanning tree.

From this point of view, it is clear that the *full-mesh* will have $\rho_{mesh} = 1$ since it will surely contain all the overlay links of the minimum spanning tree. On the other side, the maintenance of a mesh link requires the exchange of overlay control messages and related, "induced", signalling at the network layer. The overall amount of signalling increases with the square of the multicast group size, M, since a full mesh has $M \cdot (M-1)$ overlay links. As a consequence, maintaining a full mesh is in contrast with the aim of achieving a good scalability performance, and a trade-off is in order.

For this reason, the *mesh-create* procedure builds a "trade-off mesh" that tries to limit the number of overlay links while at the same time tries to include as many as possible minimum spanning tree overlay links. We will see that the mesh created by our procedure will contain the *overlay links that connect nearest members*, plus other overlay links eventually needed to avoid group partition.

If we translate this in the terminology introduced in section II.B, we can say that the *mesh-create* procedure will surely include in the mesh at least the *first-level edges of the minimum spanning tree.* On the other side, the inclusion of the other overlay links of the minimum spanning tree is not guaranteed; this implies a small loss of mesh efficiency. This inefficiency is the price to be paid to curb the signalling overhead, by limiting the number of mesh links.

The *mesh-create* procedure is made up of three elementary sub-procedures: *hello*, *fast-hello* and *link-pruning*. The *hello* and *fast-hello* sub-procedures periodically establish or refresh mesh links; in other words, their aim is to find the neighbours of each member, and to estimate their hop distance. We need to estimate the hop distance for a number of reasons; for instance, data are distributed differently, depending on the hop distance.

The *link-pruning* sub-procedure manages the removal of a mesh link.

The mesh is created (and maintained) in a distributed way: each member uses a number of parameters and state variables. An important data structure maintained by each member is the *neighbours list*, which contains the status information of the mesh links connected to the member. This status information is stored in a record containing 12 fields, fully described in [12]. The most important of these fields are: neighbour IP address; `NEIGHBOUR_CORE_ADDRESS`; `HOP_DISTANCE`; `EXPIRY_TIME`; `TREE_FLAG`: true if mesh link is a tree link.

a) Hello sub-procedure

The aim of the *hello* sub-procedure is to find the neighbours of each member and to evaluate the related hop distance. It is performed by all members when the nearest member is perceived to be no more than one network hop away or when the *neighbours list* is empty; otherwise the *fast-hello* sub-procedure is carried out.

The *hello* sub-procedure exploits a so-called expanding-ring search, and is executed every `HELLO_PERIOD` seconds. To describe it, let us consider a generic member, *S*. Member *S* sends out a sequence of broadcast HELLO messages, contained in IP datagrams with incremental values of the IP Time To Live (TTL) field. The TTL field is incremented up to `MAX_HELLO_TTL` or until member *S* receives a HELLOREPLY message from another member. The IP TTL value set by member *S* is copied in a specific field of the HELLO message.

When a member, *R*, receives an HELLO message, it creates (or refreshes) a mesh link toward member *S*. This means that a *neighbours list* entry is created (or updated) by setting `EXPIRY_TIME` equal to the current time plus 1.5 * `ALLOWED_HELLO_LOSS` * `HELLO_PERIOD` and by setting `HOP_DISTANCE` equal to the IP TTL value contained in the HELLO message. Then, member *R* sends to *S* a unicast HELLOREPLY message, piggybacking the originating IP TTL.

At the reception of the HELLOREPLY, member *S* stops the expanding-ring search and creates (or refreshes) a mesh link toward member *R* by creating (or updating) the relevant *neighbours list* entry, through the same procedure described above for *R*.

At the end of the *hello* sub-procedure, member *S* knows who its neighbours are and has an estimate of how far away each neighbour is. Thus, in principle, we do not necessarily need the *fast-hello* sub-procedure. The raison d'être of this latter procedure is to improve performance.

The problem is that the estimate of the hop distance performed by the *hello* sub-procedure may be wrong, due to the fact that members are moving. This has serious implications in the data forwarding phase: if a member believes that its target is one hop away, it will use broadcast with a value of the IP TTL equal to 1 to reach it. If the target is NOT one hop away, the broadcast transmission will not be able to reach the target, causing data loss.

To alleviate this problem we have to improve the quality of the estimate of the hop distance. To do so, we could simply execute the *hello* sub-procedure more often, so that moving members are better tracked. However, this would increase the signalling overhead, especially when the hop distance is greater than one. Our solution is to devise a new sub-procedure, to be executed more frequently and only when the nearest member is perceived to be one network hop away.

b) Fast-hello sub-procedure

The *fast-hello* sub-procedure is executed only when the nearest member is perceived to be one network hop away. It is repeated with a period equal to `FAST_HELLO_PERIOD`, which is smaller than `HELLO_PERIOD`.

To describe it, let us consider a generic member, *S*. Member *S* sends out a FASTHELLO messages contained in a IP datagram with TTL=1 by means of a single broadcast transmission. When a member, *R*, receives a FASTHELLO message, it creates (or refreshes) a mesh link toward member *S*. This means that a *neighbours list* entry is created (or updated) by setting `EXPIRY_TIME` equal to the current time plus 1.5 * `ALLOWED_HELLO_LOSS` * `FAST_HELLO_PERIOD` and by setting `HOP_DISTANCE` equal to one.

An interesting comment is that the scope of the HELLO messages in both sub-procedures, *hello* and *fast-hello*, is limited to the hop distance between member *S* and its nearest member. Therefore, nearest members will be surely connected by mesh links, as previously stated.

c) Link-pruning sub-procedure

The *link-pruning* sub-procedure removes outdated mesh links by using a soft-state approach. When the EXPIRY_TIME of a given mesh links is reached, then that mesh link is pruned by deleting its entry in the *neighbours list*, unless the mesh link is also a tree link. In the latter case: i) the mesh link is not pruned, to avoid group partition; ii) the related entry in the *neighbours list* is not deleted, but its hop distance is set to MAX_HELLO_TTL + 1, which has the meaning of *unknown* distance.

3) Tree-create procedure

Once that the mesh is built, OBAMP creates a shared distribution tree over the mesh by using the *tree-create* approach proposed by AMRoute [1]. We selected this approach for its capability of avoiding persistent tree loops (even if it does not avoid *temporary* tree loops). In addition, to improve the *tree-over-the-mesh efficiency,* ρ_{tom} (see Eq. 1), we add to the AMRoute mechanism a novel feature, named *handling delay*. The *tree-create* procedure is initiated by a special member, named *core,* which is chosen by means of a suitable procedure, named *core election*, described in the following 5.b.

Each member stores the identifier of its *core* (i.e., its IP address) and accepts only control packet coming from its *core*.

The *core* sends out TREECREATE #*x* messages toward its neighbours periodically, with a period equal to TREE_CREATE_INTERVAL. Each round of such messages refreshes the tree topology; the value of *x* identifies the *x*-th refresh round and is reported in a specific field of the TREECREATE message. Another field of the TREECREATE message indicates who is the nearest neighbour. TREECREATE messages are sent via broadcasting to neighbours at one hop distance, and via unicasting otherwise.

When a member, *R*, receives a TREECREATE message from another member, *F*, it delays the handling of this message for a time equal to HANDLING_DELAY. The member *R* determines the value of HANDLING_DELAY as follows:

```
if  [(F is the nearest member of R) or (R is
     the nearest member of F)]
then HANDLING_DELAY=0
else HANDLING_DELAY =( dist -1)*$U_h$+r(0,1)* $U_h$ /10;
```

where U_h is the HANDLING_DELAY *time unit*; *dist* is the HOP_DISTANCE of the mesh link connecting *F* to *R*; *r*(0,1) is a uniform random value in the (0,1) interval, used to differentiate the delays of paths with the same hop distance.

The first time that *R* handles a TREECREATE #*x* message, it marks the member from which it received the message as *current parent member*; the parent member of the previous refresh round is marked as *old parent member*. The *current parent member* is the closest upstream vertex of the tree toward the *core*.

If there is not a tree link between *R* and its *current parent member*, then *R* creates it by using a two ways handshake setup procedure.

If *old parent member* is different from *current parent member*, then *R* tears down the tree link toward *old parent member* by means of two ways handshake tear-down procedure.

At this time, *R* can forward the TREECREATE #*x* message toward its neighbours, with the exclusion of the receiving one; TREECREATEs #*x* messages that show up after that *R* has handled the first TREECREATE #*x* message are discarded.

a) Two properties of the OBAMP tree

A first property of the tree created by OBAMP is its capability of *avoiding persistent tree loops*.

A second property is that the OBAMP tree contains at least the first-level edges of the minimum spanning tree.

The demonstration of these properties can be found in [12]. Here, we just comment on the second property: the parameter HANDLING_DELAY of the *tree-create* procedure can be seen as a way to enforce a priority level among mesh links: if we assign to a given mesh link a shorter HANDLING_DELAY we are increasing its probability to be selected as tree link. In particular, the *tree-create* procedure assigns to nearest members an HANDLING_DELAY equal to zero. This implies that the mesh links connecting nearest members (i.e., the first-level edges of the minimum spanning tree) are surely included in the tree, at least in the ideal case where the only latency introduced in the network comes from the HANDLING_DELAY. To maintain this property in a real environment, when we have other sources of delay in addition to the HANDLING_DELAY (propagation, queuing, etc.), we have to suitably choose the value of the HANDLING_DELAY *time unit*, U_h. This parameter must be chosen great enough so that the other sources of delay do not alter the order of precedence in choosing tree links established by the HANDLING_DELAY parameter by itself.

4) Member-join and member-leave procedures

When a member wants to join a group it simply elects itself *core* of a mesh formed by itself only; then the *outer-tree-create* procedure (see below) will take care of the joining of such atomic mesh with the rest of the group.

When a member wants to leave the group, it simply switches itself off. The neighbours connected to that member through tree links will perform the *tree-link-recovery* procedure (see below).

5) Other OBAMP procedures

The OBAMP protocol includes other three procedures. Here, we will only highlight their main features. Their full description can be found in [12].

a) Outer-tree-create procedure

Up to now, we have implicitly assumed that the OBAMP mesh spans all members. In reality, the *mesh-create* procedure discovers only nearest members whose distance is at maximum MAX_HELLO_TTL hops. Hence, the *mesh-create* procedure may create partitioned meshes with different *cores* and give rise to different distribution trees, one for each partitioned mesh. To cope with this issue the *outer-tree-create* procedure connects these different meshes and distribution trees by means of a tree link, creating a mesh and related distribution tree that span all members.

b) Core-election procedure

The procedure *core-election* is used to uniquely identify the *core* of the mesh and is used: i) when two meshes, each with its own core must be connected; ii) when a *core* leaves the network or when the network gets radio partitioned and a new *core* must be found.

c) Tree-link-recovery procedure

The *tree-link-recovery* procedure has the aim of re-establishing the connectivity of the distribution tree in case of a failure of a member or when a member goes out of radio coverage.

IV. Performance evaluation

In this section, we analyze the performance of OBAMP with ns-2 and compare it with two state-of-the-art protocols, namely ODMRP [2], [16] and ALMA [7].

ODMRP is a network-layer protocol; as a consequence, the comparison between ODMRP and OBAMP must also take into account this aspect: with performance being equal, an overlay protocol should be preferred for its implementation advantages, mentioned in the Introduction.

ALMA [7] is one of the most promising overlay protocol and thus a good test for OBAMP. We compare OBAMP to ALMA and also to a modified version of ALMA, that we denote by ALMA-H.

Although we have implemented OBAMP, in Java, and we have tested it on the field, to prove its feasibility (see [11]), the limited number of available computers did not allow us to evaluate the OBAMP scalability when the group size increases. Thus, we resorted to carefully-designed simulations, taking into due account the recommendations given in [15] on how to produce meaningful simulation results. We start this section by describing the criteria that we followed to assure the so-called "simulation credibility".

A. Simulation credibility criteria

Credibility criteria adopted: i) we publish all the simulator source code and support files (e.g., movement scenarios, TCL files, post-processing routines, etc.) in [11], to assure the reproducibility of our study; ii) we produce the movement traces to be used as inputs of the simulations by using the "random trip model" [10], to assure that the stationary regime is reached; iii) we repeat each simulation ten times with different movement traces and random seeds and we plot the 95% confidence intervals in all figures; iv) we check, by means of post-processing procedures, that all simulation results do not significantly change in the last 100 seconds of simulation, to assure that we have reached the stationary regime within each simulation run.

B. Simulation tool

The simulation tool is based on ns2.29 [14] running on a cygwin32 platform. Unfortunately, none of the benchmarked protocols is available in the all-in-one ns2.29 package. Thus, we had to add the code simulating the three protocols as follows. As regards OBAMP, we modelled it from scratch. The ODMRP simulation code has been taken from [13]. As regards ALMA, we developed the code from scratch too, implementing two versions of this protocol, both proposed in [7]; in the first one, the metric used for parent selection is the round trip time (and we call it simply ALMA); in the second one (that we call ALMA-H) the metric is the number of hops. According to the ALMA Authors, the latter choice improves the performance of ALMA. Finally, we assume that the ALMA signalling packets are 8 bytes long (this value is not given in [7]).

C. Simulation details

Each simulation run lasts 800 s and all members join the group within the first 10 seconds. The simulated network is made up of 50 mobile nodes that move in a region of 1000m x 1000m. Nodes move according to a random way point model with a constant speed of 10 m/s and constant pause times of 30 s. The radio channel is modelled as free-space. The transmission power is regulated so that the radio coverage of each node is 250m. The MAC layer is IEEE 802.11 operating in DCF mode with a constant 2 Mbps bit rate. We use AODV as underlying routing protocol [4], without local repair and with link-layer detection. Other AODV configuration parameters used in our simulation are given in [12].

The main configuration parameters of the benchmarked multicast protocols are reported in Tab. 1, Tab. 2 and Tab. 3 for OBAMP, ALMA and ODMRP, respectively.

Tab. 1 – OBAMP main configuration parameters

Parameter	*Value*
HELLO_PERIOD	5 sec
FAST_HELLO_PERIOD	1 sec
TREE_CREATE_INTERVAL	5 sec
MAX_HELLO_TTL	4
Δ_h (handling delay unit)	250 ms

Tab. 2 – ALMA (ALMA-H) main configuration parameters

Parameter	*Value*
UPDATE_PERIOD	5 sec
THRESHOLD_LEVEL1	30 ms(1 hops)
THRESHOLD_LEVEL2	45 ms(2 hops)
THRESHOLD_LEVEL3	65 ms(2 hops)
THRESHOLD_LEVEL4	80 ms(3 hops)

Tab. 3 – ODMRP main configuration parameters

Parameter	*Value*
JOIN QUERY refresh interval	3 sec
ACK timeout for JOIN Table	25 msec
MAX JOIN table retransmissions	3

As regards the traffic model, we consider a *multi-source* scenario where each member sends out data packets of 256 bytes with an inter-packet interval such that the overall data traffic is equal to 16 kbps. This allow us to evaluate the impact of the increase in the number of sources while the offered traffic remains constant. Our multi-source scenario is more realistic for ad hoc networks, since it models many-to-many communications (e.g., push-to-talk), which are widely believed to be the most likely sources of load in a MANET.

D. *Simulation results*

We compare the four protocols at hand in the following four figures, as a function of the group size. The 95% confidence intervals are always plotted, when they are not visible it means that they are smaller than the curve markers. The first one, Fig. 2, reports a merit figure commonly named "byte sent per byte delivered" (*BSBD*). The *BSBD* is the ratio between the overall number of bytes sent by the network nodes (on the IP-MAC interface) and the number of data bytes delivered to the multicast sinks (excluding duplicate packets).

In Fig. 3 we plot the delivery ratio, i.e. the ratio between the number of non-duplicated delivered data bytes and the number of bytes supposed to be received by the multicast sinks.

In Fig. 4 we plot the average data latency, i.e., the time elapsing between the emission of a packet and its reception by the receiving multicast sink.

In Fig. 5 we plot the number of control bytes sent by all network nodes. In the case of overlay protocols (OBAMP and ALMA) we also distinguish between network layer and overlay signalling. In the following we comment these figures by comparing our solution first to ODMRP and then to ALMA.

1) *OBAMP vs. ODMRP*

As we can see from Fig. 2, the *BSBD* decreases as the group size increases for both OBAMP and ODMRP (see also [16]). This is due to the ability of these protocols to exploit radio broadcasting for data distribution: when the group size increases, more members can be reached by the same broadcast transmission and the *BSBD* decreases. As the group size keeps increasing, the effect of control traffic becomes more noticeable and the curves flatten. The important result of Fig. 2 is that OBAMP limits the network traffic much better than ODMRP. This advantage is confirmed by Fig. 5 which shows that the number of control bytes of OBAMP is significantly smaller than that of ODMRP.

As regards OBAMP's user-perceived performance, when the group size increases, Fig. 3 and Fig. 4 show that the delivery ratio and the latency increase only moderately, demonstrating the scalability performance of the proposed protocol. The first (positive) effect is due to the natural data redundancy provided by broadcasting, which is more and more apparent as the group size increases. The second (negative) effect is due to the increase of the control traffic which implies a higher latency.

Fig. 3 shows that the OBAMP's delivery ratio remains higher than 90% for all values of the considered group sizes, while ODMRP's delivery ratio suffers a heavy decrease for group sizes greater than 20 nodes, due to its significant network load. Correspondingly, Fig. 4 shows that a similar phenomenon occurs for the data latency.

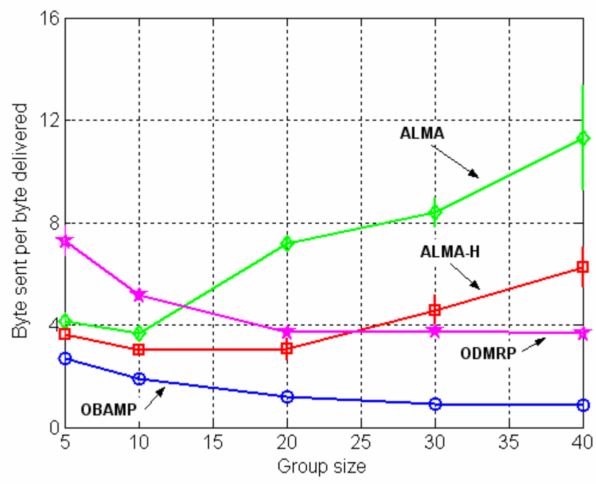

Fig. 2: Bytes sent per bytes delivered vs. group size

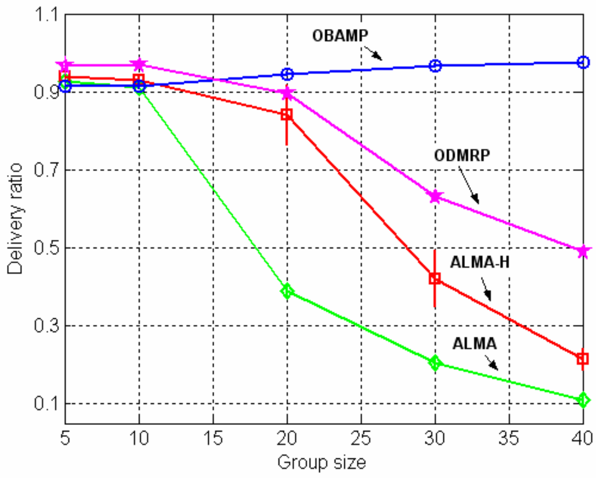

Fig. 3: Delivery ratio vs. group size

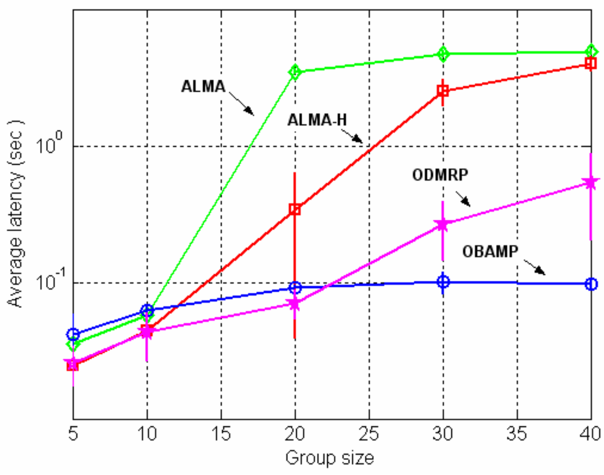

Fig. 4: Average data latency vs. group size

Finally, in all fairness, we must observe that the multi-source traffic scenario has indeed been selected because it is of interest for a MANET, but, on the other side, it turns out to be the worst possible traffic load for ODMRP. As a matter of fact, in [12], we show that in a single-source traffic scenario ODMRP and OBAMP have similar (and good) performance.

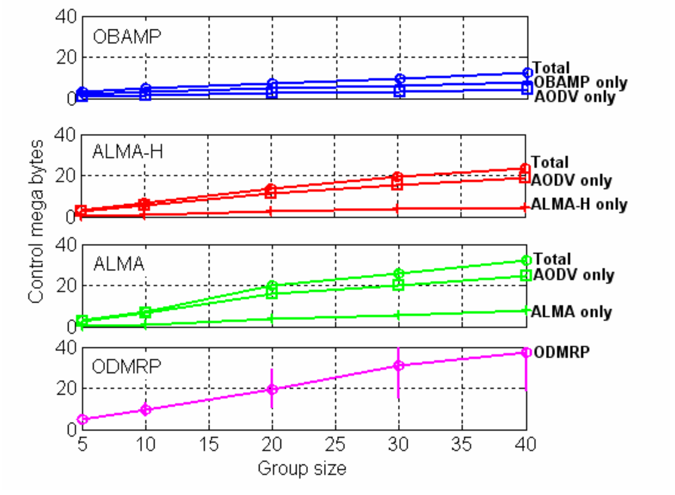

Fig. 5: Number and type of control bytes transmitted vs. group size

2) *OBAMP vs. ALMA and ALMA-H*

ALMA does not implement radio broadcasting; as a consequence, when the group size increases, the number of transmissions must increase and so does the network load, the latency and the loss phenomena. We start the comparison by looking at the performance of ALMA-H: we see that the *BSBD* rapidly increases (Fig. 2), the delivery ratio suffers a heavy decrease (Fig. 3) and the latency increases to undesirable values (Fig. 4). In Fig. 5 we can see how OBAMP and ALMA behave with respect to the amount of load induced at the network level. OBAMP succeeds in limiting the signalling overhead at the network layer (i.e., the AODV one). On the contrary, the figure relevant to ALMA-H demonstrates the need of taking into due account this source of overhead: the amount of network load induced by ALMA-H is a significant fraction of the overall signalling load.

As regards the difference between a single-source scenario and a multi-source one, in [12] we show that no significant difference can be observed between the two traffic models. The reason is that both OBAMP and ALMA-H form a unique shared tree that is not dependent “on where sources are” but only depend “on where members are”.

Finally, as regards ALMA, our results show that ALMA-H provides better performance than ALMA, as anticipated in [7].

V. Conclusion

Overlay multicasting is a valuable approach to support many-to-many communications in MANETs. A critical figure of merit for a MANET scenario is the ability of the multicast protocol to scale with the group size. To obtain this result it is important, in our opinion, to design the protocol with the follow three guidelines in mind: i) build a distribution tree that approximates the minimum spanning tree as much as possible; ii) design the protocol so as to limit not only the overlay signalling but also the induced network layer signalling; iii) exploit radio broadcasting.

We did follow these guidelines and we ended up with a protocol that exhibits better scalability performance in a many-to-many communications scenario with respect to two state-of-the-art protocols such as ALMA and ODMRP.

A last consideration is that the OBAMP protocol has the ability to account for asymmetric links. This requires only to add a suitable field in the Hello and FastHello messages. This feature is not presented in this paper for space limitations, but is implemented in the test-bed described in [11].

Acknowledgment

The work is partially funded by the FP6 EU project named E2R.

References

[1] J. Xie, et al., “AMRoute: Ad Hoc Multicast Routing Protocol,”*ACM/Baltzer Mobile Networks and Applications*, vol. 7 , no. 6, 2002, pp. 429 – 439.

[2] Sung Ju Lee, William Su, and Mario Gerla, "On-demand multicast routing protocol in multihop wireless mobile networks," *ACM/Baltzer Mobile Networks and Applications*, vol. 7, no. 6, 2002, pp. 441-453.

[3] E. M. Royer and C. E. Perkins, "Multicast Ad hoc On-Demand Distance Vector (MAODV) Routing, " In Proc. of the Second *IEEE Workshop on Mobile Computing Systems and Applications*, 1999.

[4] C. Perkins, E. Royer and S. Das. “Ad Hoc On Demand Distance Vector (AODV) Routing,” *IETF* RFC 3561.

[5] J. Nesetril, E. Milkov and H. Nesetrilov, “Otakar Boruvka on minimum spanning tree problem: translation of both the 1926 papers, comments, history”, *Discrete Math.*, vol. 233, 2001, pp. 3-36.

[6] A. Detti, C. Loreti, P. Loreti, “Effectiveness of Overlay Multicasting in Mobile Ad-Hoc Networks, ” *IEEE International Conference on Communications*, vol.7, 20-24 June 2004, pp. 3891 – 3895.

[7] Min Ge, Srikanth V. Krishnamurthy and Michalis Faloutsos, “Application versus network layer multicasting in ad hoc networks: the ALMA routing protocol,” *Elsevier Ad Hoc Networks Journal*, vol. 4, no. 2, pp. 283-300, 2006.

[8] Ki-Il Kim; Sang-Ha Kim, "A novel overlay multicast protocol in mobile ad hoc networks: design and evaluation," *Vehicular Technology, IEEE Transactions on* , vol.54, no.6, pp. 2094- 2101, Nov. 2005.

[9] C. Gui and P. Mohapatra, “Efficient Overlay Multicast for Mobile Ad Hoc Networks,” *Proc. 2003 IEEE Wireless Comm. and Networking Conf.*, vol.2, pp. 1118-1123.

[10] Le Boudec, J.-Y.; Vojnovic, M., "Perfect simulation and stationarity of a class of mobility models," *Proc. of IEEE INFOCOM 2005*, vol.4, 13-17 March 2005, pp. 2743- 2754.

[11] www.radiolabs.it/obamp

[12] www.radiolabs.it/obamp/Resorces/Documentation/obamp_ext.pdf

[13] “Wireless Multicast Extensions for ns-2.1b8” at http://www.monarch.cs.cmu.edu/multicast_extensions.html .

[14] “The Network Simulator - ns-2” at http://www.isi.edu/nsnam/ns/ .

[15] S. Kurkowski, T. Camp , M. Colagrosso, “MANET simulation studies: the incredibles,” *SIGMOBILE Mob. Comput. Commun. Rev.*, vol. 9, no. 4, pp. 50-61, Oct. 2005.

[16] S. Lee, W. Su, J. Hsu, M. Gerla, R. Bagrodia, “A Performance Comparison Study of Ad Hoc Wireless Multicast Protocols,”, *Proc. IEEE INFCOM 2000*, vol. 2, March 2000, pp. 565 – 574.

[17] A. Detti, C. Loreti, R. Pomposini, “Overlay Borůvka based Ad Hoc Multicast Protocol – Demonstration”, *Proc. IFIP Med-Hoc-Net 2006 – demo session*, 14-17 June 2006, Lipari (Italy).

[18] J. B. Kruskal, “On the shortest spanning subtree and the traveling salesman problem”, *Proc. of the American Mathematical Society*, no. 7, pp. 48–50, 1956.

[19] R. C. Prim, “Shortest connection networks and some generalisations”, *Bell System Technical Journal*, no. 36, pp. 1389–1401, 1957.