A cross layer version of OBAMP based on a proactive routing: description and MANET test-bed

Andrea Detti[#1], Remo Pomposini[*2], Roberto Zanetti[#3]

[#]*Electronic Engineering Dept., University of Rome "Tor Vergata", Italy*

[1]andrea.detti@uniroma2.it

[3]zanetti@email.it

[*]*RadioLabs, Italy*

[2]remo.pomposini@radiolabs.it

***Abstract*— This paper presents a novel version of the Overlay Borůvka-based Ad-hoc Multicast Protocol (OBAMP) [10], named *OBAMP cross-layer proactive* (OBAMPxP).**

OBAMP is an overlay protocol: it runs only in the end-systems belonging to the multicast group. User data are distributed over a distribution tree formed by a set of non-cyclic UDP tunnels.

With respect to OBAMP, the novelty introduced by OBAMPxP consists in the adoption of a cross layer approach based on a proactive routing protocol (e.g., OLSR [4]). The cross layering reduces the signalling overhead and simplifies some protocol functions. In addition, OBAMPxP does not require the underling support of flooding, as OBAMP does.

To prove the feasibility of OBAMPxP we have implemented it through Java language and we have done a set of experiments, proving the performance improvement of OBAMPxP versus OBAMP.

To allow fellow researchers to reproduce and test our work we published the source code of OBAMPxP in [9].

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). 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]).

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 setup of a multicast session and justifies the interest of the research community in this topic [6].

In this paper, we propose a novel version of the OBAMP protocol, named *OBAMP cross-layer proactive* (OBAMPxP). Whereas OBAMP does not require any specific routing layer; OBAMPxP must be placed on a proactive routing protocol, as example OLSR [4]. Through a cross layer approach, OBAMPxP reduces the signalling overhead of the protocol, by deriving topology information directly from the routing table, handled by the proactive routing protocol.

We have implemented OBAMPxP in Java and we have tested it on the field to prove its feasibility (see implementation codes in [9]).

The paper is organized as follows: section II describes the protocol; section III reports the experimental results and section IV outlines the conclusions.

II. OBAMPXP

To improve the readability of the paper, we describe all the functionalities of OBAMPxP. In all fairness, we note that with respect to OBAMP, only two procedures are significantly changed (*member and mesh join*, *mesh-create*) and the protocol concepts have been maintained.

The goal of OBAMPxP is to limit the network traffic, both user data and signalling information, so as to achieve high delivery ratio and low latency, and to maintain these properties when the group size increases. To this end, OBAMPxP: i) creates a cheap distribution tree; ii) exploits

radio broadcasting; iii) limits the protocol overhead exploiting a cross layer approach based on a proactive routing protocol.

OBAMPxP 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).

Both the mesh and the distribution tree are periodically updated, to follow the dynamics of the network topology. The mesh is maintained in a soft-state fashion, while the tree is hard-state.

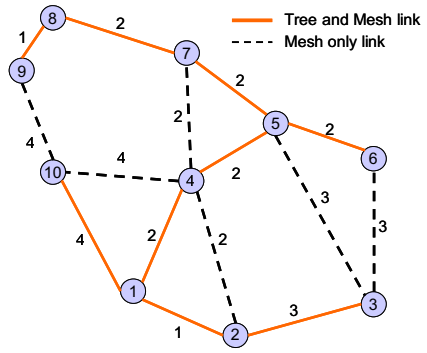

Fig. 1 - A mesh with a corresponding tree and related hop distance/cost, other MANET nodes not belonging to the multicast group are not drawn

Finally we note that, the mesh-first approach is not novel but has been derived by AMRoute [1]. OBAMPxP copes with the main lack of AMRoute that consists in having a "static" mesh [8]. Without adapting the mesh versus the topology dynamics, mesh links may become long overlay links and the resulting distribution tree may be seriously inefficient. Moreover, the distribution tree built by AMRoute can be identified as *the merging of the mesh shortest paths rooted at core*. On the contrary, the most efficient overlay distribution tree is the *minimum spanning tree*. In these terms, the OBAMPxP procedures provide a distribution tree that is an approximation of the minimum spanning tree.

In the following subsection we report the OBAMPxP data structures and procedures.

A. Data structures

This section reports the most important data structures of OBAMPxP.

1) Neighbours list

We define *neighbour* of member *M,* a member that is connected with *M* through a mesh-link.

The *neighbours list* of a member *M* contains the status information of the *neighbours*, i.e. of the mesh links incident on member *M*. Therefore, a duplex mesh link is established when the two endpoints mutually contains the other in their *neighbours list*.

A *neighbour list entry* is managed in a *soft state* fashion; this means that if the associated mesh link is not refreshed by the *mesh-create* procedure (and the links is not a tree link), then the mesh link is pruned and the relevant *neighbours list entry* is deleted.

2) Nodes list

The *Nodes list* is an off-line list containing all the "possible" nodes (i.e., their IP address) that may belong to the multicast group, independently on the fact that the specific node is running or less the OBAMPxP protocol.

3) Joined list

The *Joined list* is a subset of the *Nodes list* which contains only the members that are actually running the OBAMPxP protocol. The presence of OBAMPxP on a node, is periodically updated by each active member by sending a specific control message on the distribution tree.

B. Core and non-core nodes

Within a mesh a unique node is elected as the *core* of the mesh. A member selects the *core* by means of the *core election* procedure (see follow), which leads all members to chose the same *core*.

With difference of *non-core* members, the *core* performs additional management functionalities.

All control messages carry the *core identifier* (e.g., IP address) of the originating member.

A member only accepts control messages that carry a *core identifier* consistent with the selected *core*; unless these messages involve a new *core election*.

C. Member and Mesh Join procedures

When a novel member starts OBAMPxP, it declares itself as *core* of a mesh formed by only itself.

If another mesh is already active on the MANET, the two meshes must be joined. Therefore, the problem of "member join" resorts to the problem of "mesh join".

To join two meshes (and the "inner" distribution trees) a mesh *core* periodically tries to establish a tree link toward *cores* of disjoined meshes.

To find out a possible *core* of a disjoined mesh, the mesh *core* periodically sends the MESHJOIN control message toward a set of candidate core nodes. A node is a candidate core if: i) it is in the *Node list*; ii) it has a smaller IP address than the local one; iii) it is present on the MANET, i.e. in the IP routing table.

Each *core* that receives the MESHJOIN confirms the message and a tree link between the two meshes is established. The two meshes are now a unique mesh.

D. Mesh-create procedure

The *mesh-create* procedure refreshes the mesh topology.

It is periodically repeated by each member.

Considering a generic member *M*, the procedure setups duplex mesh links among the member *M* and the *nearest members set*. The *nearest member set* is formed by members having the minimum hop distance from *M*. The *nearest members set* is derived from the IP routing table and from the *Joined list*.

The setup of a duplex mesh link is accomplished by sending an HELLO control message to the remote member. An exception to this rule occurs when the remote member is at one hop distance. In this case none HELLO message is sent, because also the remote member will surely insert *M* in its *nearest member set* and, hence, the duplex property is implicitly achieved.

It is important to define that within the *nearest member set* the member *M* selects the *nearest member*, as the member that has the smaller IP address. This definition will be used during the following *tree-create* procedure.

Regarding the repetition period of the *mesh-create* procedure, it differs in case that the *nearest member set* is formed by members that are distant one or more network hops.

In case that the *nearest member set* contains one hop members, the repetition period is "short" (e.g., 1 sec.). This is due to the fact that when members are distant one hop, the data forwarding (hereafter explained) is done within an IP broadcast packet, without involving the network layer routing; therefore, the confidence of the distance estimation should be high to avoid data loss.

On the contrary, when the *nearest member set* contains members that are distant more than one hop, data forwarding is performed by means of a sequence of IP unicast packets; hence, the underling routing protocol will care to deliver the data, independently on the distance estimation performed by OBAMPxP. Therefore, the confidence on the distance estimation may be more loose and the repetition period may be "long" (e.g. 10 sec.).

The mesh resulting from the *mesh-create* is characterized by the following qualitative properties: i) the mesh is formed by a limited set of overlay links and this limits the protocol overhead; ii) this limited set of mesh links is formed by short overlay links and, therefore, the resulting tree will be formed by short overlay link, so improving the distribution efficiency. For formal analysis of the mesh properties we refer to the extended version of OBAMP reported in [9].

E. Tree-create procedure

The *tree-create* procedure builds a distribution tree by selecting a non-cyclic subset of mesh links.

The *tree-create* procedure follows the one proposed by AMRoute [1]. In addition, to improve the efficiency of the resulting tree, OBAMPxP extends the AMRoute *tree-create* procedure with a novel feature, named *handling delay,* hereafter explained.

The *tree-create* procedure is periodically started by the *core*. At the start of the *x*-th *tree-create* round, the *core* sends out TREECREATE #*x* messages toward its neighbours, where *x* is the sequence number of the *tree-create* round. TREECREATE #*x* messages are sent by means of an IP broadcast packet for neighbours at one hop distance, and by means of a sequence of IP unicast packets for the other neighbours.

When a member, *R*, "receives" a TREECREATE #*x* message from another member, *F*, it delays the "handling" of this message. 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)*Uh+r(0,1)* Uh /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.

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 *tree-create* round is marked as *old parent member*. Consequently, *R* setups a tree link with the *current parent member* and tears down the tree link toward the *old parent member* (in case that the two members differ).

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

Fig. 2 shows the *tree-create* procedure performed in an example network, with a distribution tree having a cost of 4 hops and formed by two tree links: *A*-*B* and *A*-*C* (Fig. 2a). A is the *core*. Member *B* is the *A*'s nearest, while the *B*'s nearest is *C* and vice-versa.

The core A sends out two TREECREATE #x messages toward B and C.

At the reception of the TREECREATE #x message (Fig. 2a): i) *B* applies an *handling delay* equal to zero seconds, since *B* is the nearest of *A*; ii) *C* applies an *handling delay* equals to 0.25*1 + 0.95*0.25/10 seconds, since the hop distance *A*-*C* is equals to 2 hops, U_h is equals to 0.25 and assuming that `r(0,1)` draws the random number 0.95.

Consequently, *B* immediately handles the TREECREATE #x message coming from *A* (Fig. 2b), confirms the overlay link toward *A* as a tree link and *B* forwards the TREECREATE #x message to *C*.

When *C* receives this TREECREATE #x message from *B*

(Fig. 2c), it applies an handling delay equals to zero seconds, as *B* is its nearest member. Consequently, *C* immediately handles such control message, setups a tree link toward *B* and tears down the tree link toward *A* (Fig. 2d).

We observe that the *handling delay* mechanism has changed the distribution tree from a sub-optimal tree, with a cost of 4 hops (Fig. 2a), to a more efficient tree, formed by 3 hops (Fig. 2d). Without the *handling delay* mechanism (i.e. as in AMRoute), the tree would have remained that of Fig. 2a.

Finally, we report three graph properties of the distribution tree, which are formally demonstrated in the extended OBAMP documentation available at [9]. We remind the reader that at application layer the best obtainable tree is the *minimum spanning tree* and that the *minimum spanning tree* can be derived by the Boruvka algorithm [5]. The algorithm is recursive and at each recursion determines an additional set of tree edges, until all members are connected. We call the set of tree edges derived at the first recursion as *first-level edges of the minimum spanning tree*.

Property A: *the tree created by OBAMPxP does not have persistent tree loops.*

Property B: *OBAMPxP builds a distribution tree that contains at least the fist-level edges of the minimum spanning tree.*

Property C: *the links of the OBAMPxP tree that are not first-level edges of the minimum spanning tree are the links of the macro-mesh that belong to the shortest-path rooted at the core.*

Therefore, looking at Properties B and C we can say that the distribution tree is an approximation of the *minimum spanning tree*.

F. Tree link-recovery procedure

OBAMPxP faces the issue of tree link failure by means of the *tree-link-recovery* procedure.

The activity of tree links is continuously monitored by the members. If no data packets need to be transferred, a periodic heartbeat control message is sent.

If a tree link becomes inactive, at the elapsing of a timeout the tree links is declared as broken.

The responsibility of re-establish the group connectivity is demanded to the *descendant* member of the broken tree link, assuming the tree rooted at *core*. To this aim, the descendant member setups a tree link with the *core*; otherwise, in the case that the other endpoint of the broken link is just the *core*, then the *descendant* member elects itself as *core*.

G. Core election procedure

The selection of *core* is performed by means of the *core-election* procedure that occurs during the *mesh-join* and *tree-create* procedures.

During the *mesh-join* procedure, the two involved *cores* select as *core* of the new joined mesh, the one of them with smaller IP address.

During the *tree-create* procedure a member change its *core* only when a TREECREATE comings from the *parent member* is originated by a different *core*.

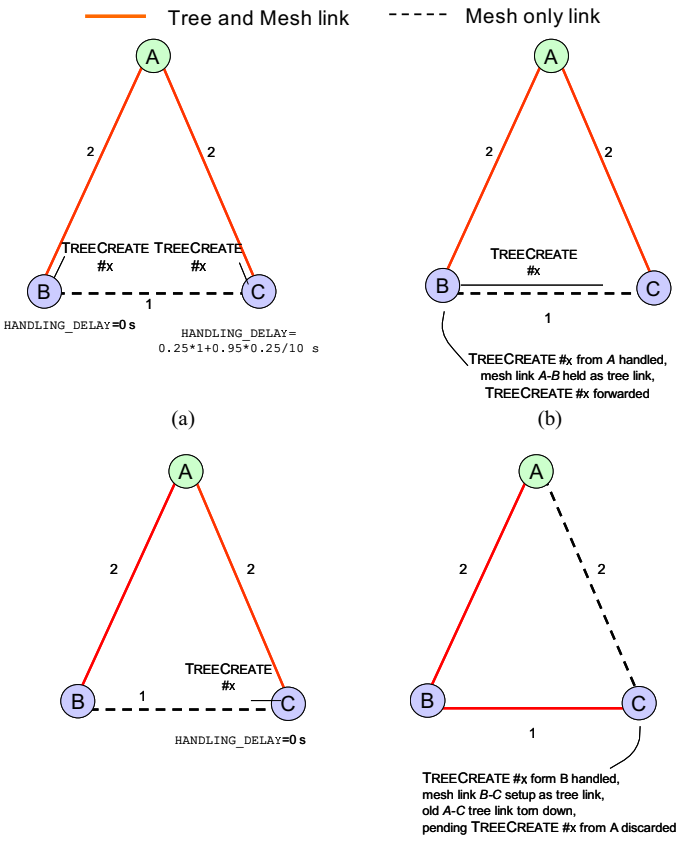

Fig. 2 - Steps of the *tree-create* procedure in an example network (a,b,c,d,e), non-member nodes are not drawn.

H. Data-distribution procedure

Usually [7][8], an overlay multicast protocol provides that data distributions is performed on the tree links through a sequence of IP unicast packets. This means when a member receives a data packet from a tree link, it forwards the data packet on all its tree links, with the exclusion of the receiving one.

As an exception to this modus-operandi, OBAMPxP provides that a received data packet is forwarded also on "mesh" links that are one hop long (i.e. the remote members are at one hop distance). At network layer, this transmission is done by means of a unique IP broadcast packet. So doing, OBAMPxP increases the transmission efficiency by exploiting the natural radio broadcasting property.

III. MANET TEST-BED AND PERFORMANCE EVALUATION

The test-bed architecture is formed by 6 laptops (o.s. Windows XP) equipped with IEEE 802.11b working in ad-hoc mode. The multicast group is formed by three laptops: 1,3 and 4. We test both OBAMP over AODV [3] and OBAMPxP over OLSR. The radio connectivity is emulated through MAC filtering and the connectivity patterns are shown in Fig. 2. These patterns differ in the average member distance, which values are: 1 hop (A), 1.6 hops (B), 2 hops (C), 2.6 hops (D), 3.3 hops (E). So doing, we test the performance versus the spatial density of the group, by changing the connectivity pattern.

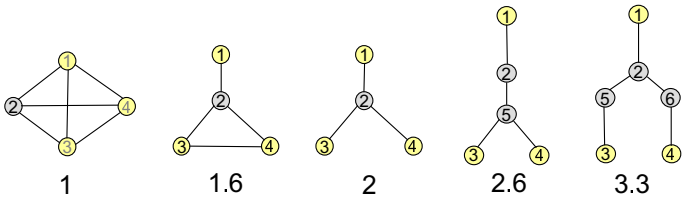

Fig. 3 – Radio connectivity pattern, links indicate available radio connectivity

As far as the protocols configuration, we exactly use the settings of the implementation codes reported in [9].

The comparison of OBAMP and OBAMPxP is done with respect to the signalling bit rate transmitted on the MANET. The data distribution performance is expected to be the same, since the same is the distribution tree and the data distribution procedure. Each test run for 60 sec and the MANET bit rate is captured through a WLAN card, operating in promiscuous mode.

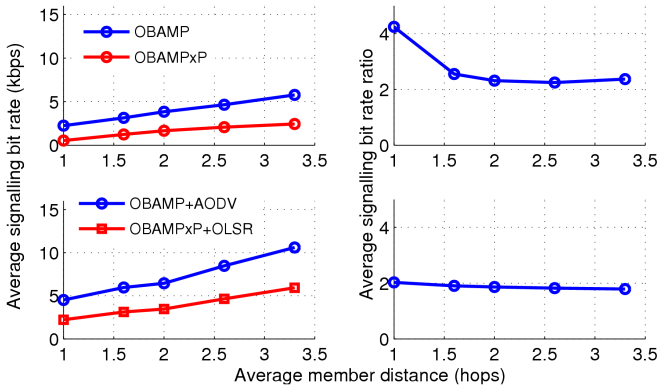

Fig. 4 - Average signalling bit rate of the overlay layer (upper-left plot), of the overlay+network layer (lower-left plot); of the ratio OBAMP/OBAMPxP (upper-right plot) and of the ratio (OBAMP+AODV)/(OBAMPxP+OLSR)

The plots in Fig. 4 report the average signalling bit rate of the overlay packets (upper plots) and of the overlay+network packets (lower plots). Moreover, the left plots show the relevant values and the right plots the OBAMP/OBAMPxP ratio.

We notice that *OBAMPxP halves the signalling load of OBAMP*. This occurs considering both the only overlay packets and all packets.

The constant behaviour of the signalling bit rate ratio is an indication that this conclusion should be true also for further increase of group sparseness.

Regarding the performance comparison at the increase of group size, we perform some additional test including also the laptop 2, 5 and 6 in the same patterns of Fig. 3. For lack of space we do not report here these plots. Anyway, the group size analysis has quite confirmed that OBAMPxP halves the OBAMP signalling.

IV. CONCLUSION

In this paper we have proposed a cross layer version of OBAMP, named OBAMPxP. The cross layering consists in the exploitation of the routing table of an underling proactive protocol.

OBAMPxP has be implemented in JAVA and the source code is available at [9] with GNU licence.

In a small test-bed OBAMPxP has halved the signalling load of OBAMP.

OBAMPxP is actually under test on the Freifunk mesh (in Berlin) and TuscoloMesh (in Rome). The proposed killer application is amateur "radio" diffusion.

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

[4] T. Clausen and P. Jacquet, "Optimized Link State Routing Protocol (OLSR)," *IETF* RFC 3626.

[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] 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.

[9] www.radiolabs.it/obamp

[10] A. Detti, N. Blefari, C. Loreti, "Overlay, Borůvka-based, Ad-hoc Multicast Protocol: description and performance analysis", to appear on IEEE ICC 2007.