Proc. IFIP Med-Hoc-Net 2006 – demo session, 14-17 June 2006, Lipari (Italy)

Overlay Borůvka based Ad Hoc Multicast Protocol – Demonstration

Andrea Detti
University of Rome Tor Vergata,
Electric Engineering Department,
Via del Politecnico 1, Rome, Italy
andrea.detti@uniroma2.it

Claudio Loreti, Remo Pomposini
RadioLabs
Via del Politecnico 1, Rome, Italy
{claudio.loreti,
remo.pomposini}@radiolabs.it

Abstract – **This paper describes the main functionalities of OBAMP protocol for multicasting in MANETs. Moreover, the paper describes two demonstrations of OBAMP in an emulated MANET environment: i) comparative performance evaluation versus flooding and unicast; ii) show cases of streaming and walkie-talkie applications.**

1 INTRODUCTION

An attracting use of Mobile Ad-hoc NETworks (MANETs) consists in performing cooperative applications during occasional team task. In such scenarios, network facilities should offer the support both for unicast and for multicast communications; moreover, in this case multicast traffic may cover a more important role with respect to the one that multicasting would get in public Internet; as a matter of fact, the occasional occurrence of the team task (e.g., tactics, rescue, etc.) may make more the need of real time multipoint-to-multipoint information sharing rather than classic point-to-point data transfer.

At network layer, a large number of multicast protocols have been designed for MANETs, but their standardization process is yet at a premature state involving the software developer to realise the multicast functionalities at application (or overlay) layer. We have designed an overlay multicast protocol for mobile ad-hoc networks named Overlay Borůvka based Ad-hoc Multicast Protocol (OBAMP). The OBAMP performance goal is the minimization of the number of transmission per packet sent while taking care of the packet delivery ratio and of the signalling induced at network layer (e.g., the signalling need to perform route discovery). After an extensive simulation campaign verifying the OBAMP performance, we have implemented OBAMP through JAVA language verify its functionalities in an emulated ad-hoc network environment, that is the focus of this paper. More detailed references and software code are available at [7].

2 OBAMP OVERVIEW

OBAMP maintains a dynamic set of overlay links, named *mesh*, spanning all member nodes. A subset of not-cyclic mesh links spanning all members forms the distribution tree; in Figure 1 we report an example of mesh and tree links. Although the OBAMP conceptual approach in managing the distribution tree is the AMRoute one [1], OBAMP implements relevant enhancements which cope with the AMRoute inefficiencies, while conserving the AMRoute pros in avoiding partitions and in requiring only a minimal knowledge of the network topology. Mainly, these enhancements consist in: i) OBAMP mesh is dynamic and runs after the radio connectivity evolution; ii) by means of delay mechanisms the distribution tree is more efficient; iii) OBAMP exploits the radio broadcast.

2.1 MESH management.

Let us discuss now how OBAMP manages the mesh. Each member node maintains in the *neighbours list* the status information [(1)] related to the mesh links to which the node is an edge; the remote edge is said to be a *neighbour*. The neighbours list, i.e. the mesh connectivity, is updated in a distributed way as follow: periodically each member starts a *neighbour discovery* round during which the member sends out broadcast HELLO messages with incremental IP TTL (i.e., expanding ring approach) that is also reported within the HELLO. When a member node receives the HELLO, it inserts the sender in the neighbours list and sends back the HELLO REPLY message including the originating HELLO TTL. At the reception of the HELLO REPLY the sender stops the current neighbour discovery round and inserts the source of

[(1)] Some examples of status information are: neighbour IP address, hop distance, expiration time, the link is or isn't a tree link, core address of the neighbour, etc..

the HELLO REPLY in the neighbours list. The TTL information included both in the HELLO and in the HELLO REPLY allows the receiver to know the hop distance from the sender.
Such as it is done by reactive ad-hoc protocols, the neighbours list is managed in a soft-state fashion through the use of timeouts. If a neighbour entry is not refreshed (i.e., HELLO or HELLO REPLY are not received) within a certain period of time, the neighbour, and therefore the related mesh link, is purged, unless that mesh link belongs to the current distribution tree.

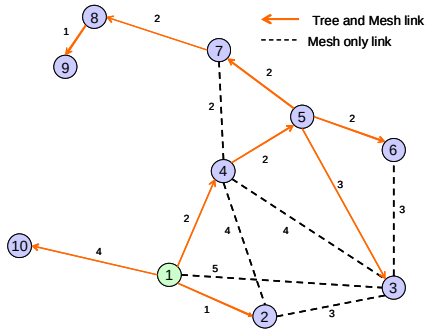

Figure 1- Example of mesh and tree links with the relevant hop distance (not member nodes participating to the radio connectivity are not drawn). Node 1 is the core.

2.2 TREE management

Over the mesh OBAMP builds a shared multicast distribution tree. OBAMP uses a tree creation/refresh mechanism regulated by a unique member node elected as the *core*. The core periodically starts refresh rounds. At the start of refresh round #i, the core sends out TREE CREATE #i message toward its neighbors. In order to build an efficient tree in terms of involved hops, at the reception of the TREE CREATE #i from a specific *forwarding* neighbour, the member node delays the handling of this TREE CREATE #i for a specific amount of time. This delay is proportional to the hop distance with the forwarding neighbour and it is equal to zero if this neighbour is the nearest one or vice-versa [(2)]. At the first handling of the TREE CREATE #i, the member node sets the mesh link toward the forwarding neighbour as tree link. Moreover, the forwarding neighbour is classified as the new *parent node* and the tree link toward the old parent node is switched off. Afterwards, the member node forwards the TREE_CREATE #i toward its neighbours with the exclusion of the receiving one. The successive TREE CREATEs #i are discarded.

[(2)] The hop distance is altered by a very small random value; so doing each mesh link get a different hop distance and the nearest is unique.

At least in case for which none network impairments occurs, by setting the nearest neighbour delay equal to zero the resulting OBAMP tree includes all the links connecting nearest neighbours; the same occurrence takes place at the end of the first step of the Minimum Spanning Tree - Borůvka algorithm. Regarding the other tree links, shortest links are preferred due to the increasing delay applied at the TREE CREATE reception.
Finally we mention that OBAMP faces with the member failures by checking the tree links activity and utilizing fast recovery procedures not reported here for lack of space.

2.3 Data distribution

The OBAMP data distribution provides that when a member node receives a data, it forwards the data on all others connected tree links. Moreover, OBAMP exploits the radio broadcast by transmitting only one copy of data for all neighbours that get hop distance equals to 1. To avoid not useful data retransmissions, the header of data packet contains a bit-map which represents the list of the members nodes for which the packet has just been forwarded.

3 IMPLEMENTATION

Although OBAMP can be implemented within the multicast application, we prefer to implement OBAMP as an independent module, which on behalf of the multicast applications is seen as a *proxy* (OBAMP proxy). The multicast application sends and receives data through local sockets, which are bound with the OBAMP proxy; on its turn, OBAMP proxy deals with the data distribution on the multicast tree.
The OBAMP proxy software architecture is depicted in Figure 2, it is composed of tree modules: *signaling manager*, *data manager* and *application handler*. The application handler sends/receives data to/from the multicast application. The signaling manager receives, handles and sends out OBAMP control messages while updating the neighbours list. Finally, data manager on the one hand handles the data distribution on the multicast tree, and, on the other hand sends/receives data to/from the application handler.
We developed OBAMP proxy in JAVA using the version 1.4.2. All network communications are realized through four JAVA Datagram Sockets: two for unicast/broadcast signaling and two for unicast/broadcast data. As far as the underlying unicast routing protocol is concerned, we resort to the UoB WinAODV [2] opportunely patched in order to support flooding functionalities.

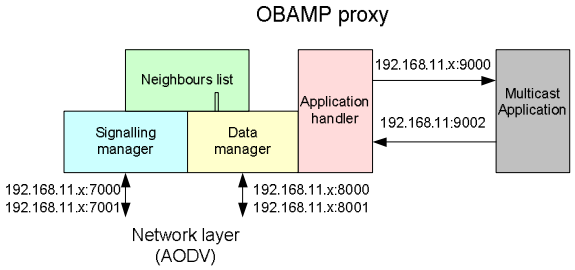

Figure 2 – OBAMP proxy software architecture

4 DEMONSTRATION

The hardware architecture is formed by 5 laptops (o.s. Windows XP) equipped with IEEE 802.11b working in ad-hoc mode. As depicted in Figure 3, three laptops participate to the multicast session, one laptop belongs to the ad-hoc but does not participate to the multicast group (i.e., only AODV) and the final laptop is the *connectivity manager* node. Connectivity manager is never MAC filtered and does not run AODV protocol. Moreover, the connectivity manager is the statistics collector, which are obtained through Ethereal [3].

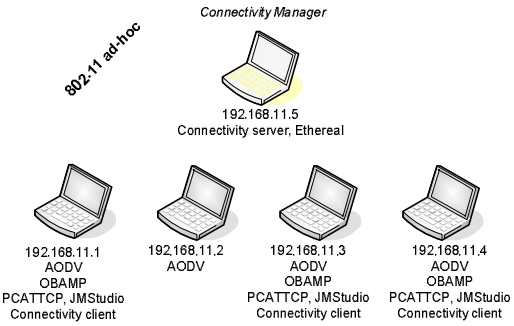

Figure 3 – Demo hardware architecture

The demonstration takes into account of four connectivity patterns reported in Figure 4. In order to perform the demonstration in a room and to emulate dynamic multi-hop radio connectivity, we resort to the well-known MAC filtering approach; as instance, in case C the nodes 1,3 and 4 mutually filters out their MAC addresses. The movement is emulated by the connectivity-server residing on the connectivity manager and by the related connectivity-clients residing on the member nodes. The MAC filtering configurations are timely broadcasted by the connectivity-server; at the reception of the current filtering configuration the connectivity-client applies the MAC filter on the node which is responsible to. During the demonstration every 30 sec the connectivity scenario is cyclic arranged according to the patterns reported in Figure 4.

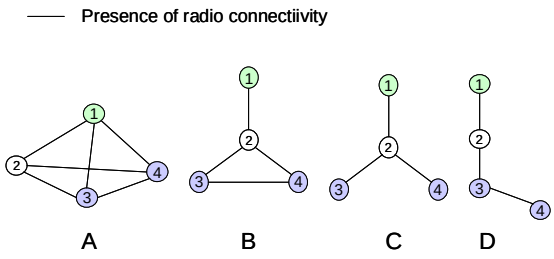

Figure 4 – Demo connectivity patterns

The demonstration consists in two parts:

- *performance demo* - through PCATTCP [4] and netperf [5] tools we compare the overall average bitrate yields by OBAMP versus unicast and flooding distribution strategies. The results obtained in our laboratory in case of a single 32 kbps CBR source placed on node 1 are reported in Table 1;
- *show-case* - through JMStudio [6] we reproduce both an audio streaming and a walkie-talkies multicast session.

	A	B	C	D
OB	42	119	157	118
FL	149	149	149	149
UN	77	152	152	189

Table 1 – Average network bitrate (kbps) for OBAMP (OB), Flooding (FL) and Unicast (UN) versus different connectivity patterns (A,B,C,D) in case of AODV routing and 32 kbps CBR source on node 1.

REFERENCES

[1] J. Xie, et al., "*AMRoute: Ad Hoc Multicast Routing Protocol*", ACM/Baltzer Mobile Networks and Applications, vol.7 No. 6, Dec. 2002.
[2] http://www.aodv.org
[3] http://www.ethereal.com
[4] http://www.pcausa.com/Utilities/pcattcp.htm
[5] http://www.netperf.org/netperf/
[6] http://java.sun.com
[7] http://www.radiolabs.it/obamp.htm

ACKNOWLEDGMENT

This work has been developed within the project Virtual Immersive COMmunications (VICOM) founded by the Italian Ministry of Instruction University and Research (MIUR).