Implementation of the OBAMP Overlay Protocol for Multicast Delivery in OLSR Wireless Community Networks

Francesco Saverio Proto, Claudio Pisa,
University of Rome Tor Vergata
{lastname}@ing.uniroma2.it

***Abstract*—The Optimized Link State Routing (OLSR) protocol is widely employed in Wireless Community Networks (WCNs), where users are connected to a wireless backbone with high bandwidth links. In these networks users can access both the Internet and internal network services. However the routers lack multicast support and users cannot exploit the high capacity of wireless links to stream and receive multimedia services. We implemented an extension to the OLSR protocol to support the delivery of multicast traffic using the Overlay Borůvka-based Ad-hoc Multicast Protocol (OBAMP). We present a demo of our implementation that is devised for the GNU/Linux operating system. The implementation has been tested both on standard PCs and on embedded devices running OpenWRT and it is distributed as a plugin within the olsr.org OLSR implementation. This protocol enhancement makes possible to create a multicast distribution tree among a subset of nodes, providing the mesh users with multicast community services.**

I. Introduction

OLSR [1] is one of the most widespread routing protocol in Wireless Community Networks[1] [2] [3] [4].

The goal of this work is to provide a simple tool to make possible for the users to run multicast streaming services over their WCN. We integrated the OBAMP overlay protocol [5] for multicast delivery in the OLSR routing protocol. The OLSR network is enhanced with the creation of an overlay multicast distribution tree for delivery of multimedia streaming.

In a real WCN is not feasible to install a multicast protocol daemon on every router. WCNs are characterized by a highly decentralized administration and, consequently, network nodes have very different hardware capabilities. For this reason we choose to use an overlay protocol: to avoid the installation of dedicated software on all the routers of the network.

The work we present in this paper is designed for use in real networks, it is backward compatible with the already deployed nodes in the network and it is transparent to the end-user.

II. Scenario

WCNs are characterized by fixed nodes mounted on the roofs of buildings and houses. OLSR was originally devised for MANETs, with mobile nodes, but in the case of WCNs most of the nodes have fixed positions. These nodes act as OLSR routers, where one or more radio interfaces are connected to the mesh backbone and send and receive OLSR protocol routing packets. The other interfaces, typically wired, are attached to IP subnets announced into the mesh network with HNA (Host and Network Association) messages. The end-user terminals do not have to run the routing daemon as their IP addresses are advertised by the nearest OLSR node. It is common that end-users get their IP address via DHCP from the nearest OLSR node.

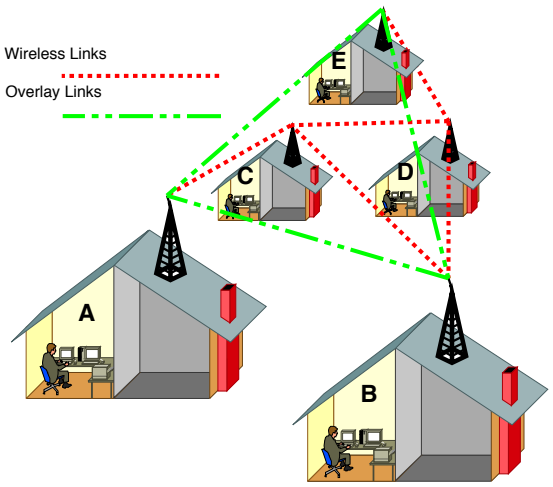

Fig. 1. Sketch of a Wireless Community Network

In figure 1 we can see a sketch of a typical WCN deployment. Users can communicate directly using the high capacity wireless links. The OLSR network does not support the delivery of multicast streams and the multicast domain of the user is thus confined to her subnet. To extend the multicast domain of the user, an OLSR node equipped with our OBAMP plugin can passively capture multicast packets and forward them into the overlay distribution tree. The captured traffic is then received by the other OLSR nodes participating in the overlay that can reproduce it on their attached subnets.

III. The OBAMP protocol

OBAMP is an overlay protocol: it runs only on the hosts willing to participate to a multicast group. User data is distributed over a shared distribution tree formed by a set of non-cyclic UDP tunnels.

[1]This work has been made possible from the collaboration of the SESAME research project http://www.sesame-project.net/ and the Freifunk Google Summer of Code 2009

The use of an overlay protocol makes possibile to add new features without redeploying the routing software on all network nodes at once.

The characteristic of OBAMP is that the protocol first makes a mesh network with overlay links (udp tunnels) between all the OBAMP nodes, and then it creates a distribution spanning tree over these mesh links. To limit signalling and improve the system scalability, OBAMP nodes do not build a full mesh among them, but create only the necessary links to keep the OBAMP overlay network connected.

Previous research [5] shows that the OBAMP protocol exhibits better scalability performance in a many to many communications scenario than two state-of-the-art protocols such as ALMA [6] and ODMRP [7]. In WCNs many to many communcation is the default scenario, because network traffic is generated by the users.

IV. Integrating OBAMP and OLSR

The OLSR routing protocol is designed for Mobile Ad Hoc Networks (MANETs) and characterized by the use of the mechanism of MPRs (Multipoint Relays) to optimize the diffusion of routing information. The protocol is extensible to provide additional functionality if desired.

In this work we extend the OLSR routing protocol devising and implementing an OBAMP extension. The OBAMP protocol has been already implemented [8] as a standalone application. However, the integration with the underlay routing protocol leads to optimization in terms of signalling and efficiency. Functions like neighbor discovery and path-cost calculations are done at once for both the overlay and the underlay protocol. This leads to greater accuracy and lower CPU consumption, that is critical on embedded devices like wireless routers.

This version of the OBAMP protocol, implemented as an OLSR plugin, is simplified for use in WCNs, where we assume the nodes to be in fixed positions on the roof of the houses. Thus all protocol features regarding mobility have not been implemented.

A WCN is usually composed by an OLSR backbone and subnets, attached to the OLSR routers, where the users are connected. Implementing OBAMP on the OLSR routers, and thus making the multicast delivery service transparent to the user, avoids the deployment problems that would be introduced by the installation of dedicated software on end user terminals.

An OLSR packet is composed by a header and a sequence of messages (Figure 2), thus it can be seen as a transport container for different messages. Two message types are fundamental for the OLSR protocol: HELLO and TC. All the others are optional: the protocol can be extended with other message types to support new OLSR applications. A new OLSR application can deliver information to all the other OLSR nodes, even if only a subset of nodes are aware of the new OLSR application. This is possibile because unknown OLSR messages are forwarded with a default algorithm.

To extend the OLSR protocol we define the *OBAMP alive* message as in Figure 3.

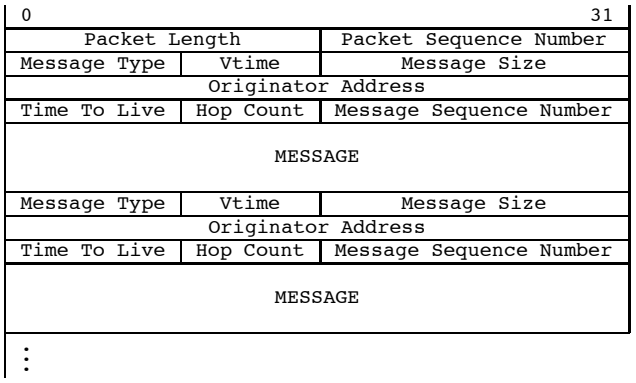

Fig. 2. Basic OLSR Packet Format

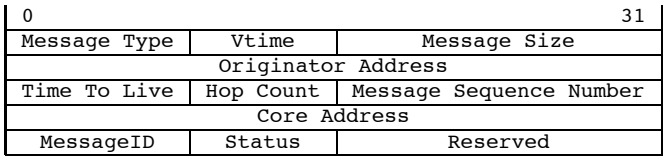

Fig. 3. OBAMP alive OLSR message

OBAMP-enabled OLSR nodes (*OBAMP nodes* from now on) generate periodically *OBAMP alive* messages. These are forwarded in the whole network (also by the nodes that are not OBAMP-enabled thanks to the OLSR design). Because of the flooding mechanism every OBAMP node has a complete list of all the other OBAMP nodes in the mesh network. The OBAMP nodes discover each other exploiting the OLSR signalling, to later start their own unicast signalling for the OBAMP protocol to create the overlay distribution tree.

V. Implementation

The OBAMP OLSR extension is developed as a plugin for the UniK OLSR Implementation, also known as *olsrd* [9]. The source code is distributed open source, and is freely available in the official olsrd distribution [10].

The natural choice for the OBAMP OLSR extension was to implement it as a plugin as it is not required for every node to support the overlay OBAMP protocol, but only a subset[2].

A. Operation Description

Each OBAMP node should have at least one interface participating in the OLSR network and at least one interface not participating in the OLSR network (*non-OLSR interface*), to which user hosts are supposed to be attached.

The key idea is to capture the IP packets containing the multicast traffic and encapsulate them in UDP tunnels for delivery on the overlay distribution tree. These messages, in order to avoid duplicate packets, are identified by a sequence number and by the IP address of the OBAMP source node where the traffic has been generated and hence encapsulated.

More specifically, the transport protocol is defined as follows: when a multicast packet is captured on a non-OLSR

[2]For example, backbone-only nodes, with no end users directly connected, may avoid using the plugin.

interface of an OBAMP node, it is encapsulated in the payload of an *OBAMP data* message and forwarded into the overlay multicast distribution tree. The other OBAMP nodes receiving the *OBAMP data* message will forward the data on the non-cyclic overlay spanning tree and will also decapsulate the contained IP packet and then send it through all their non-OLSR attached interfaces.

B. Implementation details

The plugin uses the olsrd scheduler to control the raw sockets descriptors to sniff multicast IP packets on the non-OLSR interfaces.

The OLSR scheduler has a default polling rate that is too slow for OBAMP to function correctly. However this is a tunable parameter (through the configuration file), and we found an optimal setting at 1 ms, that leads to an acceptable CPU consumption, even on embedded devices.

The plugin also registers itself to the scheduler to receive incoming *OBAMP alive* OLSR messages to discover the other OBAMP nodes. The rest of the signalling is received in unicast on the UDP port 6226.

C. Plugin Configuration

To enable the OBAMP plugin the following block must be added to the olsrd configuration file:

```
LoadPlugin "olsrd_obamp.so.1.0.0"
{
PlParam     "NonOlsrIf"  "eth1"
}
```

The only parameter the user has to specify are the interfaces, not participating to the OLSR network, where she wants to capture and decapsulate multicast traffic.

VI. Demonstration

We present a demo with heterogenous embedded IEEE 802.11 devices running the OpenWRT [11] operating system and the OLSR routing protocol. We enable the OBAMP plugin only on a subnet of nodes (like in the scenario in figure 1). These devices form an IEEE 802.11 ad-hoc mesh network. Users' computers attached with Ethernet cables to the devices where the OBAMP plugin is installed are able to send and receive multimedia streams over the mesh network with standard multimedia applications like VideoLAN [12]. We show how our implementation behaves when a new node enters or leaves the overlay network, reconfiguring automatically the overlay multicast distribution tree.

VII. Testing and Future Work

We proved the functionality of our implementation by testing it both in a virtual environment with Netkit [13] and in a real Wireless Community Network [3].

In future we want to optimize the implementation to support many OBAMP trees at once. Instead of having a single OBAMP tree shared among all the users, we want to extend the protocol to have different trees with different cores for each multicast stream in the network. This will lead to smaller OBAMP groups and lower bandwidth consumption.

VIII. Conclusion

In this paper we presented an extension to the OLSR protocol and its implementation, to deliver multicast traffic in wireless mesh networks. We make use of the optimized OLSR flooding mechanism to integrate the underlay and the overlay protocol signalling. As a result we obtain improved efficiency and reduced signalling. For improved usability we capture the multicast traffic at the wireless router to avoid the installation of dedicated software on the end user terminals. The protocol has been tested and disseminated in the Wireless Network Communities.

References

[1] T. Clausen and P. Jacquet. Optimized link state routing protocol (OLSR). IETF RFC 3626, October 2003.

[2] Freifunk: non commercial open initiative to support free radio networks in the German region. http://www.freifunk.net/.

[3] Ninux.org wireless community network. http://ninux.org/.

[4] Funkfeuer free net. http://www.funkfeuer.at/.

[5] Andrea Detti and Nicola Blefari-Melazzi. Overlay, Borůvka-based, ad-hoc multicast protocol: description and performance analysis. *Wireless Communications and Mobile Computing*, 2007.

[6] 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*, 4(2):283–300, 2006.

[7] Sung Ju Lee, William Su, and Mario Gerla. On-demand multicast routing protocol in multihop wireless mobile networks. *ACM/Baltzer Mobile Networks and Applications*, 7(6):441–453, 2002.

[8] Andrea Detti, Claudio Loreti, and Remo Pomposini. Overlay Borůvka-based ad hoc multicast protocol - demonstration. In *IFIP Med-Hoc-Net 2006 - demo session*, 2006.

[9] Andreas Tønnesen. Implementing and extending the optimized link state routing protocol. Master's thesis, UniK University Graduate Center - University of Oslo, 2004.

[10] Olsrd official website. http://www.olsr.org/.

[11] OpenWRT official website. http://openwrt.org/.

[12] VideoLAN - VLC media player. http://www.videolan.org/vlc/.

[13] Massimo Rimondini. Emulation of computer networks with netkit. Technical Report RT-DIA-113-2007, Dipartimento di Informatica e Automazione, Roma Tre University, January 2007.