The OLSR mDNS extension for service discovery

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

***Abstract*—In wireless community networks users can communicate with each other directly without accessing the public Internet. However, the typical use of these networks is to provide Internet access, with a subset of users sharing their broadband connectivity acting as gateways. Other services offered inside the community network are not exploited by the majority of the users. We believe this happens because essential tools like DNS and service discovery are hard to deploy in highly distributed and anarchic networks.**

In this work we propose an extension to the OLSR protocol to support the delivery of mDNS traffic. 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. This protocol enhancement makes possible to perform distributed name resolution and service discovery in community networks, using standard tools already installed on most users' computers.

I. Introduction

OLSR is the most widespread routing protocol in wireless community networks [1] [2] [3]. These open infrastructures are ran by volunteers to offer Internet connectivity to neighbourhoods or villages. Even if the users are connected directly one to each other on high speed wireless networks, few services other than Internet connectivity are popular among the members of the communities. To the best of our knowledge [1] community services are not exploited, because wireless community networks (WCNs) usually lack reliable internal DNS servers and service directories. WCNs are highly distributed and unstable, and this makes very hard to deploy centralized services. A DNS or service directory server could be most of the time unreachable to the majority of the users due to failures or network splits. Moreover, the standard DNS protocol [4] requires a centralized server as source of authority for the domain, but in the anarchic scenario of communities there is not a single entity responsible for the network's deployment, management and maintenance. In other words, WCNs are not distributed only in terms of topological position of the nodes, but also in terms of administrative domains. Setting up a DNS service for the domain of a WCN presupposes that an administrative domain covers the whole network, but this is not necessarily true.

When a central server is not available, name service protocols installed today on most users' terminals [5] [6] send broadcast queries. However, in a network running the OLSR routing protocol, the broadcast domain is limited to the 1-hop neighborhood. This makes it impossible to use a name service protocol that requires a broadcast medium.

The goal of this work is to provide distributed name resolution and service discovery in WCNs running the OLSR protocol. We devise a transport mechanism for mDNS [7] packets in a OLSR network, to perform name resolution where it was not possible with limited broadcast-multicast domain. Moreover, once the network is able to transport mDNS packets, it is possible to exploit DNS based Service Discovery to run decentralized applications with other users in the community network. The solution we present in this paper is completely distributed, it is backward compatible with the already deployed nodes in the network and it is transparent to the end-user. Standard user applications that are off-the-shelf in recent operating systems, like browsers or chat clients, are already able to exploit services announced via DNS based service discovery.

II. Multicast DNS and Service Discovery

Multicast DNS (mDNS) [7] is a protocol that uses APIs that are similar to the ones of the normal unicast Domain Name System, but implemented differently. The details of the mDNS protocols are out of the scope of this paper; in the following we give a short explanation functional to understand our work. Each host on the multicast domain stores its own list of DNS resource records (e.g. A, MX, SRV, etc) and acts as a DNS server. When an mDNS client wants to resolve a name, it sends a DNS query in multicast, and the host with the corresponding A record replies with its IP address. There are no central hosts responsible for the functioning of the whole system, and in case of failure or network split, the service still works between the users connected. Because of the lack of a central repository, names are assigned on a first-come basis. If a host discovers that its hostname is already taken from another host in the network then it chooses a new hostname.

DNS based Service Discovery (DNS-SD) is built on top of the Domain Name System. It uses DNS SRV, TXT, and PTR records to advertise Service Instance Names. The hosts offering the different services publish details of available services like instance, service type, domain name and optional configuration parameters. Service types are given informally on a first-come basis.

III. Scenario

Wireless community mesh networks are characterized by fixed nodes mounted on the roofs of buildings and houses. OLSR was originally devised for MANETs, with mobile

[1]This work was developed within the SESAME research project http://www.sesame-project.net/

nodes, but in the case of wireless community networks 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 networks 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 by the nearest OLSR node.

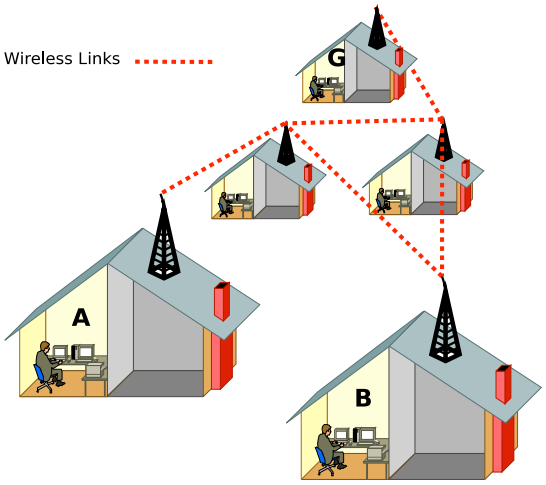

Fig. 1. Sketch of a Wireless Community Network

In figure 1 we can see a sketch of a typical wireless community deployment, where the node G shares its Internet connectivity acting as a gateway for all the other nodes. Obviously if the user attached to node A wants to communicate with the user attached to node B, it is not necessary to pass through the public Internet. However, if a user is offering a service on his host (e.g. a Web server) another user willing to connect to this service must know a priori the IP address and the port where the service is located. This is not feasible because i) the IP address may change over time, ii) users may not know each other a priori and iii) most of them are not skilled enough to go into the details of IP addresses and TCP/UDP ports.

To solve the problems listed above, we take advantage of the fact that most users' computers will generate mDNS traffic to announce the active network services. Moreover, hardware produced in the past few years like network printers and network attached storage boxes (NAS), generate mDNS traffic to announce their presence in the network and the available network services.

The mDNS traffic scope is inside the subnet of the user, and will not pass beyond the OLSR router. To extend the multicast domain for mDNS traffic, an OLSR node equipped with our mDNS plugin can passively capture mDNS packets and forward them into the mesh network as OLSR signalling (as we explain in detail in section IV). The captured traffic is

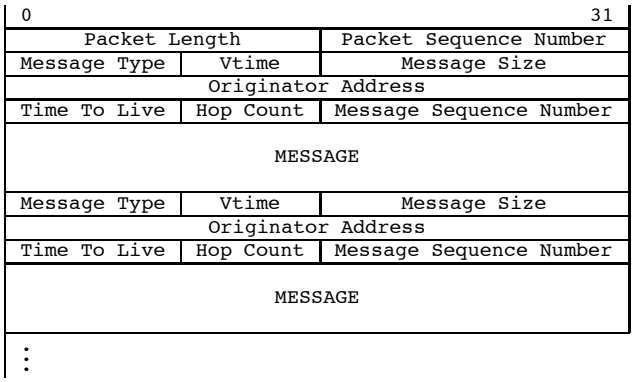

Fig. 2. Basic OLSR Packet Format

then received by the other OLSR nodes that can reproduce it on their attached subnets.

Network addressing configuration (or autoconfiguration) for the OLSR nodes and their attached subnets is not in the goal of this work. We assume that the network has a consistent addressing, and that all the nodes are configured correctly. Every host must have a unique IP address and must be routable: IP unicast connectivity between any pair of node must be possible. If the network is not fully routable, unreachable services may be announced, as a route to reach the host running the service may not exist. Because all the users must be able to offer services from their hosts, NAT must be avoided. It is useless to announce a service on a host behind a NAT: the service will not be available if does not exist a route to the IP address where the service is announced.

IV. Protocol

The OLSR protocol [8], is a routing protocol 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. The OLSR packet (Figure 2) is a transport container for different messages.

Two are the fundamental messages of the OLSR protocol: HELLO, and TC. HELLO messages are used for neighbor discovery and link sensing; these packets expire after one hop and are never forwarded. TC messages are used for network topology information diffusion; these packets are forwarded away from the originator to deliver topology information encapsulated into new OLSR packets at each hop.

The protocol can be extended with other message types to support new OLSR applications. An optimized flooding mechanism is devised for all the OLSR control traffic. A new OLSR application can deliver information to all the other OLSR nodes minimizing the traffic, even if only a subset of nodes are equipped with the new OLSR application. This is possible because unknown OLSR messages are processed according to the default forwarding algorithm. For example an OLSR node may want to advertise a subnet attached to one of its interfaces. The node floods an HNA (Host and Network

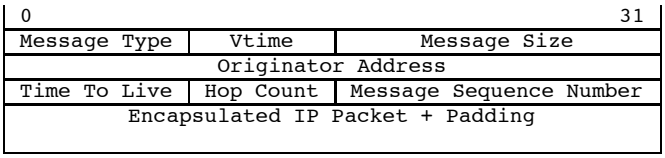

Fig. 3. mDNS OLSR message

Association) message to all the other nodes. A node capable of understanding the HNA application will add a route for the announced network in its routing table, while a node not aware of the HNA application will forward the HNA messages according to the default forwarding algorithm.

To extend the OLSR protocol we define the *mDNS message type* as in Figure 3.

The key idea is to capture the IP packets containing the mDNS traffic[2] and encapsulate them in the payload of the mDNS OLSR messages. Because our payload is limited by the MTU of the wireless interface, we introduced the smallest possible overhead in the message header. We have 44 bytes and 76 bytes of total overhead when working respectively in IPv4 and in IPv6. This means that with a typical MTU of 1500 bytes our protocol can deliver IPv4 packets up to 1456 bytes and IPv6 packets up to 1424 bytes. However, mDNS packets are typically smaller and different mDNS OLSR messages may be aggregated in a single OLSR packet.

The transport protocol is defined as follows: when mDNS traffic is captured, it is encapsulated into a mDNS OLSR message and flooded into the network according to the default forwarding algorithm. The flooding is natively optimized by OLSR to avoid useless retransmissions. An OLSR mDNS-aware node receiving a mDNS OLSR message, decapsulates the IP packet and sends it on its non-OLSR attached interfaces.

V. Implementation

The mDNS OLSR extension is developed as a plugin for the UniK OLSR Implementation, also known as olsrd [9]. The source code is distributed under the GPLv3 Licence, and is freely available in the official olsrd distribution [10].

The natural choice for the mDNS OLSR extension was to implement it as a plugin, that is loaded upon configuration, as it is not required for every node to support the mDNS application[3], but only a subset.

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

```
LoadPlugin "olsrd_mdns.so.1.0.0"
{
PlParam     "NonOlsrIf"  "ethX"
}
```

Where one or more interfaces not participating in the OLSR network are specified.

The plugin has two main functions that are registered into the olsrd scheduler. After creating raw socket descriptors to sniff IP packets on the desired interfaces, these socket descriptors are passed under the control of the OLSR main scheduler. When a mDNS packet is captured, a mDNS OLSR message is generated on all the OLSR interfaces of the node.

The plugin also registers itself to the scheduler to receive incoming mDNS OLSR messages: upon reception the encapsulated IP packet is sent over all the non OLSR interfaces of the node, and the message is processed according to the default forwarding algorithm.

The implementation is IPv6 ready. The olsrd daemon works with both IPv4 or IPv6 (but does not support the two protocols simultaneously). The mDNS plugin is able to work with IPv4 or IPv6 packets depending on the current configuration.

VI. Future Work

We proved the functionality of our implementation testing it both in virtual environment with Netkit [11] and in a real community network [12]. In future work we plan to test the plugin on other community networks with a higher number of users.

To reduce bandwidth consumption and for higher scalability, a future version of the protocol will compress the payload of the mDNS OLSR messages. This is mainly composed of ASCII text, therefore even with a lightweight compression it is possible to significantly reduce its size.

VII. Conclusion

In this paper we presented an extension to the OLSR protocol to deliver mDNS traffic in a wireless mesh network. We make use of the optimized OLSR flooding mechanism to solve the problem of the limited broadcast and multicast domain. This new feature lets users take advantage of DNS based service discovery tools already installed on their computers. The protocol has been implemented and disseminated in the wireless network communities.

[2]It is easy to filter mDNS traffic because it is sent over the well known UDP port 5353

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

References

[1] Freifunk: non commercial open initiative to support free radio networks in the German region - http://start.freifunk.net/
[2] Unimos.net comunidade portuguesa de aficionados da tecnologia wireless http://unimos.net/
[3] Ninux.org Wireless Community - http://ninux.org/
[4] J. Postel, “Domain Name System Structure and Delegation”, IETF RFC 1591, March 1994
[5] NetBIOS Working Group, “Protocol standard for a NetBIOS service on a TCP/UDP transport: detailed specifications”, IETF RFC 1002, March 1987
[6] B. Aboba, D. Thaler, and L. Esibov. “Link-local Multicast Name Resolution (LLMNR)”. RFC 4795 (Informational), January 2007.
[7] Cheshire and Krochmal, “Multicast DNS”, http://tools.ietf.org/html/draft-cheshire-dnsext-multicastdns-07, September 2008
[8] T. Clausen, P. Jacquet, “Optimized Link State Routing Protocol (OLSR)”, http://tools.ietf.org/html/rfc3626, October 2003
[9] Andreas Tønnesen, “Implementing and extending the Optimized Link State Routing Protocol”, UniK University Graduate Center - University of Oslo, 2004
[10] Olsrd Official Web Site http://olsr.org/
[11] Massimo Rimondini, “Emulation of Computer Networks with Netkit”, Dipartimento di Informatica e Automazione, Roma Tre University, http://www.netkit.org/ , RT-DIA-113-2007, January 2007
[12] Ninux.org Wireless Community Network testbed http://tuscolomesh.ninux.org