On the development of a IEEE 802.11s Mesh Point prototype

Rosario G. Garroppo, Stefano Giordano, Davide Iacono, Luca Tavanti
Dip. Ingegneria dell'Informazione – Università di Pisa
Via Caruso, 16 – 56127 Pisa – Italy
+39 050 2217511
{name.surname}@iet.unipi.it

ABSTRACT
Wireless Mesh Networks are a new, flexible and cost effective access technology that is gaining wide popularity replacing the traditional sets of wired IEEE 802.11 Access Points. Vendors and network operators have developed and deployed their own proprietary solutions, which are not compatible. The IEEE set off Task Group 802.11s to harmonize these equipments around a common standard. The draft however is far from its final version, and, though the major features are defined, work is still in progress. To support the test and evaluation of the features of this upcoming standard, we have built a prototype 802.11s Mesh Access Point. Starting from common off-the-shelf technology, we developed a modular software framework that can easily and quickly embed new features. In its current state, the prototype allowed us to experimentally evaluate the basic 802.11s characteristics, pointing out some shortcomings, such as the path instability due to the airtime metric, and suggesting possible improvements.

Categories and Subject Descriptors
C.2.1 [**Computer Communication Networks**]: Network Architecture and Design – *Wireless communication*; C.2.5 [**Computer Communication Networks**]: Local and Wide-Area Networks – *Access schemes*; C.2.2 [**Computer Communication Networks**]: Network Protocols – *Protocol verification*.

General Terms
Experimentation, Measurement, Standardization, Verification.

Keywords
IEEE 802.11s, prototype, wireless mesh networks, experimental testbed.

1. INTRODUCTION
The increased popularity and the growth in the number of deployed IEEE 802.11 wireless LANs [1] has posed new challenges to network operators. The typical 802.11 network is based on the presence of a single access point (AP) which is connected to the Internet by a wired link (e.g. an Ethernet cable). Many APs are then placed close to each other in order to extend the coverage range and capacity of the network. Hence, it is necessary for the operator to reach each AP with a cable, to coordinate the interactions between several adjacent APs and to manage user mobility and meet new application demands.

Wireless Mesh Networks (WMNs) have been introduced to solve these points. A few nodes interface to the external world through a cable, whereas several other nodes form a wireless backbone acting as relay nodes. Beyond forwarding the traffic received from their neighbors, mesh nodes can also provide network access to clients that are inside their coverage range.

From this description, it is apparent that WMNs inherits many of the features of ad-hoc networks. Yet, whereas the latter were conceived for occasional formation by mobile user devices, the former are mostly regarded as fixed, managed networks. Therefore, though some of the protocols and algorithms designed for ad-hoc networks may be profitably employed for WMNs as well, in most cases they must be adapted or re-designed, while some issues are completely new [2].

Thanks to their architecture, WMNs are suitable for both residential premises like offices (or even whole towns) and for rural areas or hardly accessible places, where bring a network cable might be problematical. From this point of view, this technology is much more cost-effective than the traditional set of wired APs. As a consequence, many companies have already put on the market their own WMN solutions (e.g. Motorola [3], Belair [4], Tropos [5]). However, though most of them are based on the common 802.11 MAC, these products are not interoperable. In order to harmonize these technologies, in May 2004 the IEEE set up the 802.11s working group (TGs) whose goal is defining a standard architecture for WMNs. Unfortunately, the drafting of this new standard has turned out to be rather troublesome, and the release of the official draft was postponed several times.

At the moment the major details of the IEEE 802.11s architecture have been defined [6]. Nevertheless, many issues are still open and without a clear definition and new proposals are still being submitted to TGs in order to complete and improve the upcoming standard [7]. These have been so far evaluated analytically or through simulations (see for example [8], [9], [10], [11]). However, since models and simulation results may sometimes be quite far from reality, there is no experimental evidence of the goodness of the proposed solutions. This has led us to start an

TRIDENTCOM 2008, 17th – 20th Mar 2008, Innsbruck, Austria.
Copyright © 2011 – 2012 ICST ISBN 978-963-9799-24-0
DOI 10.4108/icst.tridentcom.2008.3129

implementation of a 802.11s compliant WMN node to be used to experimentally verify the effectiveness of the draft standard and its possible amendments. Our prototype, built using common hardware and software tools, has already allowed us to spot some shortcomings of the 802.11s proposal and also to find some possible improvements.

The paper is organized as follows. The next Section provides a detailed overview of the main features of the 802.11s draft. Then, in Section 3, we describe how we realized our prototype, highlighting the major implementation issues. The section ends with describing an experimental WMN testbed. Remarks on its outcome are presented in Section 4, while Section 5 concludes the paper.

2. THE IEEE 802.11s DRAFT

Although an official draft has not been released, the general architecture of the system and the definition of the major protocols and algorithms seem to be stable [6]. The names of some elements, however, change as updates are brought to the draft. In the paper we refer to Draft D1.02.

The IEEE 802.11s builds on some already approved amendments to the standard, like 802.11a/b/g/n for the physical interface, 802.11e for accessing the medium and 802.11i for security, but it also conceives a new network architecture. Therefore it introduces new mechanisms and frame formats for the configuration and operation of the Mesh network.

2.1 Mesh architecture

As outlined in the Introduction, the basic network entity is the Mesh Point (MP). Beyond having all the characteristics of any traditional 802.11 station, each MP is also called to relay the traffic generated by other MPs to let them reach their intended destination through a multi-hop path in the Wireless Distribution System. The set of connections among the MPs forms the wireless backbone of the Mesh.

A MP can also have additional features, such as gateway/bridging functions, which allow it to connect to an external network, like the Internet. In this case, the MP is called Mesh Point Portal (MPP), or just Portal. Every 802.11s network may have one or more Portals, and each MP chooses which Portal to use to get access to the external world. Legacy client stations (STAs) are also supported. Accordingly, another option for a MP is to give STAs access to the distribution system. In this case a MP becomes a Mesh Access Point (MAP) and must offer all the functions provided by the basic 802.11 BSS. From the STA point of view, the Mesh must be completely transparent. Stations do not have awareness of the mechanisms working within the Mesh, and each MAP shall then act as a Proxy for its associated STAs. The typical Proxy operations will be detailed in the next Sections. An example mesh network with all the elements defined by the 802.11s draft is reported in Figure 1.

2.2 Frame formats

The draft adds new frame formats and modifies some of the existing ones. Most updates deal with the management of the Mesh services and algorithms. The new frames are the Mesh Management and Mesh Data, both of the Extended type. Both holds a Mesh Header, which consists of 4 or 16 bytes borrowed from the data field. In particular, the Mesh Header is used to store the two extra addresses to forward the frames generated by user stations. The changes to the existing frames consists in adding new Information Elements (IEs) in the data field of the Management frames (the Beacon, in particular). Describing all the changes is out of the scope of the paper, therefore we point the interested reader to references [6] and [12].

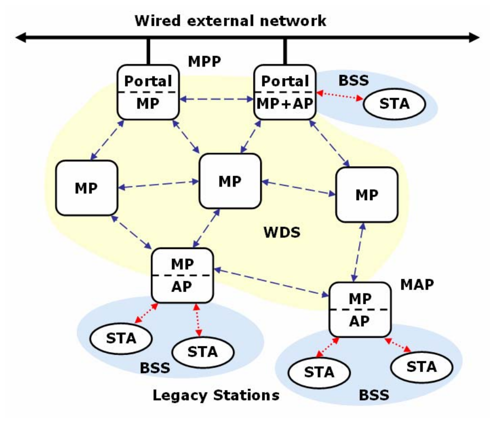


Figure 1. An example IEEE 802.11s network.

2.3 Path selection: HWMP

A forwarding algorithm is necessary to build the set of paths that form the Mesh backbone. Instead of exploiting any common layer three routing protocol, 802.11s brings this feature at level two of the TCP/IP protocol suite. This technique makes the Mesh transparent to upper layer protocols (e.g. IP), that see any intended destination within the Mesh to be only one hop away. The Hybrid Wireless Mesh Protocol (HWMP) is the mandatory algorithm that all MPs must implement to guarantee the full functioning of the Mesh. However, other proprietary protocols may also be employed.

HWMP combines a flexible on-demand algorithm (mainly drawn from AODV [13]) with a proactively built tree topology. The second technique is used when at least one Portal is present in the network, since the tree shall be rooted at the Portal. This protocol will then set up and maintain a tree that connects all MPs to the Portal, so that a path is always available towards the outside and between all MPs. Having this tree at the ready brings several advantages: broadcast management traffic is reduced, flooding to search for destinations can be restricted to devices that are inside the Mesh, and on-demand paths can use the tree links as back-up.

Both the on-demand and the proactive techniques use common messages and processing rules. The Route Request (RREQ), Route Reply (RREP), Route Error (RERR) and Root Announcement (RANN) frames are flexibly structured to allow for the needs of both protocols. Path selection control messages are transported in the new Mesh Management frames. HWMP uses sequence numbers to avoid loops.

The proactive tree can be set up in two ways. The Portal can broadcast either Proactive RREQ (PRREQ) frames or Portal Announcement (PANN) frames. The first technique aims at creating and maintaining a set of paths towards the root from all MPs. A MP receiving the PRREQ replies with a unicast Gratuitous RREP (GRREP), where it can also insert the address of the MPs that use it to reach the root (called dependent nodes).

The second technique just disseminates information on how the root can be reached, leaving each MP the possibility to set up the path whenever it needs it. The procedure is the same for the on-demand path selection algorithm, with the exception that the RREQ is sent directly to the root (unicast). Both PRREQ and PANN are re-broadcasted by each MP.

A MP can decide to change its path to the root (due for example to a change in the path metrics). It must therefore use a unicast GRREP sent to the root. To allow a rapid path recovery when a link or node failure occurs, a Route Error (RERR) message is broadcast by the MP that detects the failure. However, the draft is not very clear about how this RERR should be handled.

On-demand path selection in HWMP works much like AODV. A source Mesh Point S wanting to send data to a destination MP D broadcasts a RREQ frame indicating the MAC address of D. Two flags in the frame specifies the handling policy of the frame: DO (Destination Only) and RF (Reply and Forward). If the DO flag is set, only the destination is allowed to reply to the RREQ, otherwise any intermediate node having a path to D can answer to S's request. If RF is set, intermediate nodes may reply to S, but shall nevertheless re-broadcast the RREQ. In any case, any node receiving the RREQ can add or update its path to S. Once D receives the RREQ, it adds the path to S to its forwarding table and sends S a unicast RREP. Intermediate nodes shall then forward this RREP to S along the best path. When the RREP reaches S, the path is set up and can be used for exchanging data.

If a tree is also present in the Mesh, S can also use the tree to send the first data packet to the Portal, which will then forward it to D asking him to establish a direct path to S using the on-demand technique. This feature is very useful when the source MP has no knowledge whether the destination is internal or external to the Mesh. Since the Portal has this information, sending the frame to it will permit to take the best decision with the least use of network resources.

Since HWMP works at layer two, frame forwarding is performed on the basis of the MAC addresses. Usually four addresses must be present in the frame header: source, destination, transmitter and receiver. Sometimes, however, two more addresses must be added. This is the case of a communication between two non-mesh devices, which do not have awareness of the path selection and frame forwarding mechanisms. Client stations are the most common case of non-mesh devices. The MAPs shall then modify the frames generated by the STAs. The frame relayed across the Mesh will then carry five or six addresses: the source and/or destination STAs, which are the actual end points of the path, their Proxy MPs, necessary to forward the frame within the Mesh, and the transmitter/receiver pair. However, the frame header can still hold four addresses at most, and the two extra addresses shall be accommodated in the Mesh Header.

2.4 The Airtime metric

To select the best paths, the 802.11s draft defines the mandatory "airtime" metric. Other metrics can also be used. The airtime metric is very similar to the well known ETX metric [14]. The definition is the following:

$$C_a = \left[O + \frac{B_t}{r} \right] \frac{1}{1 - e_{pt}}, \qquad (1)$$

where O is a constant that quantifies the protocol overhead, B_t is the test frame length (in bits), r is the transmission rate (in Mbps), and e_{pt} is the test frame loss ratio. MPs should continuously monitor and probe their links to the neighbors to keep the metric up to date with the current network state.

2.5 Station management

The draft provides for the management of user stations (STAs) through an exchange of control messages in the Mesh. Each MP shall keep a list of all the STAs in the network, associated to any MAP, and the address of their MAP Proxy. A unicast Proxy Update (PU) frame is sent by a MAP every time a STA associates or disassociates with it. A PU Confirmation (PUC) shall be sent back by the MP which the PU was addressed to. Many other aspects of the handling of such messages are still unclear in the draft. For example the RREQ frame too holds some fields where the addresses of the proxied stations can be added.

2.6 Other features

802.11s also provides for MAC enhancements. The basic MAC access technique shall be based on the approved 802.11e amendment. However, a new optional access scheme has also been proposed. The Mesh Deterministic Access (MDA) is a distributed scheduling algorithm based on the reservation of contention free time slots.

It is not infrequent that MPs are equipped with more than one radio interface, and therefore using multiple channels is a sensible choice to considerably increase the capacity of the network. 802.11s takes this aspect into account defining a multi-channel framework. A series of interfaces tuned on the same channel forms a Unified Channel Graph (UCG). More UCGs can coexist in the Mesh, and a MP may belong to more than one UCG, depending on how many radio interfaces it has. The draft describes a protocol to select and update the channel for each interface and also allows for vendor specific algorithms. However, since channel allocation is not the subject of the paper, its details will not be given here.

Finally, the draft provides for two address types: individual address, which is the classical unicast address, and group address, which refers to a group of MPs, thus being the traditional multicast and broadcast addresses.

3. IMPLEMENTING A IEEE 802.11s MP

Our goal was to implement a prototype 802.11s station based on the existing hardware and using open source software. The main innovation of 802.11s is the path selection facility, which in fact is routing brought to layer two. Integrating at this level the various routing functionalities has indeed been the most challenging task, as it was sided with the need of creating the new 802.11s frames.

An efficient implementation would have asked for a deep-rooted integration with the device driver. However, some reasons discouraged this choice. At first, the same 802.11 defines the new Mesh services as a dedicated logical interface, independent from the legacy MAC functions. For instance, some actions are based on the analysis of the new IE fields, which are part of the data structure of the legacy 802.11 management frames. The two interfaces can therefore coexist within the same station.

Secondly, 802.11s defines a rather complex set of features, many of which are still in evolution. Bringing them all at driver/firmware level would have reduced the possibility to frequently change and update them without much gain in terms of performance (as most driver operations are about handling data frames). In the context of our development activity, whose goal is to readily test solutions and protocols in the framework of the last available draft version[1], this solution was not advisable. Therefore we decided to place our software partly within the driver (handling data frames) and partly on top of it (the management functions).

Our attention was mainly focused on the procedures for the creation of the tree topology and for the operation of the HWMP proactive protocol. Throughout our work we assumed that, since a major task of the Mesh is providing broadband access to the users, most traffic is directed towards (or comes from) the Portal. Hence, for the parts of the draft which are unclear or incomplete, we resolved to fix them in the perspective of this assumption.

3.1 Hardware and Software tools

The software building our prototype has been written in the C language under the Linux operating system (Slackware distribution, kernel 2.4). The wireless cards are based on the Atheros chipset, which comes with the Multiband Atheros Driver for Wi-Fi (MadWifi) [15]. This driver has the great advantage of being completely open source, thus allowing an easy integration.

MadWifi provides for the creation of several separate instances of the driver that appear to the operating system as different wireless interfaces. Each instance is called Virtual Access Point (VAP)[2]. All VAPs, however, work with the same physical device. A VAP can be set up in one of these modes: *ad-hoc*, *ap*, *sta*, *wds*, *monitor*. The mode determines the VAP behavior. The first three modes are quite self-explanatory. The driver instance, and thus the station hosting it, behaves as defined in the basic 802.11 standard for the respective name (e.g. a *sta* VAP behaves as a managed mode wireless interface). The *wds* mode, instead, can be used to form a point-to-point wireless link between two stations (usually two APs). In practice it is realized using all the four addresses of the 802.11 frame header with the “from DS” and “to DS” flags set. The *monitor* mode can be exploited to sniff traffic from the network, but also to send “fake” frames over the air. These frames can be built directly by the user and are not processed by the driver. This mode has been exploited to create the new Mesh Management frames introduced by the 802.11s draft. Frames entering or exiting the physical device can therefore undergo a different set of processing rules, depending on the VAP they pass through.

A native Linux module, the bridge, has been used to connect the interfaces created by MadWifi. This module behaves exactly like a hardware bridge, forwarding on its ports the frames according to their destination MAC address. Each port of the bridge can be either blocked or open. In the latter case, we say it is in the “learning” state, since the bridge learns the association between the source MAC address of the incoming frame and the port it entered. Then, when it receives a frame with that address, it immediately knows to which port it shall be sent. A forwarding (or learn) table is built with all the learned associations. If an address is not in the table, the bridge, like a hardware bridge, sends the frame over all ports (apart the one it entered, of course).

Finally, Ebtables [16] has been used to filter and modify the frames traversing the station. It is placed between the VAPs and the bridge, allowing processing the frames passing from one interface to the other (dropping, changing the source and/or destination addresses). In particular it has been used to virtually close the bridge ports when building the HWMP tree.

3.2 Implemented framework

In the implemented solution (see Figure 2) the new Mesh services are transparently added to the existing 802.11 functions. The whole system appears as an 802.11s software overlaid over the legacy 802.11 hardware/software. The software framework we developed has been organized in a series of modules, to allow testing different WMN functionalities in several configuration options.

Every prototype MP provides both mesh connectivity and network access to the clients, which are the typical functions of a MAP. Three modes have been used to set up the VAPs in each station: the *ap* mode, to support the clients, the *wds* mode, to create one or more interfaces to form the WMN backbone, and the *monitor* mode, to send and receive the new 802.11s frames. Clearly, since each *wds* VAP establishes a single static link to another MP, it would have been more sensible to use a single interface in the *ad-hoc* mode. Unfortunately, the current release of MadWifi (version 0.9.3.1) does not support the coexistence of the *ap* and *ad-hoc* modes.

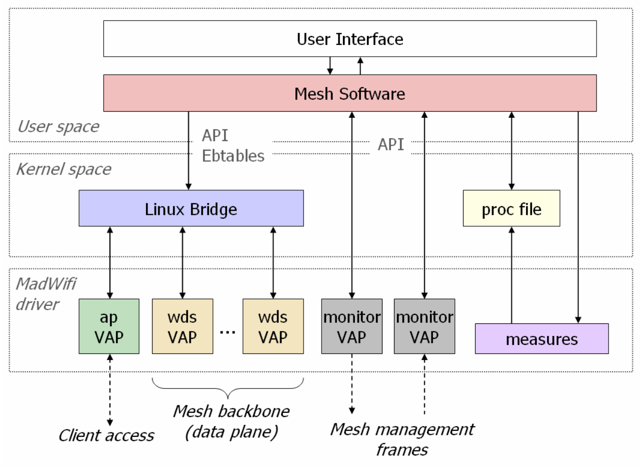


Figure 2. Architecture of the implemented software framework.

Every *ap* and *wds* VAP is connected to a port of the bridge, that forwards the frames between the interfaces. The monitor VAPs, instead, are not connected to the bridge, but are managed directly from the framework at user space. These interfaces are used to receive all management frames (legacy and new) and to send the new Mesh Management frames. Finally, the MadWifi driver can

[1] At the moment of our implementation, the latest available draft was version D1.02.

[2] Note that the MadWifi naming is misleading, since a Virtual Access Point does not necessarily behave as an Access Point.

provide the Mesh software several statistics about the quality of the links.

The framework implements two functional levels: the management plane and the data plane. The former is made of all the procedures to build and maintain the Mesh, run by the software at user space, that exploits the monitor VAPs and controls the behavior of the bridge. The latter just handles the data frames, and works entirely at kernel space (Linux bridge and *ap* and *wds* VAPs), exploiting the paths set by the management plane.

Each *wds* VAP is connected to its homologue in a neighbor MP. Since the *wds* VAPs are statically configured, the network appears as a series of LAN segments. The formation of the mesh paths is then achieved by enabling the ports that are connected to MPs that are parent or children in the tree built by the HWMP proactive mode. The other ports are closed, and are enabled only if a change in the tree occurs. An example is reported in Figure 3.

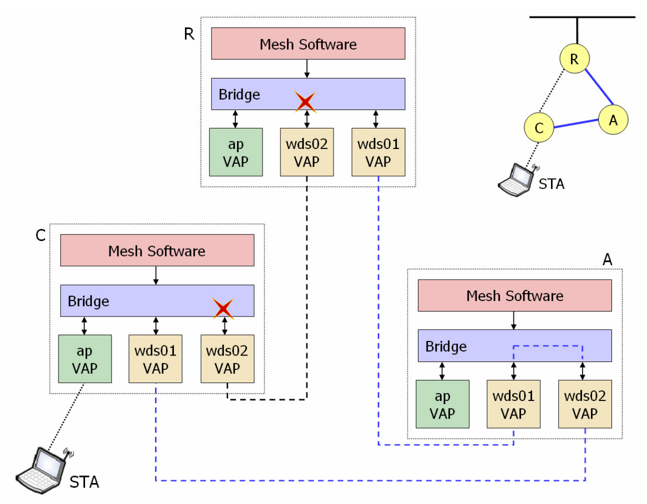


Figure 3. An example of bridge port configuration for the reference tree (in the upper right corner). Blue lines are the tree paths.

The bridge works with Ethernet frames, where only two addresses (transmitter and receiver, which are supposed to be directly connected to the bridge ports) are present. However, 802.11 frames can hold four (sometimes even five or six), addresses, all used in the Mesh framework: transmitter, receiver, and source and destination of the end-to-end path. The card driver is in charge of translating between the two frame formats. In doing this, it maps the 802.11 source and destination addresses to the Ethernet addresses. Therefore it can easily happen that these addresses are not in the forwarding table of the bridge, which is usually limited to MPs one-hop away. As a consequence, it can not be directly used to forward the frames along an end-to-end path built by the HWMP protocol. Rather it must be instructed to do so, i.e. entries must be added to its learn table. This can easily be done by setting the ports in the learning state and having proper frames pass through the bridge. With "proper frame" we mean a frame in which the address of the source (of the end-to-end path) MP/station appears. This is actually not infrequent, as many frames are suitable for this purpose, e.g. RREQ and ARP. An example is given in the Section 3.4.

3.3 Implementing issues

The 802.11s Mesh Management frames have been created and handled entirely through the developed mesh software at user space. However, it was not possible to modify the basic 802.11 frames to include the new IEs, nor create the six-address Mesh Data frames. This would have required a deep revision of the whole driver, which was well beyond our goal. On the other hand, processing all the frames at user space is not efficient, as it requires much more computational resources and offers much lower performance than the driver level. Therefore we brought at user space only the incoming legacy management and the new Mesh Management frames, which are a very restricted part of the whole traffic and nonetheless are enough to let the software fully operate the management plane. To let the rest of the system work, we had to find other ways to cope with the lack of the other 802.11s frames.

For instance, we had to use the beacons generated by the VAP in the *ap* mode for the neighbor discovery process, and the SSID has been used in place of the Mesh ID. Once a new neighbor is detected, a new *wds* VAP is set up and configured with the address of the neighbor. The bridge is connected to this VAP, but its port is kept blocked, and will be opened (i.e. set in the learning state) only when that link enters the HWMP tree (see again Figure 3). The quality of the link is then continuously monitored by both MPs and the measured values are exchanged with LLSA (Local Link State Announcement) frames to share the same metric value.

The feature of having all MPs know the complete list of associated STAs and their Proxy makes the management of the stations rather cumbersome, as the whole network should be flooded every time a STA associates or disassociates. In our implementation we reduced this burden by putting the knowledge of the whole set of associated STAs on the Portal. This is coherent with the assumption of the traffic going towards the portal. Clearly, all MPs can keep their own list of STAs using the information held in the PU messages they hear and/or forward.

In the implemented prototype broadcast frames may represent a serious weakness. Since every MP is connected to the others through a series of point-to-point links, broadcast can only be realized by transmitting the same frame over all interfaces, with a considerable waste of resources. Moreover, since the *wds* interfaces work with the legacy frames, the sequence number, meant to prevent the formation of loops, is not available.

We resolved to handle broadcast frames in a different way, yet keeping the prototype adherent to the behavior expected from a 802.11s compliant device. We divided broadcast into the uplink and downlink directions. Uplink broadcasts are those generated by the stations. Since they usually aim at getting some information from the Portal (e.g. ARP messages), they are converted by the proxy MAP into unicast frames, addressed to the MPP. Downlink broadcasts are instead generated by the Portal. Their addressing has not been changed, but their propagation has been constrained to the tree rooted at the Portal. Thanks to these measures, we could overcome the problem while retaining all network functionalities.

Note that the previous two solutions, i.e. concentrating on the Portal all the information on the associated stations and limiting the use of broadcast frames, have an "historical" background. They are in fact similar to the LAN emulation paradigm employed in wired networks to solve problems related to the mapping of address at different network layers (e.g. the ARP function). The 802.11s framework provides access to many clients which believe they are connected to a one-hop broadcast domain. The dimensions of the wired LAN and the wireless Mesh are also comparable, as the draft standard foresees to run networks of about 50 Mesh Points at most. Under this light, employing LAN-emulation-like solutions could therefore be a sensible and historically sound choice.

Proactive path selection has been implemented according to the draft specifications (see Section 2). However some routines have been slightly changed to improve the performance. For instance, each MP, after receiving a PRREQ, waits a short interval before re-broadcasting it to be sure to also receive the PRREQs that have followed a different path. The CSMA/CA mechanism does not guarantee that the first received PRREQ is the one coming from the shortest path. This trick increases the protocol stability, since the MP re-broadcasts only the PRREQ received from the neighbor closest to the root. As long as the MPs build the tree, they open the ports of the bridge connected to those VAPs whose link is part of the tree. Then, in the GRREP, instead of inserting the complete list of dependent nodes (as dictated by 802.11s), we put just the direct children of the MP. This achieves two goals: to reduce the protocol overhead and allow the MPs to have better knowledge of the topology of the Mesh.

The 802.11s draft gives MPs the possibility to change their route to the Portal, but does not specify under what conditions. In our framework, we resolved to have a MP change the path after receiving a number of consecutive PRREQs announcing a better metric. In this case, the MP sends a GRREP along the new path and a RERR along the old path, to inform all the parent MPs of the new tree topology. To increase the stability of the system, we made this number greater as the distance from the Portal is smaller. The reason is that a change in the path close to the MPP has an impact on the majority of the network (it conveys a lot of traffic and all the children must re-compute their path), while a change in a leaf MP remains pretty bounded to the MP itself.

3.4 Interaction and exploitation of ARP

The ARP mechanism offers a very good example to illustrate how our framework operates.

A STA opening a communication with the outer world (e.g. web browsing) begins with an ARP Request to find the MAC address associated to the IP of the intended destination. This broadcast frame is mapped to a unicast frame by the proxy MAP (with the STA address as the source) and sent towards the Portal along the HWMP tree. At each MP, the ports of the bridge forming the tree are in the learning state, thus they update the forwarding table with the STA MAC address. When the MPP casts the ARP Reply message to the STA, the bridges of all the MPs previously traversed by the ARP Request know to which port they should forward the frame. The same works in the opposite direction as well, i.e. when the MPP must broadcast an ARP Request following to an incoming connection towards a station in the Mesh.

If the connection is between two STAs internal to the Mesh, after the ARP request/reply exchange, the proxy MAP of the originating STA is able to recognize that the destination MAC, being different from the Portal MAC, is internal to the Mesh. Hence it can start the HWMP on-demand procedure to find a path to the destination.

3.5 Functionality tests

This Section describes some tests we carried out to verify the correct behavior of our prototype. In the testbed, the Portal was provided with proper routing capabilities, to separate the test network from the rest of the world at IP layer.

A simple test network (see Figure 4) was built with common laptop PCs: one PC set up as Portal, three as MAPs and one as a legacy 802.11 user station. The Portal was then connected to the department intranet, where a gateway provided Internet access.

Traffic on the Mesh was sniffed with the common Ethereal software [17] installed on another laptop used only as a monitor device. This allowed us to see all the frames exchanged between the MPs. Ethereal is able to recognize only the existing 802.11 frames, but is unaware of the changes brought by 802.11s. For instance, Figure 5 shows a typical LLSA frame, used by the MPs to exchange information about the state of the links (i.e. to build the link metric). The frame is recognized by Ethereal as an 802.11 Management frame, but its category is unknown, since it is one of the 802.11s modified frames (category is 5, i.e. the LLSA).

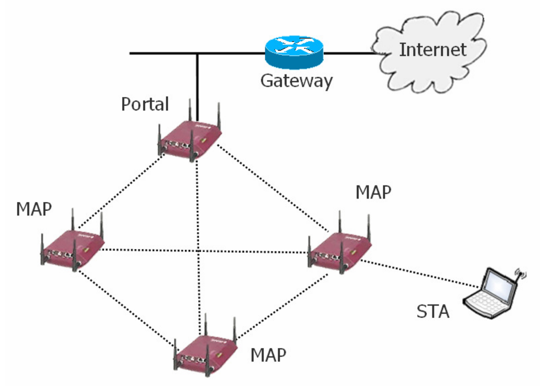
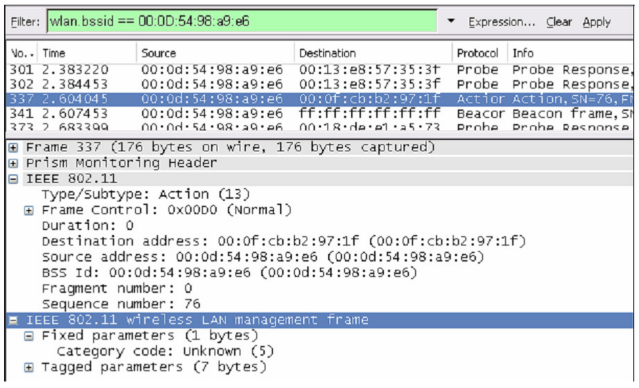


Figure 4. The testbed mesh network.


Filter: wlan.bssid == 00:0D:54:98:a9:e6
Expression... Clear Apply

```
No. Time      Source             Destination        Protocol Info
301 2.383220  00:0d:54:98:a9:e6  00:13:e8:57:35:3f  Probe    Probe Response,
302 2.384453  00:0d:54:98:a9:e6  00:13:e8:57:35:3f  Probe    Probe Response,
337 2.604045  00:0d:54:98:a9:e6  00:0f:cb:b2:97:1f  Actior   Action, SN=76, F
341 2.607453  00:0d:54:98:a9:e6  ff:ff:ff:ff:ff:ff  Beacor   Beacon frame, SI
373 2.683399  00:0d:54:98:a9:e6  00:18:de:e1:a5:73  Probe    Probe Response

⊞ Frame 337 (176 bytes on wire, 176 bytes captured)
⊞ Prism Monitoring Header
⊟ IEEE 802.11
    Type/Subtype: Action (13)
  ⊞ Frame Control: 0x00D0 (Normal)
    Duration: 0
    Destination address: 00:0f:cb:b2:97:1f (00:0f:cb:b2:97:1f)
    Source address: 00:0d:54:98:a9:e6 (00:0d:54:98:a9:e6)
    BSS Id: 00:0d:54:98:a9:e6 (00:0d:54:98:a9:e6)
    Fragment number: 0
    Sequence number: 76
⊟ IEEE 802.11 wireless LAN management frame
  ⊟ Fixed parameters (1 bytes)
      Category code: Unknown (5)
  ⊞ Tagged parameters (7 bytes)
```

Figure 5. Screenshot of Ethereal showing a LLSA frame.

To verify the behavior of our system, we first tried the formation of the HWMP tree. In the testbed, all stations are in direct radio visibility. To drive the MAPs to set up a particular topology, a custom module was built to give the mesh software pre-assigned link metrics. The formation of the tree is illustrated in Figure 6, (a) and (b). After turning the stations on, both A and B receives the PRREQ directly from the Portal, which becomes their parent node. MP C, receiving a PRREQ from both A and B, chooses B as its parent, since the announced metric is better. However, after a short time, A too changes its parent to B, as the metric to the Portal announced in the PRREQs received from B is better than the one of the direct connection to R.

The following are two excerpts of the Portal tree and "routing" tables that confirm the progress from the initial star topology to a tree-based configuration. Looking at the tree table, after a transient during which the Portal has not yet discovered its neighbors (at 2.9 s), all MPs becomes children of R (at time 14.7 s). Then C connects to B (at 31.9 s) and finally both A and C are children of B, or nephews of R (at time 34.9 s).

As for the Route table we can see how at the beginning A, B and C are all one hop away from R. Then, right after the final topology change (at time 36.1 s), B has become the next hop to reach both A and C. All the related parameters (e.g. metric) have also been updated.

```
TREE at time: 2.97051 s
=============================================
Parent: -> 00:0F:CB:B2:97:1F  (R)

TREE at time: 14.74586 s
=============================================
Parent: -> 00:0F:CB:B2:97:1F  (R)
Child: -> 00:0D:54:98:A9:E6  (B)
Child: -> 00:0D:54:99:50:F4  (A)
Child: -> 00:0F:CB:B2:97:EB  (C)

TREE at time: 31.95626 s
=============================================
Parent: -> 00:0F:CB:B2:97:1F  (R)
Child: -> 00:0D:54:99:50:F4  (A)
Child: -> 00:0D:54:98:A9:E6  (B)
Nephew: -> 00:0F:CB:B2:97:EB  (C)

TREE at time: 34.95971 s
=============================================
Parent: -> 00:0F:CB:B2:97:1F  (R)
Child: -> 00:0D:54:98:A9:E6  (B)
Nephew: -> 00:0D:54:99:50:F4  (A)
Nephew: -> 00:0F:CB:B2:97:EB  (C)


ROUTE TABLE at time: 4.97784 s
=============================================
DESTINATION: 00:0D:54:98:A9:E6  (B)
NEXT HOP: 00:0D:54:98:A9:E6  (B)
HOP COUNT: 1
PATH METRIC: 10
Flag NEIGHBOR: 1
---
DESTINATION: 00:0D:54:99:50:F4  (A)
NEXT HOP: 00:0D:54:99:50:F4  (A)
HOP COUNT: 1
PATH METRIC: 50
Flag NEIGHBOR: 1
---
DESTINATION: 00:0F:CB:B2:97:EB  (C)
NEXT HOP: 00:0F:CB:B2:97:EB  (C)
HOP COUNT: 1
PATH METRIC: 100
Flag NEIGHBOR: 1

ROUTE TABLE at time: 36.10375 s
=============================================
DESTINATION: 00:0D:54:98:A9:E6  (B)
NEXT HOP: 00:0D:54:98:A9:E6  (B)
HOP COUNT: 1
PATH METRIC: 10
Flag NEIGHBOR: 1
---
DESTINATION: 00:0D:54:99:50:F4  (A)
NEXT HOP: 00:0D:54:98:A9:E6  (B)
HOP COUNT: 2
PATH METRIC: 20
Flag NEIGHBOR: 0
---
DESTINATION: 00:0F:CB:B2:97:EB  (C)
NEXT HOP: 00:0D:54:98:A9:E6  (B)
HOP COUNT: 2
PATH METRIC: 20
Flag NEIGHBOR: 0
```

Subsequently, we abruptly turned B off to check the capability of the Mesh to recover after a node failure. Since MPs A and C no longer receive the LLSA messages from B, after a timeout they delete station B from their neighbor list and also remove the path to the Portal. Several options are now open to allow the two MPs to find a new path. The first 802.11s draft suggested an active on-demand search, but this course of action has been removed in the successive updates. In our implementation we just let the MPs wait for the next PRREQ to come. Though successful, this strategy may be rather slow, as it depends on the PRREQ broadcast interval. A smarter alternative could consist in keeping a backup path to the root, for example related to the second best metric announced. However, this approach does not guarantee an immediate recovery, as in bigger networks it might happen that the backup path also includes the crashed station.

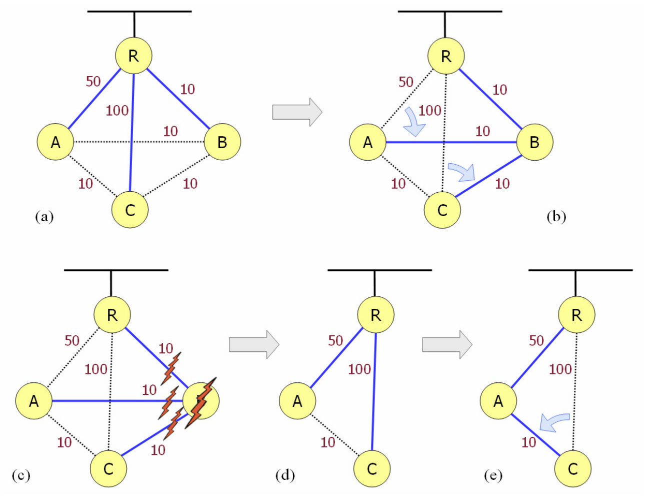
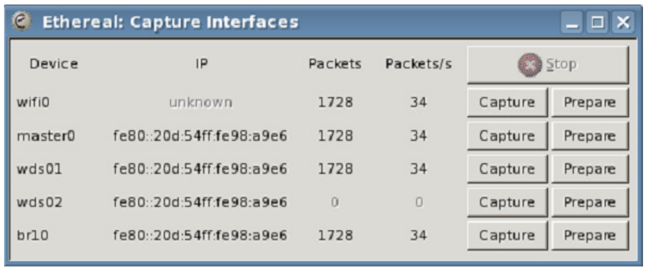


Figure 6. Topology formation at network start-up (a and b) and following to a node failure (c, d and e). Link metrics are also reported.

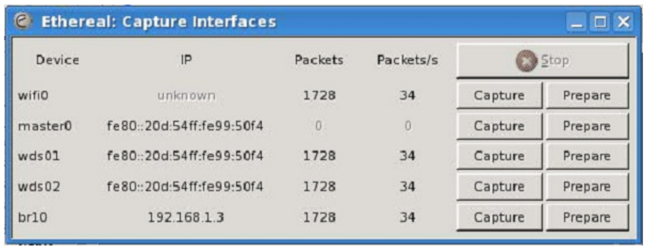
Figure 6 (c, d and e) shows the evolution of our test network. At first, both C and A establish a direct one-hop path to the Portal (d), but as long as the new PRREQs announce a better metric, C switches to the more convenient two-hop path that passes through

A (e). The final paths are in accordance with the imposed link metrics.

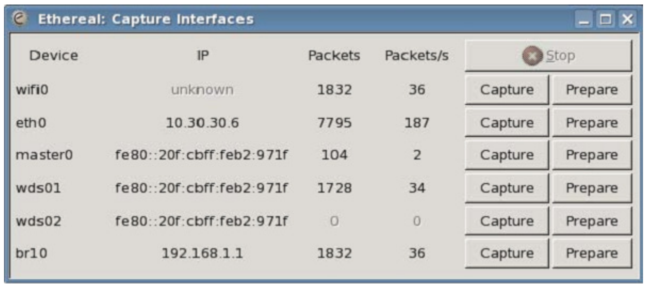
The last functionality to test is the support of the client stations. We connected a station to MAP C, as already seen in Figure 3. After the station completed the association procedure, C sent the PU frame to the root to inform it of the new station, which can now use the Mesh to access the Internet. We let the user do some web browsing and analyzed the traffic on the links. Figure 7 reports the data (IP) traffic sniffed at the three MPs. Frames originated or destined to the station pass through the master0 interface of MAP C, which is set in the *ap* mode and where the station is associated. The same amount of frames then pass through the bridge (br10) and then enters (or exits) the Mesh from wds01, which is connected to MAP A. wifi0 is the physical device and clearly measures the same values. Then, in MAP A, both *wds* interfaces handle the same number of frames, to confirm that A acts as a relay MP between C and R. Finally, in the Portal, frames only pass through interface wds01, which is the bridge port connected to MAP A. The other interface, wds02, is blocked by Ebtables in accordance to the tree topology, and no frames pass through it. At last, eth0 is the LAN interface connected to the Internet.



Device	IP	Packets	Packets/s
wifi0	unknown	1728	34
master0	fe80::20d:54ff:fe98:a9e6	1728	34
wds01	fe80::20d:54ff:fe98:a9e6	1728	34
wds02	fe80::20d:54ff:fe98:a9e6	0	0
br10	fe80::20d:54ff:fe98:a9e6	1728	34



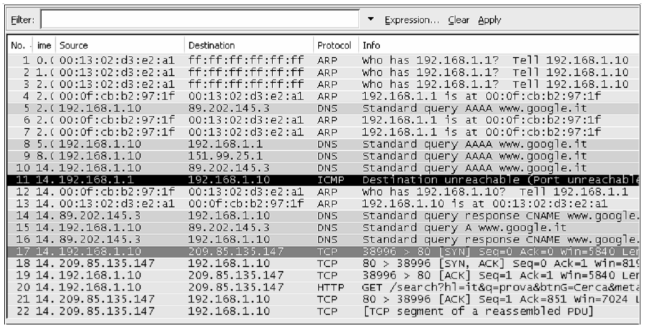
Device	IP	Packets	Packets/s
wifi0	unknown	1728	34
master0	fe80::20d:54ff:fe99:50f4	0	0
wds01	fe80::20d:54ff:fe99:50f4	1728	34
wds02	fe80::20d:54ff:fe99:50f4	1728	34
br10	192.168.1.3	1728	34



Device	IP	Packets	Packets/s
wifi0	unknown	1832	36
eth0	10.30.30.6	7795	187
master0	fe80::20f:cbff:feb2:971f	104	2
wds01	fe80::20f:cbff:feb2:971f	1728	34
wds02	fe80::20f:cbff:feb2:971f	0	0
br10	192.168.1.1	1832	36

Figure 7. Screenshots of the data traffic through MP C (top), MP A (middle) and the Portal (bottom).

Finally, a snapshot of the client traffic, taken directly on the user station, is shown in Figure 8. Is it possible to distinguish the ARP procedure, in which the reply message (line 4) comes directly from the Portal (whose MAC address is 00:0F:CB:B2:97:1F). This confirms that the station is unaware of the existence of the Mesh and believes the MPP is just one hop away. Then, we can also note how a broadcast frame sent by the Portal to find the station MAC address (during the DNS procedure) arrives at the station as a unicast frame (line 12). This is an evidence that the broadcast reduction strategy described in Section 3.3 works properly.



```
No. ime Source            Destination       Protocol Info
  1 0.( 00:13:02:d3:e2:a1 ff:ff:ff:ff:ff:ff ARP      Who has 192.168.1.1?  Tell 192.168.1.10
  2 1.( 00:13:02:d3:e2:a1 ff:ff:ff:ff:ff:ff ARP      Who has 192.168.1.1?  Tell 192.168.1.10
  3 2.( 00:13:02:d3:e2:a1 ff:ff:ff:ff:ff:ff ARP      Who has 192.168.1.1?  Tell 192.168.1.10
  4 2.( 00:0f:cb:b2:97:1f 00:13:02:d3:e2:a1 ARP      192.168.1.1 is at 00:0f:cb:b2:97:1f
  5 2.( 192.168.1.10      89.202.145.3      DNS      Standard query AAAA www.google.it
  6 2.( 00:0f:cb:b2:97:1f 00:13:02:d3:e2:a1 ARP      192.168.1.1 is at 00:0f:cb:b2:97:1f
  7 2.( 00:0f:cb:b2:97:1f 00:13:02:d3:e2:a1 ARP      192.168.1.1 is at 00:0f:cb:b2:97:1f
  8 5.( 192.168.1.10      192.168.1.1       DNS      Standard query AAAA www.google.it
  9 8.( 192.168.1.10      151.99.25.1       DNS      Standard query AAAA www.google.it
 10 14. 192.168.1.10      89.202.145.3      DNS      Standard query AAAA www.google.it
 11 14. 192.168.1.1       192.168.1.10      ICMP     Destination unreachable (Port unreachabl
 12 14. 00:0f:cb:b2:97:1f 00:13:02:d3:e2:a1 ARP      Who has 192.168.1.10?  Tell 192.168.1.1
 13 14. 00:13:02:d3:e2:a1 00:0f:cb:b2:97:1f ARP      192.168.1.10 is at 00:13:02:d3:e2:a1
 14 14. 89.202.145.3      192.168.1.10      DNS      Standard query response CNAME www.google.
 15 14. 192.168.1.10      89.202.145.3      DNS      Standard query A www.google.it
 16 14. 89.202.145.3      192.168.1.10      DNS      Standard query response CNAME www.google.
 17 14. 192.168.1.10      209.85.135.147    TCP      38996 > 80 [SYN] Seq=0 Ack=0 Win=5840 Ler
 18 14. 209.85.135.147    192.168.1.10      TCP      80 > 38996 [SYN, ACK] Seq=0 Ack=1 Win=819
 19 14. 192.168.1.10      209.85.135.147    TCP      38996 > 80 [ACK] Seq=1 Ack=1 Win=5840 Ler
 20 14. 192.168.1.10      209.85.135.147    HTTP     GET /search?hl=it&q=prova&btnG=Cerca&meta
 21 14. 209.85.135.147    192.168.1.10      TCP      80 > 38996 [ACK] Seq=1 Ack=851 Win=7024 L
 22 14. 209.85.135.147    192.168.1.10      TCP      [TCP segment of a reassembled PDU]
```

Figure 8. Screenshot of Ethereal showing the beginning of a TCP connection.

4. REMARKS

During our tests, we often noted that the airtime metric suggested by the 802.11s draft may rise a serious stability problem if the path change is easily allowed. Due to the random nature of the medium access method, the metric of the links carrying more traffic can degrade rather quickly. As a result, a MP receiving a PRREQ announcing a better metric (usually from an unloaded link) changes its path and forwards all the traffic on the new route (an example is shown in Figure 9). Shortly, the new route will suffer the same metric degradation of the old path, which is now unloaded and whose metric has become preferable. The MP is therefore bound to change path again, thus creating an oscillating phenomenon.

The support of STA mobility is another serious issue. Apart from the association/disassociation procedures, which are unchanged and out of the scope of the draft standard, 802.11s plans that all MPs are informed of the movement of every STA. This clearly generates a lot of control traffic, which is actually unnecessary. Furthermore, this solution does not scale very well. In our testbed we proved that we can efficiently let this information be stored only in the Portal.

Two situations can take place. If a STA is communicating with an external destination, since all MAPs have a path to the MPP, the new Proxy MAP automatically knows where to send the frames generated by the STA, while the bridges of all the MPs along the tree automatically learn the new port mapping for the STA, as already explained in Sections 3.2 and 3.4 (we remind that the driver places the station MAC address on the Ethernet frame it passes to the bridge).

If a STA is communicating with another STA internal to the Mesh, a new on-demand path must be set up. If the PU is spread across the whole network, the MAP can immediately start the on-demand procedure. Conversely, the MAP must previously ask the Portal who is the new Proxy. In a comparison, the two techniques are not very different, as they produce a similar amount of control traffic. The one suggested by the 802.11s draft concentrates it into a single PU flood, while our solution splits it in two more constrained parts.

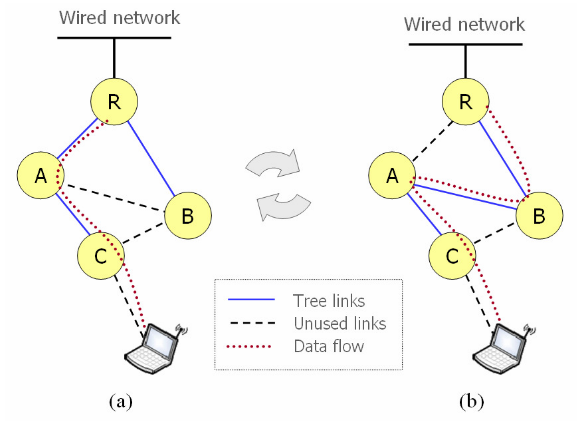


Figure 9. Instability of the paths.

Finally, it is worth noting that the 802.11s draft avoids dealing with ARP, reckoning it is an exclusively level three issue. However, in the implementation phase, we realized how it may be effectively used to improve the performance of the Mesh. For internal destinations, the ARP works much like the on-demand scheme of HWMP. Therefore, two consecutive floods are executed: at first, to find the MAC address associated to a given IP, and then to build the HWMP path. The two actions could be combined into a single one, as we explained in Section 3.4. An alternative may also be the one suggested in [9], which combines the ARP request and the RREQ into a single frame. However, apart from being more suited for on-demand path selection, this approach has the drawback of mixing an upper layer data message (what the ARP actually is) with a level two management frame. Hence it is not clear who and how is going to handle it.

5. CONCLUSIONS

The paper presented a prototype of a Wireless Mesh node as much compliant as possible with the IEEE 802.11s draft. The implementation was realized using common tools, a portable Linux-based PC and open source software. A software framework was implemented on top, and partly integrated with the network interface card driver. This framework is flexibly structured in a series of modules and can be easily upgraded to the newer editions of the draft (as soon as they are available). The functioning of our solution was then successfully verified in a testbed network.

From this work, some useful considerations can be drawn. At first, the availability of a prototype 802.11s network allows the researcher to readily test the features and the amendments to the draft as soon as they are proposed, thus returning immediate and significant feedbacks on their effectiveness. This is an important milestone, as simulation trials, though very useful, often do not give answers on the actual feasibility of the tested feature.

Secondly, our work pointed out a couple of flaws. At first, the suggested airtime metric, used to set up the paths to the Portal, may cause severe instability problems. A suitable tuning of the path selection parameters is therefore advisable, but seeking other metrics is probably an even better option. Then, spreading the information on end-user stations across the whole network is a cumbersome task that can be avoided without detriment to the Mesh operation. We proved that, when a HWMP tree is present, concentrating this information only at the Portal is a practical and effective solution.

Finally, though the draft does not deal with upper layer protocols, it might be useful to exploit the interaction with common mechanisms like ARP to save network resources. Having much in common with the on-demand path selection procedure, we can use it to set up paths towards the client stations thus avoiding broadcasting the successive RREQ as imparted by the 802.11s draft.

6. ACKNOWLEDGMENTS

This work was supported by the Italian MIUR-FIRB project "Integrated System for Emergency" (InSyEme), under grant number RBIP063BPH. The authors would like to thank Ivan Muccini for his significant help in the implementation of the software framework and the realization of the test network.

7. REFERENCES

[1] IEEE Std. 802.11-2007, "Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specifications", IEEE Computer Society, Jun. 2007.

[2] Ian F. Akyildiz, Xudong Wang, Weilin Wang, "Wireless mesh networks: a survey", Computer Networks, Volume 47, Issue 4, March 2005.

[3] Motorola Wireless Broadband Mesh Networks, available at: http://www.motorola.com/mesh.

[4] Belair Networks, http://www.belairnetworks.com.

[5] Tropos Netwrorks, http://www.troposnetworks.com.

[6] IEEE P802.11 Task Group S, IEEE Unapproved draft P802.11s/D1.0, March 2007, and successive redlines.

[7] IEEE P802.11 Task Group S – Status of Project IEEE 802.11s – Mesh Networking, at: http://grouper.ieee.org/groups/802/11/Reports/tgs_update.htm.

[8] Qiang Shen, Xuming Fang, "A Multi-metric AODV Routing in IEEE 802.11s", International Conference on Communication Technology (ICCT), Nov. 2006.

[9] Sung-Hee Lee, Young-Bae Ko, "An Efficient Multi-hop ARP Scheme for Wireless LAN based Mesh Networks", First Workshop on Operator-Assisted (Wireless Mesh) Community Networks (OpComm), Sep. 2006.

[10] Hiraku Okada, Kenichi Mase, Masanori Nozaki, Bing Zhang, "A Low-Overhead Handling Scheme of STA Association Information for IEEE 802.11s", IEEE 18th International Symposium on Personal, Indoor and Mobile Radio Communications (PIMRC), Sept. 2007.

[11] D. J. Shyy, "Military Usage Scenario and IEEE 802.11s Mesh Networking Standard", Military Communications Conference (MILCOM), Oct. 2006.

[12] Guido R. Hiertz, Sebastian Max, Rui Zhao, Dee Denteneer, Lars Berlemann, "Principles of IEEE 802.11s", Proceedings

of 16th International Conference on Computer Communications and Networks, (ICCCN), Aug. 2007.

[13] C. Perkins, E. Belding-Royer, S. Das, “Ad hoc On-Demand Distance Vector (AODV) Routing”, IETF RFC 3561, July 2003.

[14] D. S. J. De Couto, D. Aguayo, J. Bicket, R. Morris, “A High-Throughput Path Metric for Multi-Hop Wireless Routing”, ACM Mobicom, 2003.

[15] Multiband Atheros Driver for Wi-Fi, at: http://madwifi.org/

[16] Ebtables, available at: http://ebtables.sourceforge.net.

[17] Ethereal, available at: http://www.ethereal.com.