An Enhanced Bridged-Based Multi-hop Wireless Network Implementation

Stefano Maurina, John Fitzpatrick, Laurentiu Trifan and Liam Murphy
Performance Engineering Lab,
School of Computer Science and Informatics,
University College Dublin,
Dublin, Ireland.
Email: stefano.maurina@gmail.com

***Abstract*—In this paper an enhanced Layer-2 multi-hop wireless network implementation for Infrastructure based Wireless Mesh Networks is presented. This work combines the flexibility of Layer-2 Wireless Bridging with the dynamic self-configuring capabilities of MANET routing. The main contribution of this paper is an investigation of the issues encountered when applying a pure bridging based solution to wireless multi-hop networks and the development of several mechanisms to overcome these problems. This work was implemented and deployed in a real testbed environment using Routerboard hardware and utilising a number of open-source network tools in accordance with the needs of our platform. The developed testbed incorporates self-healing and self-configuration features without requiring a traditional MANET routing protocol. Instead the 802.11 beacon frames sent by the Access Points were extended with link information to allow optimal construction of the mesh topology. Results are presented which demonstrate the automated topology construction mechanism. Further results also show the enhancements made to the normal 802.11 Layer-2 mobility mechanism.**

***Index Terms*—Multi-hop Wireless Networks, Wireless Distribution System (WDS), Layer-2 Wireless Bridging, MANET routing, Beacon, Handover**

I. Introduction

WIRELESS Mesh Networks are gaining increasing attention as a low cost approach for providing wireless Internet access in a similar fashion to IEEE 802.11 based Access Points (APs). However, unlike traditional wireless APs, Wireless Mesh Networks (WMNs) allows wireless network coverage to be extended without requiring the installation of expensive cabling. Although a significant amount of work has been done in the WMN domain, much of the research has been based upon simulation models. Although these can provide a good basis for the development of new algorithms and mechanisms, it is essential that an understanding of the issues relating to the development and implementation of real WMNs is gained.

There are primarily two approaches for building a WMN [1]. The first is based on IP routing while the second on Wireless Bridging. IP routing based WMNs utilise Mobile Ad Hoc Network (MANET) routing protocols. There are two main types of MANET routing protocols, reactive and proactive approaches. A lot of work has been carried out in this area with the two most successful solutions being Ad hoc On-Demand Distance Vector Routing (AODV) [2] and Optimized Link State Routing Protocol (OLSR) [3].

These protocols are implemented as routing daemons that run on top of Layer-3 (IP layer) and packets are routed toward the destination solely based on their IP address. The disadvantage is that other Layer-3 protocols, such as DHCP, cannot be used or they need a different, tailored version of the MANET protocol, like AODV6 [4] for IPv6. However, the main advantage of these approaches is that the connectivity of the nodes and the topology of the network are constantly monitored and routes are determined dynamically. MANET protocols are therefore robust to changes in the network, self-configuring and self-healing.

The Wireless Bridging based solutions use the Wireless Distribution System (WDS) [5] which provides forwarding functionality for extending the range of a wireless network by allowing APs to act as repeaters. WDS is the Layer-2 bridging alternative to routing and a WMN can be built simply by using bridging. However, this solution comes with pros and cons.

A. Wireless Bridging Pros

The forwarding performed at Layer-2 offers great flexibility since packets are routed according to the destination MAC address regardless of the upper layer protocol. This means that it is transparent to higher layer protocols, such as DHCP and IPv6, and software can be used without any modification. The implementation of such solutions is easier since it just requires the set up of each AP to support the 4-addressing scheme format (WDS).

Furthermore, since frames are bridged at Layer-2 there is no special mechanisms required to develop in order to get basic mobility support. Other advantages are that it eliminates the overhead introduced by a routing protocol, the computational requirement on an embedded system and the impact of periodic link-quality metric updates which have to cope with unreliable wireless links.

B. Wireless Bridging Cons

However, since WDS bridging is usually static, with the neighbour APs' MAC addresses entered manually in each AP, the dynamic self-configuration advantages of MANET protocols are lost. Also, when using only one interface, all the APs

Digital Object Identifier: 10.4108/ICST.WICON2010.8510
http://dx.doi.org/10.4108/ICST.WICON2010.8510

have to communicate in the same wireless channel causing interference and contention issues with obvious performances degradation. Furthermore, the standard AP-to-AP WDS format requires that all of the wireless cards in the backbone operate in AP mode. This means that a lot of overhead is generated by IEEE 802.11 Management frames.

WDS also inherits many of the limitations of bridging. Bridges use broadcast mechanisms until they learn a route to a host and in the case of a broadcast protocol, such as ARP and DHCP, the network is always being flooded. For this reason they are slow to converge and do not scale well. It is also worth noting that in a Layer-2 solution clients are no longer transparent to the Mesh Nodes (MNs) since an end-user is not known to a bridge until the client sends a packet to it.

Finally, the AP-to-AP topology implies that all APs within the same radio range connect with one another. As a consequence the mesh network obtained is highly likely to incorporate loops. This would not be a problem if a routing protocol was being used but bridges cannot be used in loop topologies unless Spanning Tree Protocol (STP) is used. However, STP leads to a loop-free topology by disabling some ports and enabling others according to its own algorithm and link costs without considering any link quality metrics which would be more suitable to the wireless medium.

The remainder of the paper is structured as follows; Section II presents a brief overview of related work. Section III introduces the main aspects of the proposed solution and provides a detailed description of the implementation. In Section IV an evaluation of the obtained results is presented. Finally, the paper is concluded in Section V.

II. Related Work

Layer-3 routing based Mesh Networks - Nowadays, most WMN implementations are experimental and run by government agencies, non-profit organizations, municipalities, and research institutions. Most of these implementations are based on open-source technologies and make use of a Layer-3 MANET routing protocol. An example of these experimental trials is the Freifunk Project [6]. Packets are routed based on their destination IP address using OLSR. The protocol is implemented as a routing daemon that operates on top of Layer-3.

Layer-2 routing based Mesh Networks - Ad-hoc Wireless Distribution Service (AWDS) [1] is an implementation of the OLSR routing protocol operating at Layer-2. It is implemented as a Unix/Linux daemon and operates completely in user space. For communicating with applications, the AWDS daemon creates a virtual ethernet interface. All packets sent to the virtual interface are received by the AWDS daemon and routed to other stations according to the MAC address of the wireless network card. Bridging can be used for integrating this wireless network into an existing Infrastructure.

Layer-2 bridging based Mesh Networks - These type of WMNs use the wireless bridging capabilities offered by WDS. WDS is standardized by IEEE-802.11 and is implemented by many manufacturers [7], [8]. Nevertheless, different products are rarely compatible with each other since some parts of the standard are not fully specified. The main purpose of WDS is to provide wireless multi-hop extension to the wired Distribution System (DS) allowing APs to act as wireless repeaters. WDS allows packets to pass from one wireless interface to another (on a different MN), just as if the wireless devices were ports on a wired ethernet switch.

III. System Architecture and Implementation

The solution developed in this work was done with a focus of providing end-user clients with the ability to connect to a wired network (in particular the Internet) and not to provide intra-mesh communication in which clients can communicate directly with each other. As a consequence all traffic flows are aggregated and forwarded either to or from the Gateway (GW) node. Based on this assumption several features and enhancements were developed to overcome the issues normally encountered when applying a pure bridging based solution to wireless multi-hop networks (cf. Section I-B).

A. Wireless Bridging Enhancement Solutions

It was therefore decided to use WDS over Infrastructure mode (AP-STA communication). This choice was taken for three main reasons. Firstly, in standard AP-to-AP mode, WDS bridging is static with the neighbour APs' MAC addresses entered manually in each AP. When using WDS over Infrastructure mode an interface operating in STA mode can connect dynamically to an AP without the need to know their respective MAC addresses but is still capable of using the 4-address format (WDS). Secondly, the use of some interfaces operating in STA mode instead of AP mode leads to a reduction in the overhead generated by IEEE 802.11 Management frames. Finally, the use of Infrastructure mode allows for the automatic construction of a loop-free topology. This eliminates the requirement of having to use STP to prevent loops which would interfere with our chosen link quality metrics for optimal route computation.

Unlike a router, a bridge does not limit the scope of a broadcast message. This results in uncontrolled flooding which would prevent the mesh network from scaling up to cover larger geographical areas. Based on the previous assumptions it is sufficient for end-users to only have the capability to communicate with the GW. Only DHCP discovery and ARP request messages need to be supported for this purpose while all other broadcast messages can be dropped. Besides, since these two message types are always directed to the GW, as soon as they are received by the Mesh Bridge (MB) to which the client is attached, a MAC Destination NAT (DNAT) is performed which changes the broadcast address to the MAC address of the GW and are then unicast forwarded to the GW.

As frames are bridged at Layer-2, there is no special behavior to develop in order to have some basic mobility support. Nevertheless, since an end-user is not known to a bridge until the client sends a packet to it, client-transparency is not guaranteed and there will be a re-bridging delay following the association to the new AP. To reduce this latency an ARP-Proxy and a re-bridging mechanism were developed.

Digital Object Identifier: 10.4108/ICST.WICON2010.8510
http://dx.doi.org/10.4108/ICST.WICON2010.8510

One of the key aspects of WMNs are the link-quality metrics used for the selection of the best routes. These metrics are usually computed using periodic messages exchanged by a routing protocol. In this work the normal 802.11 beacon frames sent by an AP are enhanced to allow for the computation of a link metric without the need for signalling from a routing protocol. Furthermore, since the focus of this work is on providing fixed wireless coverage, the MNs are static and therefore the routes are not very dynamic unlike what can be expected in a MANET.

In order to increase the network capacity and minimise interference each MN is equipped with multiple 802.11 radios. Specifically, one in 11g mode to provide connectivity to end-users and two in 11a mode to provide intra-mesh communication between the MNs. By utilising multiple 802.11 radios in each mesh node, our approach offers higher levels of capacity and redundancy than existing single radio solutions.

A mesh-daemon was also developed to provide automated self-configuration similar to that provided by a MANET routing protocol. The self-healing capability is provided by the daemon *ifplugd* [9], which is configured to trigger an immediate response from the mesh-daemon upon connection/deconnection of a link.

B. Overview of the Multi-hop Wireless Network

An example of the multi-hop network architecture is shown in Figure 1. As can be seen the Mesh Portal (MP) and MBs form a multi-hop wireless backbone; the MNs are static nodes which act as relays for the mobile clients. Each MN is equipped with three IEEE 802.11 Network Interface Cards (NICs), one of which operates in 11g/b Infrastructure mode and provides an AP connection to end-users. The other two antennas on each MN are used for interconnection with other MNs, this allows for the building of the multi-hop wireless backbone. In order to mitigate interference with the clients these two radios operate in the 11a frequency band.

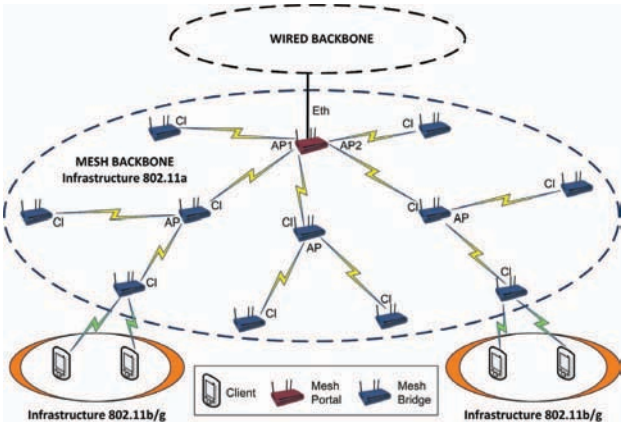

Fig. 1. Overview of our Mesh Network Topology.

C. Overview of Mesh Nodes

All the MNs in our testbed have both the same hardware and software. Each has an ethernet port and three wireless interfaces; the wireless interfaces are bridged with each other and are used for interconnection among the MNs and to provide access to the mesh to end-users.

The Ethernet interface of certain nodes is used to connect the mesh network to an external wired network. Based upon whether or not a MN has a wired connection to an external network it will play a different role in the network; it will behave as either a MP or a MB.

1) MeshPortal (MP): A MP is a MN which has a direct access to a wired network through the ethernet port; the main purpose of which is to route traffic between the mesh and external network. Essentially, a MP is a MN which acts as both a bridge and a router where the routed interfaces are the *eth0* ethernet port and the bridge logical device *br0*. The wireless interfaces are enslaved to the bridge *(br0)* and allow connections with the other MNs.

In a MP the wireless interfaces are configured as in Table I. *AP(users)* acts as a normal AP in mode 11g/b for the connection of the end-users, while *APs(mesh)* are APs in mode 11a supporting the 4 addressing scheme (WDS) which are used to extend the MeshBackbone. These wireless interfaces are enslaved to the same Linux-based bridge *br0*. Furthermore, a MP runs a DHCP server to assign the IP addresses to the clients connected to the MeshBackbone and a NAT to share the connection among them.

2) MeshBridge (MB): A MB is a MN which does not have any wired connectivity, rather it obtains access to external networks by connecting wirelessly to other MNs (both MP or other MBs). It is the wireless interconnection between all the MNs that builds the multi-hop wireless network backhaul. Each MB acts as a relay station to forward the traffic from clients and other MBs to and from the MP. As it happens on a MP the three wireless interfaces are enslaved to the same Linux-based bridge *br0*.

In a MB the wireless interfaces are configured as in Table I. *AP(users)* is an 11b/g AP in the 2.4Ghz band used for connection of the end-users, *Client(mesh)* is an interface operating in station mode in the 5Ghz band and supporting the 4 addressing scheme WDS; it is used for connecting to an *AP(mesh)* on another MN. *AP(mesh)* is an 11a AP operating in the 5Ghz band and is used to extend the MeshBackbone by allowing other MNs to connect.

TABLE I
WIRELESS INTERFACES CONFIGURATION

MeshNode	ATH0	ATH1	ATH2
MP	AP (users)	AP (mesh)	AP (mesh)
MB	AP (users)	AP (mesh)	Client (mesh)

D. Testbed Architecture

In this section we introduce the main hardware and software components utilised to build our solution.

1) MN Hardware: The hardware in each MN is based upon a *MikroTik RouterBOARD 532A* [10]. This device comes with a MIPS32 little-endian based 400MHz embedded processor,

Digital Object Identifier: 10.4108/ICST.WICON2010.8510
http://dx.doi.org/10.4108/ICST.WICON2010.8510

64MB of onboard DDR memory, two MiniPCI slots, a CompactFlash interface and three Fast Ethernet ports. Since we need three wireless interfaces we extended the main board by using the *RouterBOARD 502* daughterboard. This attaches to the main board of the *RB532A* adding two more MiniPCI slots. The wireless interface cards used are *MikroTik R52* 802.11abg miniPCI cards. These cards are based on the *Atheros AR5414* chipset and operate in both the 2.312-2.499GHz and 4.920-6.100GHz frequency ranges.

2) Software:

a) Kernel-land: The Operating System (OS) used for the MNs is a Debian Linux [11] distribution with kernel version *2.6.21*. The standard kernel [12] required several patches to function correctly on the MN hardware being used. The patched OS is installed and run on a 1 Gigabyte Compact Flash Card.

The open-source MadWifi driver [13] was chosen to drive our Atheros-based wireless cards. As with the OS, the original driver required several patches to add new features which are described next in this paper.

The Layer-2 firewall *ebtables* [14] has been used and extended with new modules to implement the main features of our mesh network solution.

ebtables and the patched Madwifi driver are the two main components that allow us to overcome the issues of a pure bridging based Layer-2 solution.

b) User-space: The main component developed in user-space is a mesh-daemon written in python. This daemon controls and coordinates the mesh-related kernel modules and user-space tools which have been developed to realise the proposed mesh network solution.

One of the main tasks of the mesh-daemon is the signal handling mechanism which enable a MN to quickly react to the connection or disconnection of a wireless interface. Indeed the mesh-daemon updates the MN state every two minutes. Nevertheless this is not often enough to react quickly to a change in the connection of a wireless interface. For this reason the *Client(mesh)* interface is monitored using the tool *ifplugd* [9]. *ifplugd* is primarily used to automatically configure an ethernet device when a cable is attached, and to automatically reset it if the cable is unplugged. Fortunately it also offers support for wireless interfaces and using link beat detection APIs is able to perform carrier detection.

E. Beacon Enhancement & Propogation for Automatic Topology Construction

Beacon frames are part of the Management frames defined by the IEEE 802.11 standard. Management frames present the same MAC header regardless of their subtype while the specific information of each subtype is usually carried in the Frame Body (cf. Figure 2).

Two types of fields are defined in a Frame Body: fixed-length fields appropriately called *fixed fields* and variable-length fields named *information elements* [15]. Information elements are encoded as a Type-Length-Value (TLV); they offer great flexibility since new fields can be defined without compromising the compatibility with older implementations which can easily skip these new elements.

The Madwifi driver is patched to add a new information element (*mesh_beacon*) to the Frame Body of the beacon frames. This modification applies to the *APs(mesh)* and allows to carry additional information on the backbone network. This information is used by the *Client(mesh)* interface during the connection process to decide which of the available *AP(mesh)* interfaces it should associate with.

A new structure has been added to the original 802.11 beacon frame structure which contains four fields. These are, a gateway identifier, the number of hops to the gateway, the number of clients connected along the path to the gateway and the number of clients connected locally (cf. Figure 2). In the current implementation only the first two of these fields are used for selecting a backbone access point to associate with (cf. Section III-F1). At the receiving node of a modified beacon frame, the *mesh_beacon* information is obtained by the user-space mesh-daemon. The communication between this user-space daemon and the kernel-land madwifi driver is carried out through the *sysctl*(*/proc/sys* directory) interface. This interface allows user space applications to read and write the kernel variables exported by *sysctl*.

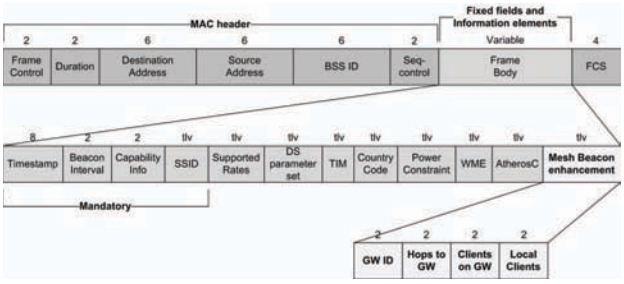

Fig. 2. Modified Beacon frame.

Upon scanning the network with the *Client(mesh)* interface, a MB chooses the best *AP(mesh)* to connect to according to the metric described in Section III-F1. Once the MB has associated with the *AP(mesh)* it must also begin broadcasting a modified beacon frame to allow other mesh nodes to connect to it thereby extending the network. Prior to broadcasting the beacon frames the MB first updates the *mesh_beacon* variables, for example it must increment the hop count by one.

F. Layer-2 Wireless Bridging

In this section we describe the main components of our Layer-2 Wireless Bridging solution.

1) Routing Metric for Implementation: The information carried in the beacon extension is used to compute the routing metric. During the bootstrap process a MB scans using a *Client(mesh)* interface for *APs(mesh)* to connect to. The AP connection decision is based on both the number of hops to the gateway and the Signal to Noise Ratio (SNR) of the *APs(mesh)*. Furthermore, an SNR threshold has been introduced to prevent the MB from connecting to an AP which is closer to the GW in terms of number of hops, but presents a very low SNR. The *AP(mesh)* with the minimum number of hops and an SNR greater than the threshold is chosen. In the

Digital Object Identifier: 10.4108/ICST.WICON2010.8510
http://dx.doi.org/10.4108/ICST.WICON2010.8510

case where there is more than one AP with the same minimum number of hops, the one with the highest SNR is selected. If none of the available APs exceeds the SNR threshold then the threshold value is reduced and the decision process is repeated.

2) Linux Bridging: In order to bridge together the wireless interfaces of each MN, the bridge functionality of the Linux kernel is used [16]. The Linux bridge code implements a subset of the ANSI/IEEE 802.1d standard.

brctl is the user-space utility to configure a Linux bridge. *brctl* allows to create a bridge device, enslave NICs to it, and configure the bridge parameters. To talk to the kernel *brctl* uses the `ioctl` interface for a subset of commands and *sysfs* interface for the others.

There are two main differences between a Linux bridge and a pure hardware switch. The first is the capability of a Linux bridge to filter and shape traffic using *Ethernet Bridge Tables (ebtables)*; the *ebtables* program is a filtering tool for Linux bridge based firewalls. The second is the ability to assign an IP address to the virtual interface formed by the bridge. All NICs in the bridge can listen and respond to datagrams destined to this IP. This is the only exception to transparency and is useful for the remote management of the bridge.

3) Ebtables based Firewall: ebtables is a framework that allows filtering and mangling of ethernet frames at Layer-2 on a Linux bridge. Combined with the *brctl* tool it allows an OSI Layer-2 bridge firewall to be built. It also provides some basic IP filtering possibilities. Therefore, *ebtables* provides extra capabilities that are not available using Netfilter; specifically the bridge firewalling that *iptables* cannot provide. Indeed, *iptables* and *ebtables* can be seen as complementary and can be used in parallel [17].

The ebtables user-space tool is used to set up the ebtables rules in the kernel. All traffic entering or leaving on a bridge port will be seen by the rules. The ebtables syntax and usage are very similar to the iptables ones. Ebtables (cf. Figure 3) provides three tables: broute, nat and filter. Each of these tables have their own chains which are attached onto the hooks of the Linux bridge. In fact, the bridging code defines six hooks where ebtables can attach itself to process the frames going through the bridged interfaces. Given its flexibility and modularity, *ebtables* is extensively used in this work. In particular it is involved in the main aspects concerning scalability and mobility.

4) MadWiFi WDS: Both *AP(mesh)* and *Client(mesh)* interfaces use the 802.11 4-address fields format known as WDS [5]. Table II shows the relation between network mode, To/FromDS fields and MAC addresses. The ToDS and FromDS bits indicate whether or not a frame is destined for the distribution system. They are part of the Frame Control field of a 802.11 frame and determine the type of network deployed and indirectly the number and function of the address fields.

In the standard Infrastructure mode three address fields are enough since receiver (AP to User) or transmitter (User to AP) correspond respectively with destination and source. However, three addresses are not enough when an 802.11 device is acting as a wireless bridge. In this case the source and destination are different from transmitter and receiver of the frame on the wireless medium, hence four addresses are needed. WDS is

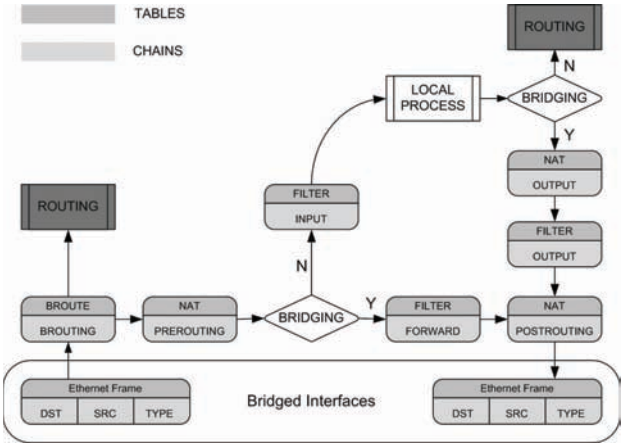

Fig. 3. Ebtables' tables and chains traversal process and interactions.

TABLE II
TO/FROM DS AND ADDRESS FIELD CONTENTS

Mode	ToDS	FromDS	No. of Address	Address 1	Address 2	Address 3	Address 4
Ad-hoc	0	0	3	RA=DA	TA=SA	BSSID	N/A
Infrastructure (AP to User)	0	1	3	RA=DA	TA=BSSID	SA	N/A
Infrastructure (User to AP)	1	0	3	RA=BSSID	TA=SA	DA	N/A
4 Addresses (WDS)	1	1	4	RA	TA	DA	SA

the mechanism that allows 802.11 frames with 4 addresses to be used. WDS allows packets to pass from one wireless interface to another, just as if the wireless devices were ports on a wired ethernet switch. As with all other data frames, WDS frames use the first address for the receiver of the frame and the second address for the transmitter. These two addresses are used for acknowledgments and control traffic, such as RTS, CTS, and ACK frames. The other two address fields are necessary to indicate the source and destination of the frame and to distinguish them from the addresses used on the wireless link. It's main application is to transparently bridge remote networks via a wireless link. Nevertheless, the applications envisaged by the use of this format are not defined or described in detail by the IEEE 802.11 standard [5].

Within our deployment we use WDS on the backbone's interfaces for two main reasons. Firstly, it is required in order to deploy a transparent bridged multi-hop wireless network. The Linux-based bridge is used to bridge together the interfaces on the same MN, but to connect the different MNs to each other in a transparent way over a wireless link, a wireless bridge is needed. The second is related to the standard limitation that makes it impossible for a wireless card in station mode to work properly when enslaved to an ethernet bridge, the reason being again the need for 4 addresses. The WDS 4-address format overcome this limitation enabling an Infrastructure mode station to operate when enslaved to an ethernet bridge. Indeed we have to point out that while most of WDS implementation provide interconnection among APs [8], [7], the Madwifi WDS implementation offers also the possibility to use WDS over the Infrastructure mode, thus allowing the use of the 4-address format among an AP and

Digital Object Identifier: 10.4108/ICST.WICON2010.8510
http://dx.doi.org/10.4108/ICST.WICON2010.8510

its associated client stations. This approach has been adopted for example by Mikrotik to transparently bridge two remote networks [18].

Figure 4 shows the interaction between the Linux bridge and WDS. A packet directed to end-user EU2 is received by the MP coming from the external network. After being processed by Layer-3 routing, the packet is sent to the internal virtual bridge interface and here it is forwarded out of the correct port based on the bridge's MAC table. In this example it would be forwarded to AP2. Similar to an ethernet bridge, the driver on AP2 keeps a table where each device's MAC address that has been discovered is associated to one of its WDS link partner.

In the above example the driver performs a look-up in the WDS table for the MAC address of EU2. This lookup will return the MAC address of the WDS peer interface Cl3, operating in station mode. Once received by Cl3 the packet is forwarded to the Linux bridge and the forwarding proceeds in the same way as before. When the packet reaches the MN which is the network point of attachment for EU2, it is transmitted from the *AP(user)* interface using the Infrastructure 3-address format. Notice that during the forwarding of the packet from the MP to EU2, the source and destination addresses do not change. On the other hand both the transmitter and receiver addresses are specific to each wireless hop and therefore are changed at each hop to the destination. This behavior is similar to the relation between IP and MAC addresses in a routed network.

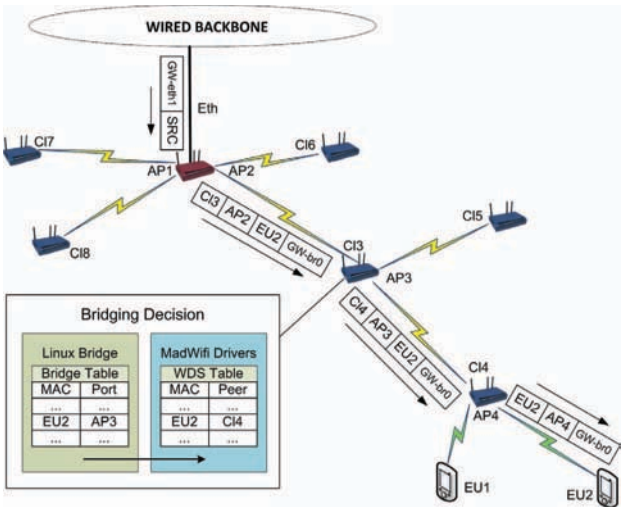

Fig. 4. Linux Bridge and WDS interaction.

G. Mesh Network Topology

There are three basic network architectures for WMNs: flat/-client WMNs, hierarchical/infrastructure WMNs, and hybrid WMNs [19]. Our solution belongs to the Infrastructure type. This architecture (cf. Figure 1) is characterized by static mesh routers connected to each other and to a few gateway nodes. The WMN routers form a multi-hop wireless access backbone for the clients. This means that the backbone nodes do not originate or terminate data traffic but they act as relay for the client nodes.

1) How the network builds itself using the beacons: Since the aim of the proposed solution is to offer wireless multi-hop access to an external network, primarily the Internet, the presence of a MP is necessary for building the backbone network. When a MB first boots, it begins scanning for other MNs using the *Client(mesh)* interface. Assuming it is not the first node in the mesh network, then it is likely to detect several *AP(mesh)* interfaces belonging either to other MBs or to the MP within its radio coverage range. The MB then chooses the best *AP(mesh)* to connect to according to what described in Section III-F1. The connection to an *AP(mesh)* is detected by *ifplugd* which will execute a configuration script. After connecting to an *AP(mesh)*, the MB begins to advertise both its own *AP(mesh)* interface to extend the wireless backbone and its *AP(user)* interface to allow end-users devices to connect. The beacon advertisement contains an updated version of the *mesh_beacon* information which was received by the *Client(mesh)* interface of the MB.

H. Mobility

In order to provide a good user experience when an end-user moves from one MN to another, a fast handover mechanism is required. Since in this work, frames are bridged at Layer-2 and all of the MNs belong to the same subnet, there is no special behavior required to get a basic mobility support.

A Link-layer handover is composed of three phases, scanning discovery, re-authentication and reassociation phase; the first of which accounts for most of the delay [20], [21]. This delay is mainly due to the behaviour of the end-user device and cannot be improved without requiring changes to the device.

However in a bridged network there is another delay which effects the handover latency. This re-bridging delay follows the association to the new AP and is the time needed to update both Linux bridge and WDS MAC tables along the path from the new MB to which the user is connected to the MP.

While this delay can be negligible in a typical 802.11 Extended Service Set (ESS) deployment where the APs are directly bridged to the switched DS, this is no longer the case in a distributed multi-hop wireless network. Furthermore, there is a difference in the delay between the upstream and downstream traffic with re-bridging time in the latter case likely to be much higher.

1) Upstream Traffic - ARP-Proxy: If the roaming end-user is sending upstream traffic or receiving downstream traffic using a protocol which requires acknowledgements, such as TCP, then the re-bridging along the path to the MP is performed automatically as soon as the client sends traffic through the new AP. However, during experimentation we noticed most devices flush their ARP cache after a handover. Therefore they need to send an ARP request to the default gateway and wait for an ARP reply before transmitting data packets.

However, since the wireless network may be composed of many nodes and may be heavily loaded, the ARP message exchange can take quite a long time leading to service disruption. For this reason each MN implements an ARP-Proxy mechanism which replies immediately to an ARP request from a client on behalf of the MP. The ARP-Proxy mechanism

Digital Object Identifier: 10.4108/ICST.WICON2010.8510
http://dx.doi.org/10.4108/ICST.WICON2010.8510

has been implemented by adding a rule in the PREROUTING chain of the NAT table in *ebtables*.

2) Downstream Traffic - Re-bridging Mechanism: If the roaming client was only receiving downstream traffic the re-bridging delay will be significantly longer. The reason is that the bridging tables will not be updated until the end-user sends data toward the MP through the new MB. Even if the ARP cache is flushed, the client will send an ARP request only when it has to send traffic upstream toward the MP. The downstream frames coming from the MP will still be forwarded toward the old MB and will therefore be lost. To insure a fast and deterministic update of the route to the MP the address learning mechanism of a bridge is exploited.

A bridge associates a host's MAC address with one of its ports through a process called *passive learning*. The bridge inspects the source MAC address of all incoming data frames and builds the MAC forwarding table accordingly; WDS works in a similar way. Therefore, since updating the path toward the MP just requires to update the bridge and WDS tables of the MNs along this route, it is enough to send a frame with the MAC address of the roaming end-user as the source MAC address. This must be done as soon as the end-user gets connected to the *AP(users)* interface of a MN. This requires the re-bridging mechanism to be hooked to the 802.11 Association sequence. Since this process is handled by the wireless drivers, modifications to the MadWifi sources were needed.

The re-bridging message is an Ethernet frame with the source MAC address of the end-user and destination MAC address of the MP. A unicast packet is sufficient and more suitable to update the route between end-user and MP. Using a unicast packet avoids flooding the network and increases the probability that the re-bridging frame will be received since, unlike a broadcast frame, a unicast frame requires acknowledgement (802.11 ACK) and therefore will be retransmitted in case of loss.

To further enhance the reliability of the re-bridging mechanism, priority is given to the re-bridging frame. This was accomplished by modifying the queuing system in the kernel sources. The standard FIFO policy would delay the transmission of the re-bridging frame in case of high traffic with consequent delay in the updating of the new route to the MP.

I. Scalability

Unlike a router, a bridge does not limit the scope of broadcast frames. Indeed, when a bridge receives a frame addressed to the ethernet broadcast address (FF:FF:FF:FF:FF:FF) or to a L2 multicast address, this is forwarded to every port except the one from which it was received. This means that if an end-user or a MN transmits a broadcast frame it will flood the entire network producing obvious traffic load issues. In the case of a relatively small mesh network consisting of only a few nodes, this is not really a problem. However, this uncontrolled flooding issue would become a problem if the mesh network was scaled up to cover larger areas. Nevertheless we must still support some protocols which make use of broadcast messages such as ARP and DHCP.

The main aim of our mesh network is to provide Internet access to the end-users and not to provide intra-mesh communication among them. Under this assumption it is sufficient to provide end-users only with the ability to communicate with the MP. Support for DHCP discovery and ARP request messages is all that is required for this purpose and therefore all other broadcast messages are dropped. Since these two message types are always directed to the MP, as soon as they are received by the MB to which the sending client is attached, a MAC DNAT changes the broadcast MAC address to the MAC address of the MP and they are unicast forwarded to the MP.

Basically, the only message type transmitted in broadcast over the mesh network is an ARP request from the MP looking for the MAC address of an end-user or a MB. The MAC NAT capabilities of *ebtables* were used to implement this behavior.

J. Topology Graph

In order to visualise the mesh network backbone, a topology graph tool showing the real-time interconnection between the MNs was developed. Given the distributed and dynamic nature of a wireless multi-hop network, the usage of a tool to assess the network topology of the testbed is essential.

An external topology server collects information which it periodically receives from each MN in the network. This information is processed so to get a complete view of the current network topology which can then be displayed as a runtime updating image.

IV. Results

In this section we present results which demonstrate the automated topology construction mechanism and the enhancements made to the normal 802.11 Layer-2 mobility mechanism.

A. Beacon Enhancement based Automated Configuration

A key aspect of WMNs is represented by their dynamic, self-configuration and self-healing features which are usually provided by a routing protocol. The drawback is the overhead introduced by the routing protocol, the computational requirement on an embedded system and the variability impact of periodic link-quality metric updates which have to cope with unreliable wireless links.

Our solution offers a basic version of these features but without the need for a routing protocol. In fact, since the use case for the testbed developed in this work is for static MNs, the routes are not likely to change unlike what can be expected in a MANET. It allows for the automatic set-up of a wireless mesh backbone without requiring any time-consuming manual configuration. The beacon enhancement incorporates a minimum-hop link metric without requiring signalling from a routing protocol. To fully support the beacon extension, the *wlanconfig* tool has been altered so that it can display the additional information included in the beacon frames (cf. Figure 5).

The self-healing feature is provided by the *ifplugd* daemon that triggers an immediate response from the mesh-daemon upon connection/deconnection updates. Another dynamic element is the ability of each node to switch smoothly from a MP

Digital Object Identifier: 10.4108/ICST.WICON2010.8510
http://dx.doi.org/10.4108/ICST.WICON2010.8510

to a MB configuration and viceversa. Finally, a basic channel assignment mechanism is present on both MP and MBs to reduce the presence of overlapping channels. Nevertheless, this point needs further investigation and will be addressed in the next development of the project.

```
mesh99:~# wlanconfig ath2 list ap
SSID            BSSID              CHAN RATE  S:N   INT   GW HOPS  COG   CC CAPS
PELBackbone     00:0c:42:0c:ed:d0   56   54M 41:0   100  117    0    1    1 EPs  WME ATH
PELBackbone     00:0c:42:0c:ed:b9   60   54M 44:0   100  117    0    1    1 EPs  WME ATH
PELBackbone     00:0c:42:0c:ed:c4   64   54M 39:0   100  117    1    1    1 EPs  WME ATH
PELBackbone     00:0c:42:0c:ed:be   36   54M 29:0   100  117    1    1    1 EPs  WME ATH
PELBackbone     00:0c:42:23:0c:04   40   54M 59:0   100  117    1    2    0 EPs  WME ATH
PELBackbone     00:0c:42:05:72:94   48   54M 58:0   100  117    2    2    0 EPs  WME ATH
PELBackbone     00:0c:42:1b:83:d0   44   54M 70:0   100  117    3    2    0 EPs  WME ATH
mesh99:~#
```

Fig. 5. Modified beacon frames displayed by *wlanconfig*.

B. Mobility Performance

Several experiments were performed to assess the utility and performance of the ARP-Proxy and re-bridging mechanisms. A client connected to the wireless backbone performs a handover between two MNs every 30 secs. Each test was run for 1000 seconds for each scenario and with a varying number of hops to the GW.

1) ARP-Proxy: For the ARP-Proxy tests the client transmits an upstream UDP flow toward an external server at 2Mbps; this traffic was generated using the *iperf* tool. A 5Mbps flow of background UDP traffic was also transmitted on the path from the MP to the client to increase the traffic load in the network. The card being used flushes the ARP-cache at every handoff. In Figure 6 we evaluate the time difference between the ARP request delivery and the moment we start sending actual data for a different number of hops, with and without the ARP-Proxy mechanism.

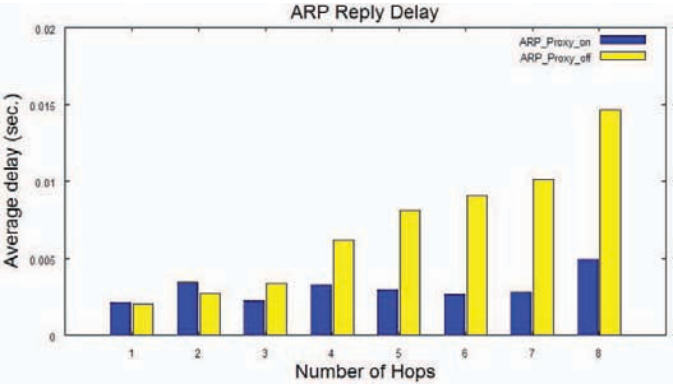

Fig. 6. ARP reply delay with/without ARP-Proxy and different number of hops.

With the introduction of the ARP-Proxy mechanism, the ARP-Reply delay is quite constant regardless of the number of hops. In this case, since each MN the client gets attached to replies to an ARP request on behalf of the GW, the delay is not correlated with the numbers of hops. When not using the ARP-Proxy mechanism, the ARP-Reply delay increases with the number of hops to the MP. This behavior is easily explained with the greater number of MBs that ARP-Reply and ARP-Request have to cross for each added hop. Furthermore this trend is likely to be accentuated with the increase of the traffic load in the network. Nevertheless the range of variation of this delay is quite small, in the order of tens of millisecond with a maximum of 15 ms for eight hops. This value is almost negligible when compared with the handover latency that is in the scale of hundreds of millisecond [20].

This small range of variation is reflected in the results of Figure 7 where we observe almost the same Packet Loss Ratio (PLR) with or without ARP-Proxy. Furthermore we can notice that there is no direct correlation with the results of Figure 6, according to which a higher PLR would be expected with the increase in the number of hops when not using ARP-Proxy mechanism. This is the confirmation that the ARP-Reply latency is almost negligible and the handover latency accounts for the most of the loss.

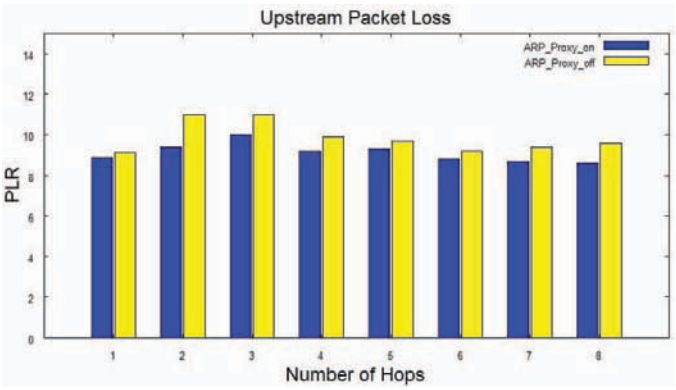

Fig. 7. Loss rate with/without ARP-Proxy and different number of hops.

2) Re-bridging: The re-bridging mechanism has been tested using the same parameters as for the ARP-Proxy but with the UDP flow in the opposite direction. Now the roaming client is receiving the stream while performing a handover every 30 secs. In Figure 8 the time difference between the moment the client receives the Association Response following a handover and the moment it starts receiving the data packets is examined. We computed this delay only when the re-bridging mechanism is used.

Indeed, when the re-bridging mechanism is disabled, the data flow coming from the MP is still forwarded to the client's previous point of attachment at the old MB. In this case the latency is not deterministic but rather it depends on the moment the MP sends a broadcast ARP-Request to the end-user. The ARP-Reply from the client will update the bridging table of the MNs en route to the MP.

On the contrary, when using the re-bridging mechanism, as soon as the end-user gets connected to the new MN, this sends an ethernet frame to update the path to the GW. In this case the delay is very short and increases slightly with the number of hops to the MP.

In Figure 9 a comparison of the PLR with and without the re-bridging mechanism is presented. As expected there is a big difference in the PLR in the two cases. The results obtained when the re-bridging mechanism is used are comparable with those of Figure 7 where the handoff process latency accounts for the most of the loss. On the contrary, when the re-

Digital Object Identifier: 10.4108/ICST.WICON2010.8510
http://dx.doi.org/10.4108/ICST.WICON2010.8510

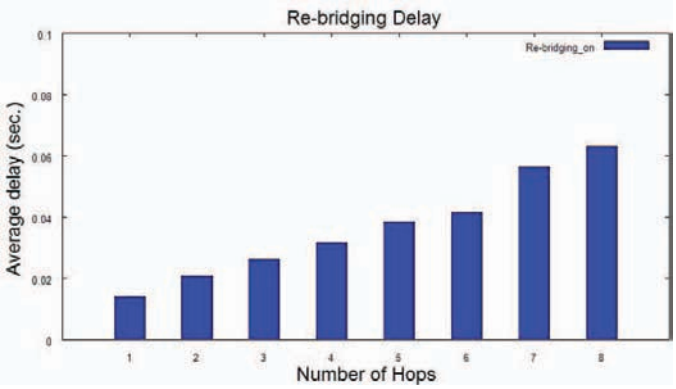

Fig. 8. Re-bridging delay with/without re-bridging mechanism and different number of hops.

bridging mechanism is not used, the PLR increases up to the order of 30%. Since the bridges are not aware of the client's switch until the end-user sends a packet from its new MN of attachment, the client-transparency is not guaranteed and there is a re-bridging delay following the association that is no more negligible when compared to the handoff latency.

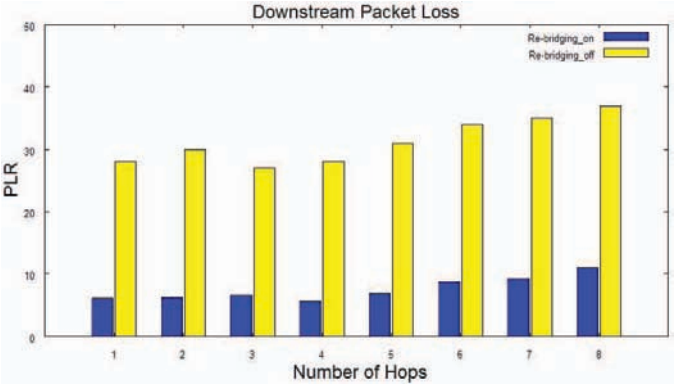

Fig. 9. Loss rate with/without re-bridging mechanism and different number of hops.

Finally it is interesting to notice that in both Figure 7 and 9 the PLR is almost constant regardless of the number of hops to the MP. This is mainly due to the adoption of multiple 802.11 radios working on non-overlapping channels that offer higher levels of capacity and scalability than single radio solutions.

V. Conclusion and Future Work

Wireless Mesh Networks are gaining increasing attention as a low cost approach for providing wireless Internet access in a similar fashion to IEEE 802.11 based APs. Although a significant amount of work has been done in the WMN domain, it is essential that an understanding of the issues relating to the development and implementation of real WMNs is gained. In this paper we presented an enhanced bridged-based implementation solution intended for Infrastructure based multi-hop wireless networks. The goal of our design is to provide dynamic, self-configuration and self-healing features without the need for a routing protocol. In this way the flexibility of Layer-2 bridging mechanism and the self-healing capabilities of a MANET routing protocol are combined. The main issues related to applying a pure bridging based solution to wireless multi-hop networks were investigated and several features to overcome these problems were introduced. Furthermore a detailed implementation description of the real testbed was presented. Finally experimental results involving measurements of the re-bridging latencies on Layer-2 handoff times with the aim of assessing the utility of our mobility improvement solutions were presented.

Future work will be to perform further experimental tests to evaluate performance and overhead of the proposed solution and compare them with results obtained using a pure AP-to-AP WDS bridging solution and with MANET routing protocols. In addition we are also working on the development of an optimal channel assignment algorithm for wireless mesh nodes equipped with multiple radios.

Acknowledgment

The research leading to these results has received funding from the European Community's Seventh Framework Programme (FP7/2007-2013) under grant agreement no. 214994.

References

[1] A. Herms and G. Lukas. Awds (ad-hoc wireless distribution service). [Online]. Available: http://awds.berlios.de/about.html
[2] C. E. Perkins and E. M. Royer, "Ad-hoc on-demand distance vector routing," in *Proc. Second IEEE Workshop on Mobile Computing Systems and Applications WMCSA '99*, Feb. 25–26, 1999, pp. 90–100.
[3] T. Clausen and P. Jacquet, "Optimized link state routing protocol (olsr)," Published Online, IETF, RFC 3626, October 2003. [Online]. Available: http://rfc.net/rfc3626.txt
[4] C. E. Perkins and E. M. Royer, "Ad hoc On-Demand Distance Vector (AODV) Routing for IP version 6," IETF, Internet-Draft draft-perkins-aodv6-01, Nov. 2000.
[5] D. Engwer. (2005, July) Wds clarifications. [Online]. Available: http://www.ieee802.org/1/files/public/802_architecture_group/802-11/4-address-format.doc
[6] Freifunk project. [Online]. Available: http://start.freifunk.net/
[7] 3COM. (2004) Configuring a wireless distribution system (wds). Technical Brief. [Online]. Available: http://www.3com.com/other/pdfs/products/en_US/104108.pdf
[8] ORiNOCO. (2002, February) Wds (wireless distribution system). Technical Bulletin (046/A). [Online]. Available: http://www.proxim.com/support/techbulletins/TB-046.pdf
[9] ifplugd. [Online]. Available: http://0pointer.de/lennart/projects/ifplugd/
[10] Mikrotik routers and wireless. [Online]. Available: http://www.mikrotik.com/
[11] Debian. [Online]. Available: http://www.debian.org/
[12] The linux kernel archives. [Online]. Available: http://www.kernel.org/
[13] The madwifi project. [Online]. Available: http://madwifi-project.org/
[14] ebtables. [Online]. Available: http://ebtables.sourceforge.net/
[15] M. S. Gast, *802.11 Wireless Networks. The Definitive Guide*. O'REILLY, 2005.
[16] U. Bohme. Linux bridge-stp-howto. [Online]. Available: http://www.faqs.org/docs/Linux-HOWTO/BRIDGE-STP-HOWTO.html
[17] C. Benvenuti, *Understanding Linux Network Internals*. O'REILLY, 2005.
[18] MikroTik. Transparently bridge two networks. [Online]. Available: http://wiki.mikrotik.com/wiki/Transparently_Bridge_two_Networks
[19] I. F. Akyildiz, X. Wang, and W. Wang, "Wireless mesh networks: a survey," *Comput. Netw. ISDN Syst.*, vol. 47, no. 4, pp. 445–487, 2005.
[20] A. Mishra, M. Shin, and W. Arbaugh, "An empirical analysis of the ieee 802.11 mac layer handoff process," *SIGCOMM Comput. Commun. Rev.*, vol. 33, no. 2, pp. 93–102, 2003.
[21] C.-S. Li, Y.-C. Tseng, and H.-C. Chao, "A neighbor caching mechanism for handoff in ieee 802.11 wireless networks," in *Proc. International Conference on Multimedia and Ubiquitous Engineering MUE '07*, Apr. 26–28, 2007, pp. 48–53.

Digital Object Identifier: 10.4108/ICST.WICON2010.8510
http://dx.doi.org/10.4108/ICST.WICON2010.8510