Route Stabilization in Infrastructured Wireless Mesh Networks: an OLSRD Based Solution

Gianni Costanzi, Renato Lo Cigno
DISI – Università di Trento
Trento, Italy
locigno@disi.unitn.it, gianni.costanzi@gmail.com

Andrea Ghittino, Stefano Annese
CSP Innovazione nelle ICT
Torino, Italy
andrea.ghittino@csp.it, stefano.annese@csp.it

***Abstract*—This paper presents a route management protocol for multi-homed infrastructured WMNs aiming to preserve active sessions of mobile nodes. Every Mesh Access Point (MAP) has at least one MAP interface configured as a traditional AP and another one dedicated to the WMN backhaul. The clients are mesh-unaware to avoid installing special software on them.**

We propose a mechanism based on dynamic IP-within-IP tunnels to pin routes and guarantee (in time) the communication between clients and Internet Gateways. MAPs select the best gateway for each new connection and maintain tunnels for the connection duration to allow the seamless communication during the whole flow life. When a client changes MAP, the old and the new one exchange tunnel information so that the new MAP re-establishes tunnels with the correct gateway ensuring smooth packet delivery.

The protocol is an extension to OLSRD, and has been implemented in Linux-based MAPs. Evaluation is carried out both emulating complex scenarios with User Mode Linux (UML) and in a real WMN testbed.

I. Introduction

Wireless Mesh Networks (WMNs) have been subject of research for many years. In the last period, in part following fundamental results, and in part as a consequence of WiFi success, WMNs moved from being only and academic research subject to an important industry topic with many implementations in the field. Famous examples are MIT-based roofnet [1] and Rice efforts in Houston [2]. Also in Europe communities and Universities are moving forward to implement large scale meshes [3], [4]. Citing all ongoing projects is unfortunately impossible due to space constraints, also because new projects and experiments are initiated on a nearly daily basis.

In its more comprehensive definition a WMN is an access network where coordinated Mobile Access Points (MAPs) offer service to a multitude of clients (User Node – UN). MAPs and Mobile Routers (MRs) cooperate to build a backhaul network connecting MAPs to Mobile GateWays (MGWs) that are the interconnection points toward the Internet. Figure 1 depicts this general definition of WMN.

MAPs, MRs, and MGWs are indeed all routers that take different roles depending on the network needs and their capabilities.

A single device can have multiple roles at the same time (e.g., MAP and MGW), or to switch from one role to another, e.g., because an Internet link is set up or tear down. We will collectively indicate these nodes MRs for simplicity, also because MAPs and MGWs are always also routers. User Nodes are instead different, in that they only access the network and do not cooperate to build it. UNs are mobile, and MRs can be mobile too, albeit most of the time their behavior will be more "volatile" than mobile: they may switch on and off, change the resources dedicated to the network etc., but in most realization they will rarely roam the network.

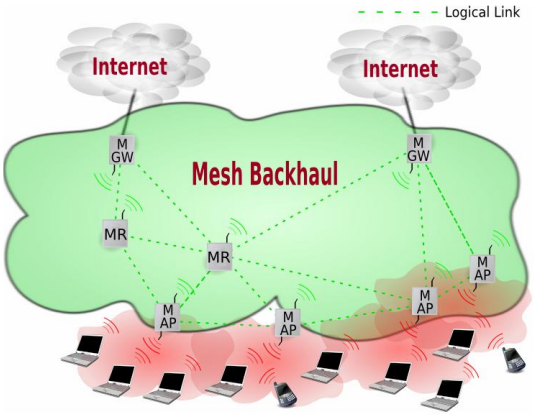

Figure 1. A generic WMN with the access part, the backhaul and access gateways

Physical resources (i.e., the 802.11 channel) dedicated to the backhaul are different from those in the access to improve performances. The presence of different physical resources at the same time implies that each MR must have at least two radio ports: one for the access part and one dedicated to the backhaul network. MGWs are connected to the WMN and also to the internet with dedicated connections (ADSL, WiMax, Ethernet, ...). The network can work also with a single 802.11 interface on MAPs and MRs, but then the difference between them and UNs becomes blurred and the system resembles more a standard ad-hoc network rather than a WMN. In many works concerning WMNs, UNs are supposed aware of the environment, i.e., they know the WMN and cooperate to make it work. We deem this constraint unrealistic for operational WMNs, so all our work aims at supporting users as if they were connected to a standard WLAN and not to a WMN.

The remaining part of this paper is organized as follows. Sect. IA discusses the possible routing strategies in WMNs, refining the definition of the problem we tackle, and Sect. I.B

978-1-4244-1959-3/08/$25.00 ©2008 IEEE

discusses recent works focused on this same problem. Sect. II is dedicated to explain the rational of starting from OLSR and extend it to obtain our goals; Sect. III and IV present our proposal and implementation in detail, and Sect. V deals with validation and performance measures obtained with its actual implementation on dual-radio, embedded Linux MRs. Finally Sect. VI concludes the paper with a short discussion on the evolution of the work.

A. *Routing Problems and Strategies*

Multi-homed WMNs can be used as the infrastructure to connect mobile users to the Internet [5], [6] and [7]. Multi homing is necessary for high performance, reliability and resilience, as well as (in some cases) to reduce the costs of maintenance. The different MGWs can indeed be connected to different providers, thus complicating the overall scenario. In most cases, since the WMN is not exactly a "public network" but is built dynamically and on-demand by the users themselves, addresses in the WMN are not routable IP addresses, but are assigned dynamically via DHCP from a set of private IP addresses, and each MGW operate as Network Address Translator (NAT) [3] and firewall.

Client stations may roam through the MWN, changing MAP as if MAPs were forming a normal 802.11 ESS. A WMN, however, is not a normal ESS: the distribution system is the mesh backhaul and routing with multi-homing may result in connections disruption if the MGW is changed. Additionally the routing protocol must react quickly in order to guarantee packet delivery to the moving node.

Even in absence of moving clients and with quasi-static MAPs, the best gateway for each mesh node (the gateway with minimal routing metric from the MAP) may dynamically change due to the fluctuations of wireless links among the MRs.

Additionally, as we already stated, UNs should be unaware of the peculiarities of the WMN, and be equipped and behave just as if they were in a normal ESS. The MAPs should mask the network architecture to the clients: the MGW is chosen by the MAP which the UN is connected to. When a UN moves from a MAP to another one, it is the new MAP that holds the burden to interconnect it with the Internet, and do it without service disruption.

Routing in a WMN should be aware of all these constraints and operate consequently, preserving the key features of a WMN, in particular its "feeling" of being just a different implementation of an 802.11 ESS. For this reasons the "standard" routing solutions for ad-hoc networks routing (e.g. OLSR or ADOV [8], [9]) are inappropriate. Ideally, the solution should operate on the MAC address space, since the whole WMN is just a multi-homed IP subnet. Unfortunately, standard layer-2 solutions are not available, and they cannot be implemented with standard, off-the-shelf devices and components. The alternative is the use of IP-level routing, adapted to the needs and requirements of a WSN. As motivated in Section II, we decided to modify an open source implementation of OLSRD [10] to introduce the desired characteristics. Before describing the contribution of this paper in Sect III and IV, we overview the works that inspired us.

B. *Related Works*

Several researchers have studied the problem of gateway selection and session management in a WMN. Tajima et al. in [11] describe a network that is based on nodes with two radio interfaces, one operating as 802.11b/g and another as 802.11a. The first one manages communications among MAPs and UNs; instead, communications between adjacent MAP are based on 802.11a. This paper proposes a heuristic to select the best gateway to minimize the total traffic in the WMN in order to reduce congestions.

[12] studies the MGW selection problem in a WMN. The authors criticize traditional metrics that are only based on hop-count because it can overload some MGWs; they suggest an algorithm that is based on node mobility evaluation, where higher mobility means higher link failure probability.

Both papers [11] and [12] modify the base routing protocol behavior and the metrics to implement the described mechanism.

A comparison between default forwarding and tunneled forwarding techniques is made by Nordstron et al. in [13]; this paper evaluates benefits of half tunnels that tunnel traffic only in a direction, from MRs to MGWs. This approach is transparent and independent of existing routing protocols and involves only the source node and the used MGW. Intermediate nodes are unaware of tunnels and gateways that are selected by the source node. This approach guarantees a high stability because new MGWs introduction or metric changes don't interfere with existing flow routing. Indeed, it diverts to a new MGW only if connectivity with the old one is completely lost. The half-tunnelling scheme discussed in this paper is part of our global solution.

A different approach to manage UN mobility is described by MobiMESH [5]. The reference scenario separates the access network, based on traditional APs, and core network, based on mesh paradigm. The algorithm introduces an OLSR HNA (Host and Network Association) based mechanism to manage UN routing between access and core network. The access network is a single IP subnet: no IP layer mobility is adopted because clients do not change their IP when associating to a different AP. The mechanism introduced to manage mobility maps layer 2 changes to layer 3, through a MAC-IP association. The implementation is based on ARP proxying, DHCP Relaying and a mobility management database that can be distributed or centralized.

An alternative approach to manage UN mobility ensuring Internet connection stability is based on SMesh system [14] and [15]. The SMesh architecture is like MobiMESH: mobile UNs are mesh-unaware and connected to the Internet through MAPs. The connection between a MAP and a UN is not in 802.11 "infrastructure mode" but in "ad hoc mode", and each UN is associated by MAPs to a unique multicast group to receive data. The handoff between MAPs is based on three steps: first of all, if a MAP detects a UN, it subscribes its multicast group then, when a packet for the specified group is received, the MAP forwards it to the UN. Finally, the handoff control is based on gratuitous ARP to force the use of the new MAP as the default gateway. Moreover, UNs in SMesh have a private address space and NAT is done at Internet gateways when a node communicates with an external host. To handle UN mo-

bility between different MGWs, SMesh introduces an inter-domain handoff protocol: when a gateway receives a data flow, it tries to detect the existing owner (the MGW from which the Internet connection was initiated) and forwards packet to it.

Most of these proposals give partial solutions to the global problem we are tackling and in some cases we were supported in our decisions by their positive results. However, none of the above solutions globally covers all issues. SMesh is complex, requires the use of multicast, is architecturally an ad-hoc and not a mesh solution and uses a tunneling mechanism that is rather cumbersome and inefficient. MobiMESH does not address the problem of multi-homing (one of the most important!) and manages UN mobility directly with HNA, which means that it does not distinguish between UNs and remote networks reachable through the MGW. Both proposals in [11] and [12] are focused on novel routing metrics and algorithms, so that they are difficult to develop as backward compatible solutions.

II. The Choice of OLSR

Before describing the design of our proposal it is important to understand why we base it on OLSR.

The first requirement was a stable implementation to focus our efforts only on innovative solutions: rewriting a complete routing protocol from scratch is, for the time being, beyond our resources. Both AODV [8] and OLSR [9] have good open source reference implementations, used in laboratories and real test beds. An interesting comparison between these protocols is done by Chen et al. in [16]. They don't evaluate the RFC OLSR, but they analyze the behavior of the Fisheye OLSR version. This mechanism was introduced in olsr.org implementation to overcome the scalability problems. Fisheye uses Time To Leave (TTL) to modify Traffic Control (TC) message flooding. TC with low TTL are sent more frequently than TC messages with higher TTL. Therefore, neighbors have more up to date information than the rest of the mesh.

The performance analysis demonstrates through simulations that AODV is efficient with few data sessions, but OLSR exhibits a much better scalability of traffic loads. OLSR has a constant messaging overhead while AODV signaling messages increase considerable when data traffic grows. In this scenario, AODV route discovery and reply messages are often lost due to high network load. Therefore, network stability and performance can be compromised.

Having selected an OLSR base for scalability, we decided to base our implementation on the olsrd daemon developed by olsr.org [olsr.org] community. It is well documented and it can be easily extended through the plug-in support. Moreover, the developer community is very active and the software is successfully used in several test beds, as Freifunk Berlin [3] and Ninux Rome [4] communities. Finally, it already presents some features to support multi-homing, as the possibility to advertise remote networks to make the Internet reachable.

III. The Proposed Solution

The goals of the solution we propose are the following:

- Full transparency for UNs, which should not be forced to install any new software, not to participate in OLSR routing;
- Continuity of service also to long term, secure (TLS based) connections when the WMN backhaul routing changes and when MGWs are added/removed (unless the connection was through a removed MGW);
- Seamless support of mobility and handovers across the entire WMN.

The first goal can be obtained through the exchange of dedicate OLSR messages: the MAPs will participate in OLSR routing on behalf of the UNs in their BSS. The second goal is obtained by properly building and maintaining half tunnels. The third one requires the exchange of context information between MAPs.

A. *Transparency to the UNs*

As shown in [5] to hide OLSR to UNs, it is sufficient that MAPs advertise their addresses through HNA messages, which are standard OLSR messages to announce remote networks. This solution, which can be adopted in our implementation if required, has the advantage that non-MAP MRs can be entirely standard, so that only MAPs and MGWs are to be modified to obtain the enhanced WMN. On the other hand, however, this solution does not allow the distinction between local nodes and remote networks, and would also reduce the performances during handovers. To solve this problem we define a new type of messages (a modification of HNA) that allows more information to be exchanged between MAPs, thus supporting seamless handovers and allowing al MRs to be aware of the locality (or not) of advertised addresses.

B. *Avoiding Connections' Break-down*

The first objective (in order of importance) of our project is to give a solution to the connections' breakdowns that are caused by the changes of MGW toward the external networks. To achieve this result we must route the IP packets relative to a specific connections through a fixed MGW, which is the one selected as "best" at the beginning of the connection. When the connection is established, the packets are directly sent through an IP-within-IP tunnel toward the MGW that is currently the best for the OLSR routing. This solution, discussed in [17], is quite simple but has a major drawback: when a MAP selects a MGW, it continues to use the same MGW for every new connection. Instead, our approach will use for each connection the best available gateway; to implement this mechanism is necessary to monitor continuously OLSR messages (in particular HNA) and routing table to detect changes.

C. *Per-connection MAP to MGW IP-IP Tunnels*

Each node of the mesh that runs our software must keep track of every connection toward external networks and for each of them it must remember the MGW to be used. To do this every MAP intercepts the layer-3 traffic of the UNs in its BSS and directs it to an Encapsulator software module which mainly accomplishes the tasks of identifying new connections or recognizing existing ones and encapsulate the packets into another IP packet destined to the right MGW.

We do not want the nodes of the mesh to encapsulate the traffic received from other mesh nodes destined to an external network: in fact, we assume that if a node sends IP packets with an out-of-the-mesh destination, then it does not want to do

encapsulation, otherwise it would have encapsulated them itself. Such packets will be routed toward the actual best GW, as it happens in the regular olsrd daemon.

We decided to use half-tunnels, from a mesh node toward a GW. Tunneling in the reverse direction is not necessary, since the destination is a UN that is always best reached with the up-to-date OLSR decisions.

The presence of mesh nodes that do not run our software do not prevent tunneling-aware mesh nodes to encapsulate the traffic, since they correctly forward IP-encapsulated IP packets, which are seen as normal IP packets with an in-mesh destination. Even MGWs may not run our software but they must at least support the IP-IP tunneling protocol, in order to be able to de-encapsulate the received packets.

D. *Mobility Support*

The mobility of UNs implies the support of seamless handoff support, which can be particularly delicate in presence of tunnels, since the tunneling information must be transferred to the new MAP as context information. The first problem is how to discover the new MAP. The simplest solution is that, upon a MAC-level re-association the new MAP immediately uses standard OLSR messages to flood the network with this information. In such way the old MAP can send to the new one all the information about the existing connections. This latter information, however, requires the definition of new messages and a new handover protocol within OLSR, since there is not even the notion of handover within OLSR.

To minimize handoff delay, the old MAP can monitor UN signal quality and, when it detects degradation, it can assume that the UN is moving. Then, the MAP adds tunnel information to the OLSR message that is used to advertise associated UNs. Through this mechanism, when the UN joins a new MAP tunnel information are available and active connections to the Internet continue without any break.

IV. Implementation Details

We decided for an olsrd plug-in implementation to ensure a modular approach and, at the same time, to permit a tightly cooperation between our software and the routing daemon. This cooperation is required to guarantee the routing information exchange, avoiding the communication overhead required by using two separate daemons.

Our solution is based on three main blocks: the "Encap plug-in", the "Encapsulator thread" and the "GW tables", as indicated in Figure 2. We are also defining a new module, the "Handoff manager" to handle UNs that are moving among MAPs, but it is not yet implemented.

A. *GW Tables*

This block is the main information repository and is used by our two software modules; it is based on two tables: the current best GWs' table and the connections' table.

The first one is managed by the Encap plug-in and it is used to store the current best GW for each of the networks that are advertised in the received HNA messages.

The connection table is used to store active tunnel data. The connections are identified by a tuple that includes: source IP address, destination IP address, layer four protocol, source port and destination port. This table stores also the IP address of the MGW that is used by each connection.

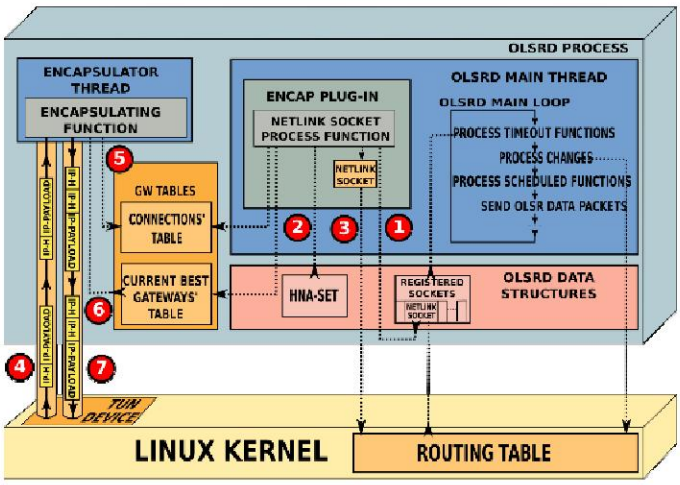

Figure 2. Interaction among software modules and between User Space and Kernel Space

B. *The Encap Plug-in*

We want to keep the information within the tables shared with the encapsulator thread up to date, so it must be notified in some way when changes happens in the HNA set and more in general in the routing table. For example, the deletion of a route toward a MGW present in some of the entries of the best MGWs' table may require a deletion of those entries and the re-computation of the best MGWs for those networks that had the deleted MGW as best MGW.

To avoid the entire table re-computing we use a particular type of sockets provided by the Linux kernel, the netlink sockets. These sockets allow User Space tools to interact with the kernel, manipulating the routing table and the ARP cache or interacting with the Linux firewall. Furthermore, they allow the kernel to push into the User Space some multicast messages whenever particular events happen. One of the events that can be monitored from the User Space is the modification of the routing table, which is what we want to trap with the Encap plug-in.

Through the Netlink Socket, the Encap plug-in can modify the routing table and decide to update the MGW Tables.

For example, when a route toward a MGW has been deleted, all the tunnels that have that MGW as destination must be marked for later deletion (they are not deleted to avoid breaking connections immediately, in case the MGW becomes reachable in a short amount of time). The current best MGWs' table must be recomputed if a route toward the MGW is not available also in the olsrd internal routing table. The check on the olsrd routing table is necessary, because the following may happen: olsrd may find a better route toward a MGW, which will involve a change of next hop in the olsrd internal routing table; this will cause a route deletion that must not trigger a deletion of entries within the connections' table, since it will be immediately followed by a route addition.

In Figure 2, the interaction with olsrd data structured is managed by arrow 1 and 2; the second one is dedicated to handle HNA information, required to obtain MGW IP address.

C. *The Encapsulator Thread*

This thread works in parallel to the olsrd thread and continuously waits for IP packets that must be tunneled toward a GW. It receives from the kernel only those IP packets that are generated by the UN associated to the MAP or that are generated by the mesh node itself and that have an "out-of-the-mesh" destination.

The thread waits for the next IP packet to encapsulate and retrieves it from the kernel through arrow 4 in Figure 2 and then it looks at the IP and Transport Layer headers to build a connection tuple for which it looks for a match within the connections' table (arrow 5). If a match is found, it means that the IP packet belongs to an already established and active connection, so the packet must be tunneled toward the MGW used for that connection. If there is no match, the IP packet must be considered the first packet of a new connection and the current best GWs' table must be checked to find the best GW for the destination IP address of the packet (arrow 6). When the MGW to use has been determined, the encapsulator thread adds a new IP header in front of the IP packet and sends it back into the kernel (arrow 7), which will route it toward the chosen GW.

The packet exchange between the Kernel Space and the User Space is handled trough Iproute 2 tools and the Universal TUN/TAP Driver. This driver consists of a character device which allows creating, from the user-space, virtual network interfaces (TUN or TAP interfaces). Then, reading from or writing to the TUN/TAP character device, user-space tools are able to receive packets from the kernel or inject packets into it, respectively. There are two types of virtual interfaces, TUN and TAP. TUN interfaces exchange with the user-space IP packets, while TAP interfaces exchange complete Ethernet frames. A TUN interface is exactly what we need to push the IP packets out of the kernel and inject them back again from the Encapsulator thread.

We also faced with another problem, how the kernel can establish which packets have to be sent trough TUN/TAP interface to User Space.

The rule is based on two conditions: (i) they have an out-of-the-mesh destination and (ii) are generated by the stations associated to the mesh AP or by the mesh node itself. Moreover, the kernel can not interact with the olsrd information repositories, so it is not able to determine if an IP address is related to an out-of-the-mesh host, so we must configure the routing tables in such a way that the kernel is forced to take the decision to send an IP packet over the TUN interface only if the necessary conditions are satisfied. We can accomplish this task through the addition of two routing tables which are examined before the main routing table (the kernel examines the tables in the following order): first of all, the in-mesh table that contains an entry for each of the in-mesh destinations and is maintained by the Encap plug-in through a netlink socket. Then, the tun-encap table that contains a single entry which tells the kernel to send the IP packet over the TUN interface. The kernel is configured to examine this table if and only if the packet comes from the BSS' interface or if it has been generated by the local host

The main table contains all the entries that are in the in-mesh table plus all the entries related to the external networks, and it is modified by olsrd. If the kernel is examining this table, it means that it has not found a match in the two previous tables.

We are aware that this "user-space" solution may incur in efficiency problems and high CPU usage, and we are exploring in-kernel solutions; however the need of interaction with olsrd (which is a single user-space process) prevents most of the obvious/logical solutions.

D. *Handoff Manager*

The mobility management is still under implementation, so that we cannot give the details here. Fortunately, many software structures in olsrd are already well developed to support this evolution, and we already defined the context switching environment.

Besides implementing the context switching and handover support, several optimization strategies can be envisaged starting from SNR aware mechanism to predict movement of the UN to joint schemes for supporting handovers and topology changes when handovers are not due to mobility of UN but to changes in the overall topology of the WMN.

V. EXPERIMENTAL VALIDATION

The experimental validation has been done with the focus on two main aspects: compliancy to the design and initial performance testing. The first approach is useful to validate the software behavior and can be carried out on User Mode Linux (UML) emulation environment. On the other side to evaluate performances a real network is required to do bandwidth and delay measurements.

A. *Functional Test*

UML allows the emulation of complex networks through multiple virtual machines (VMs). Each VM runs as an application within a normal Linux machine (known as the host) and is a simple process in user-space. This approach permits to develop and test network software emulating complex topologies, with several nodes. Through UML is possible to analyze software behaviors when a node fails or network topology changes. Instead, it is not possible to use UML to evaluate performance (e.g. maximum throughput or delay) because VM and network emulation share the host PC resources, and are subject to the host operating system scheduling. Besides, UML does not emulate the wireless medium but only ideal Ethernet links, without collisions and packet loss. This is not a problem as we are interested in the validation of the functional behavior of our system and not on performances.

The base validation topology is presented in Figure 3; the network is composed by five MRs, two MGWs and one MAP. Olsrd selects the best gateway evaluating hop-count and link quality through Expected Transmission count (ETX); in this VM based scenario, MAP selects MGW1 as GW to connect to the Internet. We emulate also an Internet node that can be reached through MGW1 and MGW2.

For every test, olsrd daemon is configured to send a Hello and a TC packet every second and HNA and MID (Multiple

Interface Declaration) every 5s. The link choice is based on ETX mechanism and link availability is evaluated using a 20 packet window.

When the UN starts a flow to an Internet node the MAP creates the tunnel between itself and MGW1. The target of the first test is to verify that the introduction of a new best gateway for the MAP doesn't break down the active sessions. We introduce a new MGW (MGW3) that interconnect directly MR1 to the Internet, as indicated in Figure 4. Olsrd detects the new node and the MAP selects this one as new best GW.

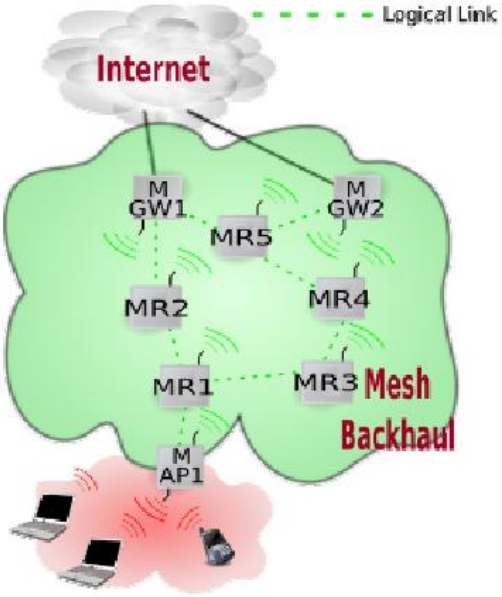

Figure 3: Base validation topology used for UML testing

Without countermeasures the connection breaks down, with a Reset for the TCP connection as soon as MAP or MR1 change their routing table. Instead, with our solution, the active session is encapsulated into TUN1 and still uses MGW1.

Moreover, when the UN activates a new flow to an Internet host, the MAP detects the new connection beginning and creates a new tunnel to the current best gateway, as indicated in Figure 4. Now, the UN1 uses two MGWs to connect to the Internet: MGW1 through TUN1 and MGW2 through TUN2.

Figure 5 shows the download progress for the two connections. The vertical line after 40s indicates that the MAP1 selects MGW3 as best gateway; the active connection, between UN1 and an Internet web server goes on without trouble at 400kb/s through MGW1. Any new connection after that moment will go through MGW3: when the second connection starts, the overall throughput for UN1 is the sum of MGW1 and MGW3 flows.

Figure 6 reports the download performance between MAP1 and MGW1 when an intermediate node fails. In this example, we suppose that the node MR1 disappear after 20s. MAP1 reacts selecting a new best gateway, but existing tunnels stay active. The download through MGW1 starts again after about 10-15s and the user connection to the Internet does not break down. The 10s delay is solely due to olsrd configuration, and in particular it depends on time intervals between Hello and TC messages and on ETX window. We are also working on the OLSR timing and dynamics and preliminary tests indicate that a 2-3 seconds reconfiguration is possible in most situations.

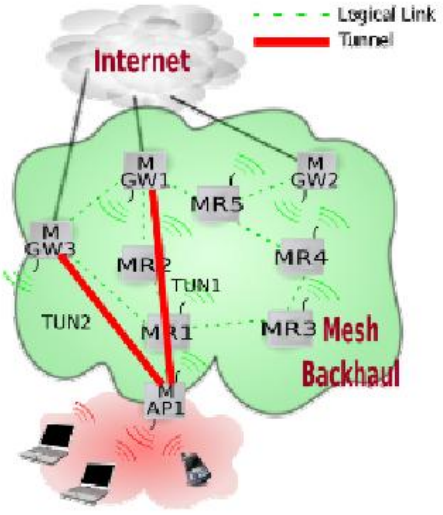

Figure 4: Base topology with a new gateway (MGW3) and multiple tunnels

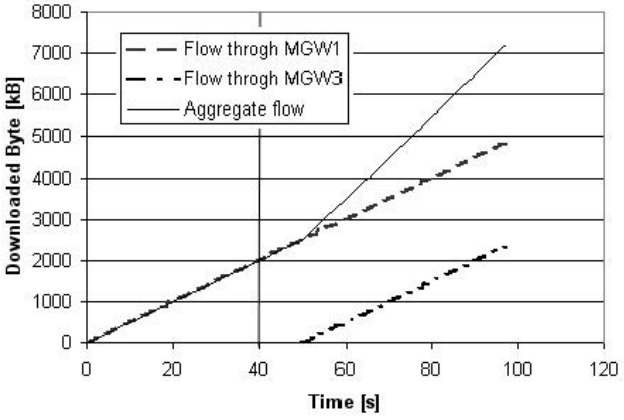

Figure 5: Cumulative downloads with MGW tunneling protection, without it the connection is broken because the MGW is changed during the connection.

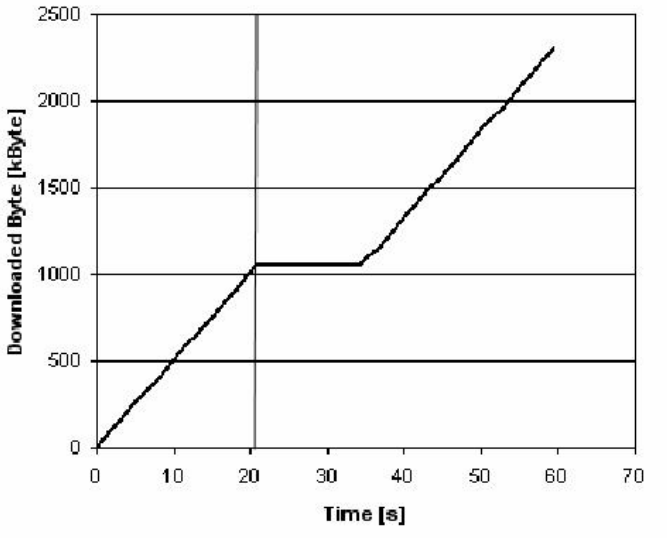

Figure 6: Download progress for TUN1 when MR2 breaks down

B. *Performance Test*

We made some tests on a commercial device equipped with two radio interfaces: the Essentia® Wifless® that is based on a Xscale® 425 NPU @ 533MHz and two Atheros® miniPCI cards.

We compare UDP throughput, evaluated by iperf packet generator. The test consisted in a comparison between the performance of the equipments with and without our software.

The test topology was based on three mesh nodes: a MGW and a MAP, interconnected through a MR. Each node has two independent radio interfaces that operate on different channels. In particular, MR routes received packets from MAP on the first radio interface to the MGW, that is interconnected through the second radio. The two wireless link (the first from the MAP to the MR and the second from MR to the MGW) used 5GHz channel 112 and 140; each link was about 400m length with about 40dB SNR.

Performances were evaluated generating UDP flows from a station that is connected to the MAP to a server behind MGW, that simulate an Internet host. For each UDP fragment size we generated a UDP flow with a throughput very close to the wireless channel limit. The duration of each flow was 13s and we evaluated performance from the fifth second to the tenth, to avoid transitory instability. We repeated each test 100 times to evaluated confidence intervals.

Figure 7 reports the measured throughput as a function of the UDP segment size. The throughput difference is not very large, and it is entirely due to CPU saturation, a consequence of user-space tunneling encapsulation. Figure 7 reports also error bars for 99,5% confidence level; error bars are very close to the average value: the relative error is less then 1%.

In order to verify if the performance loss is really due to the inefficiency in user-space encapsulation, we made some preliminary test using IP-within-IP managed by the Linux Kernel (hand configured iptables working in kernel-space). The performance is very close to the one obtained without the plug-in, indicating that the inefficiency is due to encapsulation and not to the remaining part of the architecture. Starting from this result, we are planning to optimize our module porting Encapsulator Thread in kernel-space, implementing the proper signaling between the olsrd deamon and the iptables.

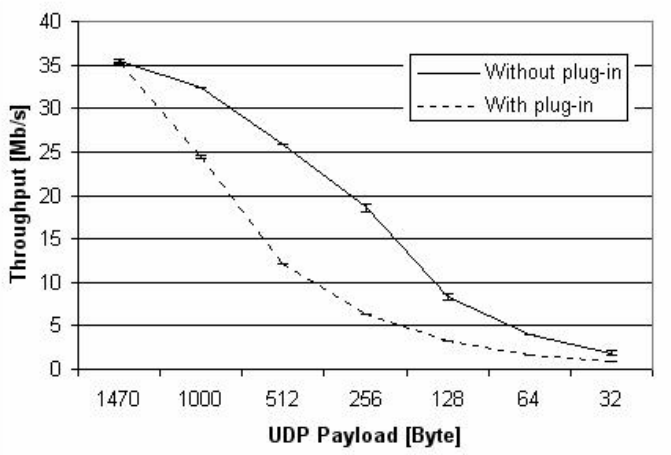

Figure 7: Throughput evaluated on Essentia® Wifless®

VI. Conclusions

In this paper we presented the design and initial implementation of an OLSR-based routing and handover management protocol tailored for the specific needs of 802.11 WMNs with dual-radio nodes, with particular attention to multi-homing strategies and support.

The goals of the protocol are: i) backward compatibility; ii) transparency with respect to user equipments; iii) seamless handover support; iv) provisioning service continuity in spite of gateways volatility and frequent topology rearrangement.

Experimentation both with User Mode Linux and with Wifeless® mesh routers prove the feasibility of the solution and also encouraging performances even with an "early beta" implementation, developed entirely in user space and without code optimization.

References

[1] http://pdos.csail.mit.edu/roofnet/doku.php

[2] http://tfa.rice.edu/

[3] http://wiki.freifunk.net/Kategorie:English

[4] http://wiki.ninux.org/

[5] A. Capone, S. Napoli, A. Pollastro, "MobiMESH: An Experimental Platform for Wireless MESH Networks with Mobility Support", In Proc. QShine 2006, Waterloo, Ontario, Canada, 7-9 August, 2006

[6] J. Bicket, D. Aguayo, S. Biswas, R. Morris, "Architecture and Evaluation of an Unplanned 802.11b Mesh Network", In Proc. MobiCom 2005, Cologne, Germany, 28 August-2 September, 2005

[7] V. Navda, A. Kashyap, S.R. Das, "Design and Evaluation of iMesh: an Infrastructure-mode Wireless Mesh Network", In Proc. WoWMoM 2005, Taormina, Italy, 13-16 June, 2005

[8] C. Perkins, E. Belding-Royer, S. Das, "Ad hoc on-demand distance vector (AODV) routing", IETF Internet RFC 3561, 2003

[9] T. Clausen, P. Jacquet, "Optimized link state routing protocol (OLSR)", RFC 3626, 2003

[10] http://www.olsr.org

[11] S. Tajima, T. Higashino, N. Funabiki, S. Yoshida, "An Internet Gateway Access-Point Selection Problem for Wireless Infrastructure Mesh Networks", In Proc. MDM 2006, Nara, Japan, 9-13 May, 2006

[12] Changui Shin, SungHo Kim, Sunshin An, "Stable Gateway Selection Scheme based on MANET with Internet", In Proc. ICCIT 2006, Dhaka, Bangladesh, 21-23 December 2006

[13] E. Nordstrom, P. Gunningberg, C. Tschudin, "Gateway Forwarding Strategies for Ad hoc Networks", In Proc. ADHOC 2004, Stockholm, Sweden, 4-5 May, 2004

[14] Y. Amir, C. Danilov, M. Hilsdale, R. Musaloiu-Elefteri, N. Rivera, "Fast Handoff for Seamless Wireless Mesh Networks", In Proc. ACM MobiSys 2006, Uppsala, Sweden, 19-22 June, 2006

[15] Y. Amir, C. Danilov, R. Musaloiu-Elefteri, N. Rivera, "An Inter-domain Routing Protocol for Multi-homed Wireless Mesh Networks", In Proc. WoWMoM 2007, Helsinki, Finland, 18-21 June 2007

[16] J. Chen, Y.Z. Lee, D. Maniezzo, M. Gerla, "Performance Comparison of AODV and OFLSR in Wireless Mesh Networks", In Proc. Med-Hoc-Net 2006, Lipari, Italy, June 14-17, 2006

[17] P.E. Engelstad, A. Tønnesen, A. Hafslund, G. Egeland, "Internet connectivity for multi-homed proactive ad hoc networks", In Proc. ICC 2004, Paris, France, 20-24 June, 2004