A Layer-2 Architecture for Interconnecting Multi-hop Hybrid Ad Hoc Networks to the Internet

E. Ancillotti
Dept. of Information Engineering
University of Pisa
Via Diotisalvi 2 - 56122 Pisa, Italy
Email: emilio.ancillotti@iet.unipi.it

R. Bruno, M. Conti, E. Gregori, A. Pinizzotto
IIT Institute
National Research Council (CNR)
Via G. Moruzzi, 1 - 56124 PISA, Italy
Email: {r.bruno,m.conti,e.gregori,a.pinizzotto}@iit.cnr.it

***Abstract*— Recently, the research on mobile ad hoc networks is departing from the view of stand-alone networks, to focus on hybrid self-organized network environments interconnected to the Internet. This type of networks is built on a mix of fixed and mobile nodes using both wired and multi-hop wireless technologies, and may be easily integrated into classical wired/wireless networking infrastructures. In this paper we design a lightweight and efficient architecture to build such a multi-hop hybrid ad hoc network, which will be used as a flexible and low-cost extension of traditional wired LANs. Our proposed architecture provides transparent global Internet connectivity and self-configuration to mobile nodes, without requiring configuration changes in the pre-existing wired LAN. Differently from most of the implemented solutions, which are based on complex IP-based mechanisms, such as Mobile IP, IP-in-IP encapsulation and IP tunneling, our proposed system operates below the IP level, and employs only layer-2 mechanisms. We have prototyped the core functionalities of our architecture, and we present several experimental results to verify the network performance constraints, and how different OLSR parameter settings impact on them.**

I. Introduction

A mobile ad hoc network (MANET) is a collection of mobile nodes connected together over a wireless medium, which self-organize into an autonomous multi-hop wireless network. Traditionally, MANETs have been considered as *stand-alone* networks, i.e., self-organized groups of nodes that operate in isolation in an area where deploying a networking infrastructure is not feasible due to practical or cost constraints (e.g., disaster recovery, battlefield environments). However, it is now recognized that the commercial penetration of the ad hoc networking technologies requires the support of an easy access to the Internet and its services. In addition, the recent advances in mobile and ubiquitous computing, and inexpensive, portable devices are further extending the application fields of ad hoc networking. As a consequence, nowadays, multi-hop ad hoc networks do not appear as isolate self-configured networks, but rather emerge as a flexible and low-cost extension of wired infrastructure networks, coexisting with them. Indeed, a new class of networks is emerging from this view, in which a mix of fixed and mobile nodes interconnected via heterogeneous (wireless and wired) links forms a multi-hop hybrid ad hoc network integrated into classical wired/wireless infrastructure-based networks [1].

In this paper, we propose and evaluate a practical architecture to build multi-hop hybrid ad hoc networks used to extend the coverage of traditional wired LANs, providing mobility support for mobile/portables devices in the local area environment. More precisely, we envisage a hybrid network environment in which wired and multi-hop wireless technologies transparently coexist and interoperate. In this network, separated group of nodes without a direct access to the networking infrastructure form *ad hoc "islands"*, establishing multi-hop wireless links. Special nodes, hereafter indicated as *gateways*, having both wired and wireless interfaces, are used to build a wired backbone interconnecting separated ad hoc components. To ensure routing between these ad hoc parts, a proactive ad hoc routing protocol is implemented on both gateways' interfaces. In addition, the gateways use their wired interfaces also to communicate with static hosts belonging to a wired LAN. The network resulting from the integration of the hybrid ad hoc network with the wired LAN is an *extended* LAN, in which static and mobile hosts transparently communicate using traditional wired technologies or ad hoc networking technologies.

In this work we specifically address several architectural issues that arise to offer IP basic services, such as routing and Internet connectivity, in the extended LAN. First, we propose a dynamic protocol for the self-configuration of the ad hoc nodes, which relies on DHCP servers located in the wired part of the network, and it does not require that the ad hoc node to be configured has a direct access to the DHCP server. In addition, we design innovative solutions, which exploit only *layer-2* mechanisms as the ARP protocol, to logically extend the wired LAN to the ad hoc nodes in a way that is transparent for the wired nodes. More precisely, in our architecture the extended LAN appears to the external world, i.e., the Internet network, as a single IP subnet. In this way, the hosts located in the Internet can communicate with ad hoc nodes inside the extended LAN as they do with traditional wired networks. Previous solutions to connect ad hoc networks to the Internet have proposed to use access gateways that implement Network Address Translator (NAT) [2] or a Mobile IP Foreign Agent (MIP-FA) [3]. However, such approaches are based on complex IP-based mechanisms originally defined for the wired Internet, like IP-in-IP encapsulation and IP tunneling, which may introduce significant overheads and limitations, as discussed in depth in the following sections. On the other hand, the architecture we propose in this paper is a lightweight and efficient solution that avoids these overheads operating below the IP level. By positioning our architecture at the layer 2 (data link layer), we may avoid undesired and complex interactions with the IP protocol and provide global Internet connectivity and node self-configuration in a very straightforward way.

In the past, other architectures have been proposed to provide ad hoc support below IP. For example, in [4] label switching was employed to put routing logic inside the wireless network card. More recently, the LUNAR [5] ad hoc routing framework and the Mesh Connectivity Layer (MCL) [6] have been proposed. These solutions locate the ad hoc support between the layer 2 (data link layer) and layer 3 (network layer). This "layer 2.5" is based on *virtual* interfaces that allow abstracting the ad hoc protocols from both the specific hardware components and network protocols. However, this interconnection layer requires its own naming and addressing functionalities distinct from the layer-2 addresses of the underlying physical devices. This may significantly increase the packet header overheads. On the contrary, our proposed architecture is totally located inside layer 2, reducing implementation complexity and ensuring minimal additional overheads.

We have prototyped the main components of our architecture in a general and realistic test-bed, in which we have carried out various performance measurements. The experimental results show the performance constraints with mobility and Internet access, and indicate that an appropriate tuning of the routing protocol parameter may significantly improve the network performance. The rest of this paper is organized as follows. Section II introduces our network model. In Section III, we discuss available solutions for Internet connectivity and node self-configuration in MANETs. Section IV outlines the relevant protocols used in our architecture. In Section V, we describes the layer-2 architecture we propose to build hybrid ad hoc networks interconnected to the Internet. Section VI shows experimental results on the performance constraints of our solution. Finally, Section VII draws concluding remarks and discusses future work.

II. Network Model

Figure 1 illustrates the reference network model we assume in our architecture. We consider a full-IP network in which all the traffic is transported in IP packets. In this network, mobile/portable nodes far away from the fixed networking infrastructure establish multi-hop wireless links to communicate (e.g., using IEEE 802.11 technology). As shown in the figure, gateways, i.e., nodes with two interfaces - both wired and wireless - are used to connect the ad hoc components to a wired LAN (e.g., an Ethernet-based LAN). In our architecture, it is allowed the multi-homing, i.e., the presence of multiple gateways within the same ad hoc component. Consequently, specific mechanisms are required to support the handoff between gateways without TCP-connection breaks. In general, between pairs of gateways in radio visibility of each other, two direct links can be established, both wired and wireless. However, in our model we assume that the gateways always use the wired link to communicate. The motivations behind this requirement will be clearly discussed in Section V. However, this is a quite reasonable assumption, since wired links have higher bandwidth than wireless links, and the routing protocol should assign them a lower link cost.

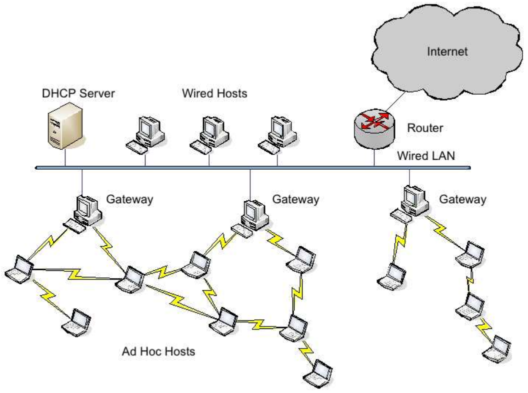

Fig. 1. Reference Network Model.

The wired LAN is interconnected to the external Internet through a default router R. In addition, one or more DHCP servers are located in the wired LAN to allocate network addresses to hosts. In the following sections, we will explain how these DHCP servers could be used to assign IP configuration parameters also to the ad hoc nodes. For the purpose of simplicity, we assume that all the IP addresses are allocated from the same IP address block $\mathrm{IP_S/L}$. According to standard notation, $\mathrm{IP_S}$ indicates the network prefix, and L is the network mask length, expressed in bits (e.g., $\mathrm{IP_S/L} = X.Y.96.0/22$). Assuming that the extended LAN adopts a unique network address implies that the extended LAN appears to the external world, i.e., the Internet network, as a single IP subnet.

Standard IP routing is used to connect the extended LAN to the Internet. However, a specific ad hoc routing protocol is needed to allow multi-hop communications among the ad hoc nodes. In this work we decided to use a proactive routing protocol as the ad hoc routing algorithm (such as the Optimized Link State Routing (OLSR) protocol [7] or the Topology Dissemination Based on Reverse-Path Forwarding (TBRF) routing protocol [8]). The motivation behind this design choice is that proactive routing protocols usually support gateways, allowing these nodes to use special routing messages to set up default routes in the ad hoc network. Indeed, default routes are an efficient mechanism to forward traffic that does not have an IP destination locally known to the ad hoc network. In addition, proactive routing protocols, adopting classical link state approaches, build the complete network-topology knowledge in each ad hoc node. This topology information could significantly simplify the operations needed to acquire

Internet connectivity. In this work, the reference ad hoc routing algorithm is OLSR, but our architecture is general and it is equally applicable to other proactive routing protocols.

III. Related Work

The implemented solutions to provide Internet connectivity in MANETs are mainly based on two different mechanisms.

One approach is to set up a Mobile IP Foreign Agent (MIP-FA) in the gateway and to run Mobile IP [3] in the MANET. In this way, the ad hoc node may register the foreign agent care-of-address with its Home Agent (HA). Whenever an ad hoc node MN wants to contact an external host X, it uses its home address (i.e., a static IP address belonging to its home network) as source address. As a consequence, the return traffic is routed to the home network through standard IP routing. The HA intercepts the traffic, encapsulating it using the care-of-address, and it tunnels the encapsulated packets to the FA. The FA removes the outer IP header and delivers the original packets to the visiting host MN. Different versions of this approach have been proposed and implemented for proactive [9] and reactive [10] ad hoc networks. A drawback of these solutions is that they require significant changes in the Mobile IP implementation since the FA and the mobile node cannot be considered on the same link. Moreover, the mobile node has to be pre-configured with a globally routable IP address as its home address, limiting both the ability of forming totally self-configuring and truly spontaneous networks, and the applicability of these schemes.

An alternative solution to interconnect MANETs to the Internet is to implement a Network Address Translation (NAT) [2] on the gateway. In this way, the gateway may translate the source IP address of outgoing packets from the ad hoc nodes with an address of the NAT gateway, which is routable on the external network. The return traffic is managed similarly, with the destination IP address (i.e., the NAT-gateway address) replaced with the IP address of the ad hoc node. NAT-based solutions have been designed for both proactive [11] and reactive [12] ad hoc networks. NAT-based mechanisms appear as easier solutions than MIP-FA-based schemes to provide Internet access to MANETs. However, a problem that arises with NAT-based solutions is multi-homing, i.e., the support of multiple gateways in the same MANET. Indeed, to avoid session breakages it is necessary to ensure that all the packets from the same session are routed over a specific gateway. A proposed solution to this issue is to explicitly tunnel all the outgoing traffic from the same communication session destined to the external network to one of the available gateways, instead of using default routes. A limitation of this strategy is the additional overhead introduced by the IP-in-IP encapsulation. Moreover, the ad hoc nodes should be provided with the additional capability of explicitly discovering the available gateways. This would eventually require extensions to the ad hoc routing protocols.

Both the two classes of solutions discusses above implicitly assume that either there is a dynamic host configuration protocol designed to configure the nodes such as to properly working in the MANET, or the ad hoc nodes are configured *a priori*. Indeed, a node in an IP-based network requires a unique IP-based address, a common netmask and, eventually, a default gateway. In traditional networks, hosts rely on centralized servers like DHCP [13] for configuration, but this cannot be easily extended to MANETs because of their distributed and dynamic nature. However, various protocols have been proposed recently in literature for the purpose of address self-configuration in MANETs. In general, with protocols using stateless approaches nodes arbitrarily select their own address, and a Duplicate Address Detection (DAD) procedure is executed to verify its uniqueness and resolve conflicts. On the other hand, protocols based of stateful approaches execute distributed algorithms to establish a consensus among all the nodes in the network on the new IP address, before assigning it. The protocols proposed in [14] and [15] are examples of the latter and former approach, respectively, while [16] presents a general overview of the several solutions currently available. Generally, all these protocols assume reliable flooding in order to synchronize nodes' operations and resolve inconsistencies in the MANET, but this is difficult to be guaranteed in ad hoc networks. Another main limitation of these solutions is that they are designed to work in *stand-alone* MANET, while no protocols have been devised to take fully advantage of the access to external networks. In addition, the problems of selecting a unique node address, routing the packets and accessing the Internet are still separately addressed, while a unified strategy may be beneficial, reducing complexities and overheads.

IV. Protocol Descriptions

This section gives a short description of the protocols, which our architecture is based on.

A. OLSR

The OLSR protocol [7], being a link-state proactive routing protocol, periodically floods the network with route information, so that each node can locally build a routing table containing the complete information of routes to all the nodes in the ad hoc network running on their interfaces the OLSR protocol. The OLSR routing algorithm employs an efficient dissemination of the network topology information by selecting special nodes, the multipoint relays (MPRs), to forward broadcast messages during the flooding process. The link state reports, which are generated periodically by MPRs, are called Topology Control (*TC*) messages. MPRs grant that TC messages will reach all 2-hop neighbors of a node. In order to allow the injection of external routing information into the ad hoc network, the OLSR protocol defines the Host and Network Association (*HNA*) message. The *HNA* message binds a set of network prefixes to the IP address of the node attached to the external networks, i.e., the gateway node. In this way, each ad hoc node is informed about the network address and netmask of the network that is reachable through each gateway. In other words, the OLSR protocol exploits the mechanism of *default routes* to advertise Internet connectivity.

For instance, a gateway that advertises the $0.0.0.0/0$ default route, will receive all the packets destined to IP addresses without a known route on the local ad hoc network.

B. ARP Protocol

IP-based applications address a destination host using its IP address. On the other hand, on a physical network individual hosts are known only by their physical address, i.e., MAC address. The ARP protocol [17] is then used to translate, inside a physical network, an IP address into the related MAC address. More precisely, the ARP protocol broadcasts the *ARP_Request* message to all hosts attached to the same physical network. This packet contains the IP address the sender is interested in communicating with. The target host, recognizing that the IP address in the packet matches its own, returns its MAC address to the requester using an unicast *ARP_Reply* message. To avoid continuous requests, the hosts keep a cache of ARP responses.

In addition to these basic functionalities, the ARP protocol has been enhanced with more advanced features. For instance in [18] it has been proposed the *Proxy-ARP* mechanism, which allows constructing local subnets. Basically, the Proxy ARP technique allows one host to answer the ARP requests intended for another host. This mechanism is particularly useful when a router connects two different physical networks, say *NetA* and *NetB*, belonging to the same IP subnet. By enabling the Proxy ARP on the router's interface attached to *NetB*, any host A in *NetA* sending an ARP request for a host B in *NetB*, will receive as response the router's MAC address. In this way, when host A sends IP packets for host B, they arrive to the router, which will forward such packets to host B.

V. Proposed Architecture

Our design goal in the definition of the rules and operations of the proposed architecture is to provide transparent communications between static nodes (using traditional wired technologies) and mobile nodes (using ad hoc networking technologies), employing mechanisms that run below the IP layer. As discussed in the introduction, in this work we address two relevant issues: node self-configuration and global Internet connectivity.

A. Ad Hoc Node Self-configuration

The main obstacle to use a DHCP server for self-configuration of ad hoc nodes is that the DHCP server may be not reachable to the new node, due to mobility or channel impairments. In addition, the ad hoc nodes may need multi-hop communications to reach the DHCP server, but a unique address is necessary to execute ad hoc routing algorithms capable of establishing such communications. To solve these problems, we assume that the DHCP servers are located only in the wired part of the network, while in the ad hoc part of the network we implement dynamic *DHCP Relay* agents. These are special relay nodes passing DHCP messages between DHCP clients and DCHP servers that are on different networks. As illustrated in Figure 2, when a new mobile host i not yet configured attempts joining the ad hoc part of the extended LAN, it broadcasts a special message, the *Neighbor_Req* message. At least one neighbor that is already configured, i.e., it has joined the ad hoc network, will respond with a *Neighbor_Reply* message. Node i selects one of the responders j as intermediary in the process of address resolution. Then, node i sends a *Conf_Req* message to the chosen node j that replies with a *Conf_Ack* message to inform node i that it will execute on its behalf the process of acquiring the needed IP configuration parameters (i.e., node j acts as a *proxy* for the node i). In fact, on receiving the *Conf_Req* message from node i, node j activates its internal DCHP Relay agent, which issues an unicast *DHCP_Request* to one of the available DHCP servers. The DHCP server receiving the request, will answer to the DHCP Relay with a *DHCP_Ack*, containing the IP configuration parameters. The configuration process is concluded when the DHCP Relay forwards the *DHCP_Ack* message to the initial node i that is now configured and can join the network. After joining the network, node i may also turn itself into a DHCP Relay for the DHCP server from which it received the IP configuration parameters, letting other nodes to subsequently joining the ad hoc component. Finally, it is worth noting that it is not needed any initialization procedure for the ad hoc network, because the gateways are directly connected to the wired LAN and can broadcast a *DHCP_Discover* message to locate available servers. In this way, the first mobile node to enter the ad hoc network may find at least one gateway capable of initiating the illustrated configuration process.

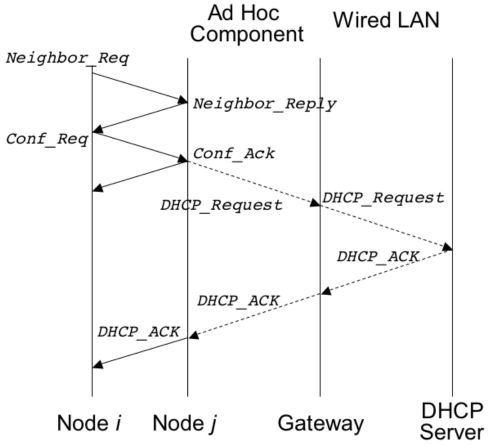

Fig. 2. Message exchanges during the ad hoc node self-configuration.

Our proposed node self-configuration mechanism is somehow similar to the one described in [15]. In that paper, a preliminary message handshake was used to discover a reachable MANET node that could act as initiator of the configuration process. On the contrary, in our solution the initiator node exploits the resources of the external wired network to which the ad hoc component is connected, to perform the IP address resolution.

B. *Global Internet Connectivity*

Our design goal is to support Intranet connectivity (i.e., communications with nodes inside the same IP subnet) and Internet connectivity (i.e., communications with nodes of external IP networks) for the mobile nodes, without any configuration change in the pre-existing wired LAN. The assumption that we take as starting point in our proposal is that the mobile nodes are configured with an IP address belonging to the same IP subnet of the wired LAN. This is achieved using the mechanism described in Section V-A.

In the following we will separately explain how the proposed architecture ensures connectivity for outgoing and incoming traffic.

1) Connectivity for Outgoing Traffic.: As outlined in Section IV-A, the OLSR protocol builds the routing tables with entries that specify the IP address of the next-hop neighbor to contact to send a packet destined to either another host or subnetwork. More precisely, to send a packet to a destination IP address, the mobile host searches for the longest IP prefix in the routing table matching the destination IP address. The matching routing table entry provides the next hop to send the packet. Since the gateways advertise $0.0.0.0/0$ as default route, all packets destined for IP addresses without a specific route on the ad hoc network, will be routed on a shortest-hop basis to the nearest gateway and forwarded to the Internet. However, using $0.0.0.0/0$ as default route for outgoing packets, introduces an inconsistency when a mobile host sends IP packets to a wired host inside the LAN. To explain this problem let us consider the simple network configuration depicted in Figure 3. For illustrative purposes we assume that the IP subnet of the extended LAN is $\mathrm{IP_S/L}{=}X.Y.96.0/22$[1]. If the mobile node N ($\mathrm{IP_N} = X.Y.97.151/22$) wants to deliver packets to the wired node H ($\mathrm{IP_H} = X.Y.99.204/22$), the routing table lookup on node N will indicate that the node H is connected to the same physical network of node N's wireless interface. This will result in a failed ARP request for the $\mathrm{IP_H}$ address. To resolve this inconsistency, we will exploit the properties of the IP longest-matching rules. More precisely, we split the original IP subnet into two consecutive smaller subnets $\mathrm{IP_{SL}/(L{+}1)}$ and $\mathrm{IP_{SU}/(L{+}1)}$, such as to have that the union of these two sets is equal to $\mathrm{IP_S/L}$. In the considered case $\mathrm{IP_{SL}/(L{+}1)}{=}$ $X.Y.96.0/23$ and $\mathrm{IP_{SU}/(L{+}1)} = X.Y.98.0/23$[2]. Then, we configure all the gateways in such a way that they announce, through the HNA messages, also the connectivity to these two subnetworks. In this way, each mobile host will have, for any host on the local wired LAN, a routing table entry with a more specific network/mask than the one related to its wireless interface. To better clarify this point, let us consider the node N's routing table as shown in Table I. The entries 8, 9, and 11 are the ones induced by the HNA messages arrived from GW1. The entry 10 is automatically set up by the operating system when the wireless interface is configured with the IP parameters. However, when searching the routing table for matching the $\mathrm{IP_H}$ address, node N will found the routing entry 9 more specific than entry 10. Consequently, the longest-match criterion applied to the routing table lookup, will result in node N correctly forwarding traffic to gateway GW1 (i.e., the nearest one) to reach node H.

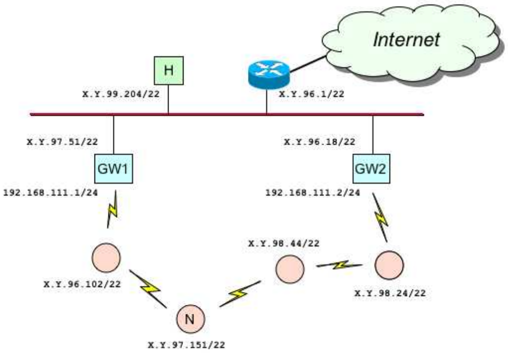

Fig. 3. Illustrative network configuration.

TABLE I
NODE N'S ROUTING TABLE.

Entry	*destination*	*next hop*	*metric*	*interface*
1	$X.Y.97.51/32$	$X.Y.96.102$	2	$eth0$
2	$X.Y.96.102/32$	0.0.0.0	1	$eth0$
3	$X.Y.98.44/32$	0.0.0.0	1	$eth0$
4	$X.Y.98.24/32$	$X.Y.98.44$	2	$eth0$
5	$X.Y.96.18/32$	$X.Y.96.102$	3	$eth0$
6	192.168.111.1/24	$X.Y.96.102$	2	$eth0$
7	192.168.111.2/24	$X.Y.96.102$	3	$eth0$
8	$X.Y.96.0/23$	$X.Y.96.102$	2	$eth0$
9	$X.Y.98.0/23$	$X.Y.96.102$	2	$eth0$
10	$X.Y.96.0/22$	0.0.0.0	0	$eth0$
11	0.0.0.0/0	$X.Y.96.102$	2	$eth0$
12	127.0.0.0/8	127.0.0.1	0	$l0$

The mechanism described above resolves any eventual IP inconsistency that could occur in the mobile hosts, but it may cause problems for the gateways. In fact, being part of the ad hoc component, the gateways will receive HNA messages sent by other gateways, setting up the additional routing entries advertised in these messages. However, when a gateway wants to send packets to a wired host on the local wired LAN (e.g., node H), the routing table lookup will choose one of these two entries, instead of the entry related to its wired interface (i.e., $X.Y.96.0/22$). The effect is that the IP packet will loop among the GW nodes until the TTL expires, without reaching the correct destination H. To resolve this problem, we statically add in each gateway two further routing entries in addition to the one related to the default router $X.Y.96.1$. These two additional entries have the same network/mask as the two announced in the HNA messages, but with lower

[1] On the gateways' wireless interfaces we set up private IP addresses to save address space. In this way, the gateways are globally reachable using the IP address on their wired interfaces.

[2] It is straightforward to observe that this operation is always feasible, at least for $L < 32$.

metric. Again, to better clarify the routing operations, let us consider the illustrative example shown in Figure 3. In Table II we have reported the GW1's routing table. In this example, $eth0$ is the GW1's wireless interface and $eth1$ is the GW1's wired interface. When gateway GW1 wants to send packets to node H, it will found two routing table entries matching the same number of bits of node H's IP address. These are entry 9 (derived from HNA messages received from GW2) and entry 11 (statically configured on the gateway). However, entry 11 has a lower metric than entry 9 (i.e., metric 0 against metric 1). As a consequence, the packets destined to host H can be correctly forwarded to the host H on the local wired LAN through the GW1's wired interface.

TABLE II
GW1's ROUTING TABLE.

Entry	*destination*	*next hop*	*metric*	*interface*
1	$X.Y.96.102/32$	$0.0.0.0$	1	$eth0$
2	$X.Y.97.151/32$	$X.Y.96.102$	2	$eth0$
3	$X.Y.98.44/32$	$X.Y.96.18$	3	$eth1$
4	$X.Y.98.24/32$	$X.Y.96.18$	2	$eth1$
5	$X.Y.96.18/32$	$0.0.0.0$	1	$eth1$
6	$192.168.111.2/24$	$X.Y.96.18$	1	$eth1$
7	$192.168.111.0/24$	$0.0.0.0$	0	$eth0$
8	$X.Y.96.0/23$	$X.Y.96.18$	1	$eth1$
9	$X.Y.98.0/23$	$X.Y.96.18$	1	$eth1$
10	$X.Y.96.0/23$	$0.0.0.0$	0	$eth1$
11	$X.Y.98.0/23$	$0.0.0.0$	0	$eth1$
12	$X.Y.96.0/22$	$0.0.0.0$	0	$eth1$
13	$0.0.0.0/0$	$X.Y.96.1$	0	$eth1$
14	$0.0.0.0/0$	$X.Y.96.18$	1	$eth1$
15	$127.0.0.0/8$	$127.0.0.1$	0	$l0$

2) Connectivity for Incoming Traffic.: A mechanism is required to ensure that the return traffic coming from hosts on the local wired LAN or from the Internet (through the default LAN router, as shown in Figure 1), gets correctly routed to the mobile hosts. Our basic idea is to introduce specific Proxy ARP functionalities into each gateway, in such a way that the gateways can hide the ad-hoc node identity on the wired physical network, which the gateways are connected to. Thus, all mobile nodes located in the ad hoc component will appear to wired hosts as being one IP-hop away. Internally to the ad hoc component, the ad hoc routing protocol will transparently provide the multi-hop connectivity and the mobility support. This is somehow similar to what is implemented in the LUNAR framework [5], in which the entire ad hoc network appears as a single virtual Ethernet interface.

In our proposed solution, a Proxy ARP server runs on the wired interfaces of each gateway. The Proxy ARP server periodically checks the gateway's routing table and ARP table, such as to publish the MAC address of the gateway's wired interface for each IP address having an entry in the routing table with a netmask $255.255.255.255$, and the next hop on the gateway's wireless interface. The former condition is verified only by mobile hosts that have joined the ad hoc network. The latter condition implies that the gateway can deliver traffic to that node only over multi-hop paths not traversing other gateways[3]. Thus, it is highly probable that the considered gateway is the default gateway selected by that ad hoc node. To illustrate how the proposed mechanism works, let us consider the network in Figure 3. When a node on the wired local LAN (e.g., node H) wants to send packets to an ad hoc node (e.g., node N), it assumes that the ad hoc node is on the same physical network. Hence, node H checks its ARP table for IP-MAC mapping and, if it is not present, it sends an ARP request. The gateway GW1 fulfills the previously defined conditions (i.e., node N's IP address has an entry in the GW1's routing table with a netmask $255.255.255.255$, which is related to its wireless interface), while GW2 does not. Consequently, only GW1 is allowed by the Proxy ARP server to answer with an ARP reply. This ARP reply will insert the mapping [node N's IP address - MAC address of GW1's wired interface] into the node H's ARP table. Thus, the packets sent from node H to node N will be delivered to GW1, which will forward them to node N. On the other hand, node N will reply to node H using GW1, as indicated by its routing table (see Table I).

There are some network configurations where asymmetric routing may occur, i.e., the forward path is different from the return path. For instance, let us consider the case in which node N is in radio visibility of two gateways GW1 and GW2. In this situation, the OLSR routing algorithm will randomly select one of these gateways as default gateway for node N. However, both gateways are allowed to send ARP replies for ARP requests issued by node H for the node N's IP address. In this case, the wired node H will update its ARP table using the information delivered in the last received ARP reply. Let us assume that GW1 is the default gateway for node N, but GW2 has sent the last ARP reply to node H. In this case, node H sends the traffic destined to node N to GW2, which routes it to node N. On the other hand, node N sends packets destined to node H to GW1, which forwards them to node H. It is important to note that asymmetric paths are not by themselves a problem. Indeed, both node N and H correctly receive and send their packets. In addition the asymmetric routing occurs only in symmetric topologies. Thus, it is reasonable to assume, in this local environment, that both paths are characterized by similar delays.

3) Mobility Support.: In general, solutions to support Internet connectivity for ad hoc networks, which are based on gateways, experience TCP-session breaks when the default route changes, depending on dynamics and mobility in the network. To avoid that TCP sessions break, in [12] it was proposed to replace default routes with explicit tunneling between the mobile nodes and the gateways. However, this complicates significantly the implementation and introduces relevant overheads. On the contrary, in our architecture the mobility is supported in a transparent way for the higher

[3] It is worth reminding that gateways are always interconnected using their wired interfaces. Hence, a route to reach a mobile node can traverse two gateways only if one of the link along the path is a wired link. In this case the farthest gateway will have the next-hop routing entry for that mobile node on its wired interface.

protocol layers. Indeed, the only effect of changing the default gateway for node N, is that the node N's outgoing traffic is routed towards the new gateway (e.g., GW2), while the initial gateway (e.g., GW1) continues to receive the incoming traffic and to forward it to node N. This results into asymmetric routing. However, this asymmetry can be easily removed by using an advanced feature of the ARP protocol. More precisely, when GW2 becomes aware that the next hop for the node N switches from its wired interface to its wireless interface, it generates a *Gratuitous ARP* on the wired interface for node N's IP address. This will update the ARP table in all of the wired hosts that have an old entry for the node N's IP address, which was mapped with the MAC address of GW1's wired interface. This action restores a symmetric path for the active packet flows destined to and/or originated from node N.

VI. Experimental Results

We have prototyped the core functionalities of our architecture. In particular, we have developed the software components described in Section V-B, concerning the support of Internet and Intranet connectivity for the ad hoc nodes. Currently, we are completing the implementation of the modifications to the DHCP Relay agents described in Section V-A. For these reasons, in the following we will show experimental results measuring the network performance with mobility and Internet access, while we left for further work the testing of the performance (such as address allocation latency and communication overheads) of the proposed node self-configuration scheme.

In our test-beds we have used *IBM R-50* laptops with *Intel Pro-Wireless 2200* as integrated wireless card. We have also used the OLSR_UniK implementation for Linux in version 0.4.8 [19]. The installed Linux kernel distribution was 2.6.9. The ad hoc nodes are connected via IEEE 802.11b wireless links, transmitting at the maximum rate of 11 Mbps. To generate the asymptotic UDP and TCP traffic during the experiments we used the *iperf* tool[4]. More precisely, the iperf server (termination of the traffic sessions) runs in a static host in the wired LAN, while iperf clients (originators of traffic sessions) have been set up on the mobile nodes. If not otherwise specified, the packet size is constant in all the experiments and the transport layer payload is equal to 1448 bytes. Differently from other studies [11], in which the network topology was only emulated by using the *IP-tables* feature of Linux, our experiments were conducted in realistic scenarios, with hosts located at the ground floor of the CNR building.

A. Performance Constraints of Internet Access

To measure the performance constraints in case of Internet access, we executed several experiments in the test-bed shown in Figure 4. The distances between the ad hoc nodes were set up in such a way to form a 4-hop chain topology with high-quality wireless links. The first set of experiments was conducted to evaluate the impact on the UDP and TCP throughput of the number of wireless hops traversed in the ad hoc network to reach the gateway. During these tests all the OLSR configuration parameters have been set up according to the default values indicated in the RFC specification [7]. Figure 5 and Figure 6 show the UDP and TCP throughput, respectively, obtained during a single experiment, as a function of the time and for different chain lengths. Several observations can be derived from the shown experimental results. First, we can note that the maximum UDP throughput is always greater than the maximum TCP throughput, for every network configuration. This is obviously due to the additional overheads introduced by the TCP return traffic, which consists of TCP ACK packets. In addition, as expected, the longer the route, the lower is the peak throughput achieved by the session flow (both TCP and UDP). The figures show also that, although the nodes are static, the throughput is not stable, but both UDP and TCP flows could be in a stalled condition for several seconds. An initial explanation of this route instability is that losses of routing control frames can induce the loss of valid routes. Indeed, the routing control frames are broadcast frames, which are neither acknowledged nor retransmitted, hence they are more vulnerable to collisions and channel errors than unicast frames. However, a careful analysis of the routing log files has pointed out another relevant condition that contributes to the route instability in our static network. Indeed, we discovered that the OLSR protocol implements an over pessimistic estimation of the link quality that may cause to consider as lost a link that is overloaded. More precisely, each node keeps updating a $link\_quality$ value for each neighbor interface. Every time an OLSR packet is lost $link\_quality = (1-\alpha)\cdot link\_quality$[5], while every time an OLSR packet is correctly received $link\_quality = (1-\alpha)\cdot link\_quality + \alpha$, where the α value is the smoothing factor of the estimator. The OLSR specification suggests as default configuration $\alpha = 0.5$. This implies that the $link\_quality$ value is halved after each OLSR packet loss. The $link\_quality$ parameter is used to estimate the link reliability, according to a procedure denoted as *link hysteresis* [7].

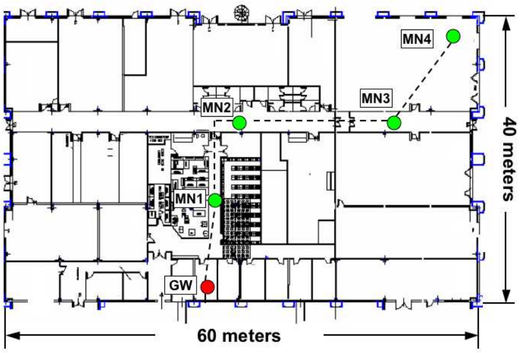

Fig. 4. Trial scenario for testing Internet access using a chain network.

[4] `http://dast.nlanr.net/Projects/Iperf/`.

[5] To identify the loss of an OLSR packet two mechanisms are used: 1) tracking the sequence numbers of the received OLSR packets, or 2) monitoring OLSR packet receptions during an *HELLO* emission interval [7].

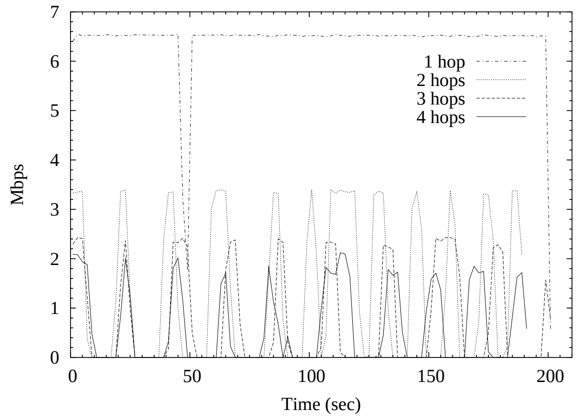

Fig. 5. Throughput of a single UDP flow for different chain lengths.

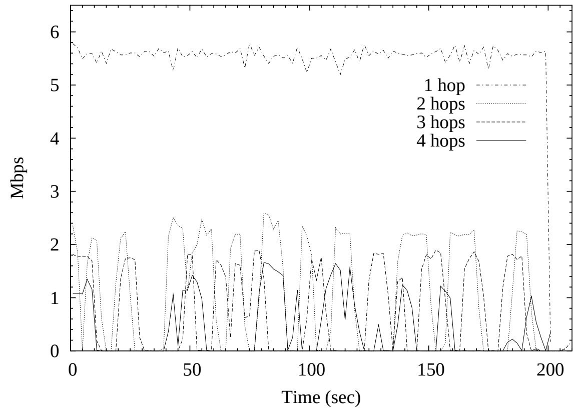

Fig. 6. Throughput of a single TCP flow for different chain lengths.

More precisely, the value of the $link\_quality$, is compared with two thresholds, called $HYST\_THRESHOLD\_LOW$ and $HYST\_THRESHOLD\_UP$. When $link\_quality < HYST\_THRESHOLD\_LOW$, the link is considered as *pending*, i.e., not established. A pending link is not completely dropped because the link information is still updated for each *HELLO* message received. However, a pending link is not a valid link when computing routing tables. In addition, a pending link can be considered again as established only when $link\_quality > HYST\_THRESHOLD\_UP$. The OLSR specification suggests as default configuration $HYST\_THRESHOLD\_LOW = 0.3$ and $HYST\_THRESHOLD\_UP = 0.8$. According to these values and to the scaling factor α, even a perfect link (i.e., a link with $link\_quality = 1$) will be purged from the routing tables when two consecutive OLSR packets are lost. We argue that the standard setting of the hysteresis parameters introduces a critical instability in the routing tables, because it is not infrequent to loose broadcast packets (as the OLSR packets are) when the channel is overloaded.

To verify our claim we have carried out a second set of experiments in the same network configuration depicted in Figure 4, disabling the OLSR hysteresis process. To provide statistically correct results, we have replicated each experiment five times. Tables III and Table IV show the average and standard deviation (in parenthesis) values of the measured throughputs for the UDP and TCP case, respectively. From the results we observe that the throughput performances are significantly improved, with the improvement for a 4-hop chain reaching 119% in the UDP case and 82% in the TCP case. The study of routing table logs clearly indicates that these throughput increases are due to an improvement in the route stability with less frequent declarations of link drops due to erroneous estimations of links' reliability. It is worth pointing out that this issue has not been identified in previous experimental studies because either the multi-hop communications where only emulated [12], or the channel was loaded with low-intensity *ping* traffic [20].

TABLE III
UDP THROUGHPUT IN A CHAIN NETWORK, WITH AND WITHOUT HYSTERESIS.

	UDP		
	HYST	NO HYST	
1 *hop*	6.124Mbps (304Kbps)	6.363Mbps (393Kbps)	$+4\%$
2 *hops*	1.252Mbps (55Kbps)	2.501Mbps (57Kbps)	$+100\%$
3 *hops*	700.4Kbps (60Kbps)	1.306Mbps (87Kbps)	$+86\%$
4 *hops*	520.6Kbps (54Kbps)	1.141Mbps (56Kbps)	$+119\%$

TABLE IV
TCP THROUGHPUT IN A CHAIN NETWORK, WITH AND WITHOUT HYSTERESIS.

	TCP		
	HYST	NO HYST	
1 *hop*	5.184Mbps (335Kbps)	5.172Mbps (393Kbps)	$\approx=$
2 *hops*	956.1Kbps (123Kbps)	1.517Mbps (57Kbps)	$+58\%$
3 *hops*	638.1Kbps (149Kbps)	891.7Kbps (77Kbps)	$+39\%$
4 *hops*	345.9Kbps (47Kbps)	631.2Kbps (74Kbps)	$+82\%$

In addition to the hysteresis process, the OLSR protocol employs several other mechanisms, as the link sensing, neighbor detection and topology discovery, which significantly affect the route stability. Indeed, recent works [20], [21] have investigated how the setting of the classical OLSR routing parameters may affect the network performances. However, these works have specifically focused on the time required for route recalculation after a link drop due to node mobility. On the contrary, to conclude this section we will analyze the impact of different OLSR parameter settings on the performance limits of Internet access in static network configurations. More precisely, each OLSR packet, and the information it delivers, has a fixed validity time. For instance, the information provided in a *HELLO* message is considered valid for a *NEIGHB_HOLD_TIME*. This implies that a node detects a link loss with a neighbor from the lack of *HELLO* messages during a *NEIGHB_HOLD_TIME*. A similar check is performed for the *TC* messages, whose validity time is *TOP_HOLD_TIME*, and for the *HNA* messages, whose validity time is *HNA_HOLD_TIME*. A possible strategy to avoid that links and routes are dropped from the routing tables because the related information has not been refreshed within the corresponding timeout, is to increase the frequency used to generate OLSR packets. This may increase the probability that at least one new OLSR packet is received before its validity time expires. The drawback of this approach is that the more frequent the OLSR protocol generates control messages, the higher is the routing overheads. To quantify the trade-off between routing overhead increases and route stability improvements, and how this impacts network performance, we have carried out a set of experiments in a 3-hop chain using the OLSR parameter settings shown in Table V. As listed in the table, we compare the default parameter setting with disabled hysteresis (*set1*) with the cases in which the frequency of OLSR packet generations is two times (*set2*) and four times (*set3*) higher, while the validity times are kept constant.

TABLE V

OLSR PARAMETER CONFIGURATIONS.

OLSR parameters	*set1*	*set2*	*set3*	*default*
HELLO_INTERVAL(s)	2	1	0.5	2
NEIGHB_HOLD_TIME(s)	6	6	6	6
TC_INTERVAL(s)	5	2.5	1.25	5
TOP_HOLD_TIME(s)	15	15	15	15
HNA_INTERVAL(s)	5	2.5	1.25	5
HNA_HOLD_TIME(s)	15	15	15	15
Hysteresis	no	no	no	yes

TABLE VI

UDP AND TCP THROUGHPUTS IN A 3-HOP CHAIN NETWORK FOR DIFFERENT OLSR PARAMETER SETTINGS.

Parameter Setting	UDP	TCP
default	700.4Kbps (60Kbps)	638.1Kbps (149Kbps)
set1	1.306Mbps (87Kbps)	838.5Kbps (79Kbps)
set2	1.605Mbps (76Kbps)	1.020Mbps (105Kbps)
set3	1.84Mbps (106Kbps)	1.306Mbps (56Kbps)

The experimental results obtained by replicating five times the throughout measurements for UDP and TCP traffic are listed in Table VI, in which the average throughput and its standard deviation (in parenthesis) are reported. The shown results indicate that increasing the frequency the OLSR packets are generated by a factor of four, and maintaining the default validity times, it is possible to improve the average throughput of 40% in the UDP case, and of 55% in the TCP case. We have analyzed the routing table logs generated during the trials, and again we have observed that the throughput increases are due to an improvement in route stability. On the other hand, the increase of routing overheads has a negligible impact on the throughput performance.

In summary, our experimental study indicates that the network performance of Internet access in static configurations can be significantly enhanced (in some cases we have more than doubled the measured throughputs) by properly setting the OLSR parameters such as to improve route stability.

B. *Performance Constraints with Mobility*

To test the mobility support in a multi-homed network configuration we considered the network layout illustrated in Figure 7. In our experiments, node MN2 alternates between position P1 and position P2. More precisely, it starts in position P1, where it is in radio visibility of node MN1. After 50 seconds it moves in position P2, where it is in radio visibility of node MN3. The time needed for moving from P1 to P2 is 20 seconds. After other 50 seconds, host MN2 goes back to position P1. This mobility patterns is periodically repeated throughout the test. The *HNA* messages from GW1 and GW2 form default routes to the external network on a short-hop basis. Hence, while connected to MN1, the node MN2 uses GW1 as default gateway. On the contrary, when connected to node MN3, the routes are recalculated and MN2 uses GW2 as default gateway. The new default gateway GW2 will also begin to act as Proxy ARP for the mobile node. The return traffic will be consistently routed through the new gateway as soon as either a new ARP request for the MN2's IP address is issued by the external host, or the gateway GW2 sends a Gratuitous ARP; otherwise it will continue to arrive at the GW1 (see Section V-B.3 for the details).

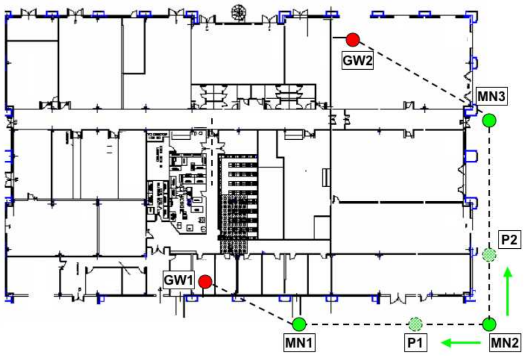

Fig. 7. Trial scenario for testing mobility support.

Figure 8 shows the TCP throughput achieved by MN2 during a mobility test. We compare these results against the throughput measured when node MN2 is fixed in position P1. During both experiments, the hysteresis process was disabled and the other OLSR parameters were set up according to the default values indicated in the RFC specification [7]. The shown results confirm that the TCP session does not break when node moves. The major effect of node mobility is to introduce holes in the TCP traffic due to the time needed to recalculate the new routes to reach the default gateway. In the considered case of ”soft” handoff, i.e., the mobile nodes is in radio visibility of both node MN1 and node MN3 when changing position, we measured up to 20 seconds for recomputing a consistent routing table in node MN2. It is worth noting that in similar experiments conducted in [11], the throughput of mobile node was approximately 30% lower when mobile node changed position. This was expected because the TCP-session continuity was ensured at the cost of using IP tunneling that introduces significant additional overheads. On the contrary our solution is very efficient and lightweight, because it operates directly at the data link layer.

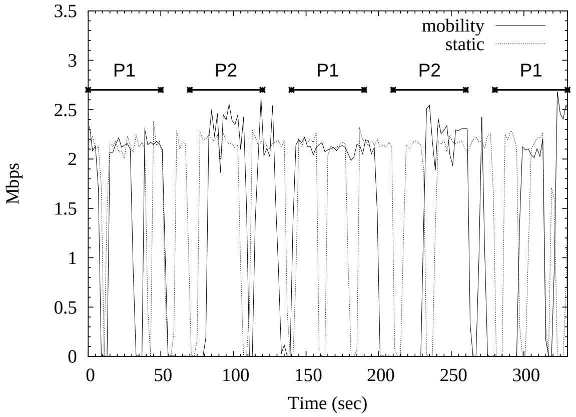

Fig. 8. Throughput of a single TCP flow with node mobility.

VII. Concluding Remarks

In this paper, we have presented a practical architecture to logically extend traditional wired LANs using multi-hop ad hoc networking technologies. Our proposed architecture provides ad hoc node self-configuration and both Intranet and Internet connectivity in a way that is transparent to the wired nodes, i.e., without requiring changes in the pre-existing wired LAN. In addition, by locating our architecture below the IP level, we have designed a lightweight and efficient ad hoc support framework, which is easy to be implemented and introduces minimal overheads.

We have prototyped the proposed architecture to test its functionalities. The shown experimental results indicates that: $i)$ the network performance of Internet access in static configurations can be significantly enhanced (in some cases we have more than doubled the measured throughputs) by properly setting the OLSR parameters such as to improve route stability; and $ii)$ the continuity of TCP sessions during node mobility is achieved without requiring additional overheads.

Several possible extensions of this architecture are currently under investigation. Firstly, different ad hoc routing protocols could be used in separated ad hoc components, either proactive or reactive. The interactions between gateways based on different solutions is not a trivial problem to handle. Moreover, providing a full IP compatibility requires that the ad hoc network should support not only unicast routing but also multicast and other advanced IP functionalities. We are currently investigating how our solution could be extended to provide the complete IP support.

VIII. Acknowledgements

The authors are thankful to the anonymous reviewers for the useful comments in improving the quality of the paper. This work was partially funded by the Italian Ministry for Education and Scientific Research (MIUR) in the framework of the FIRB-VICOM project, and by the Information Society Technologies program of the European Commission under the IST-2001-38113 MobileMAN project.

References

[1] R. Bruno, M. Conti, and E. Gregori, “Mesh Networks: Commodity Multihop Ad Hoc Networks,” *IEEE Communications Magazine*, vol. 43, no. 3, pp. 123–131, March 2005.

[2] P. Srisuresh and M. Holdrege, “IP Network Address Translator (NAT) Terminology and Considerations,” RFC 2663, August 1999. [Online]. Available: http://www.ietf.org/rfc/rfc2663.txt

[3] C. Perkins, “IP Mobility Support for IPv4,” RFC 3344, August 2002. [Online]. Available: http://www.ietf.org/rfc/rfc3344.txt

[4] A. Acharya, A. Misra, and S. Bansal, “A Label-switching Packet Forwarding Architecture for Multi-hop Wireless LANs,” in *Proc. of ACM WoWMoM 2002*, Atlanta, Georgia, USA, September, 28 2002, pp. 33–40.

[5] C. Tschuding, R. Gold, O. Rensfelt, and O. Wibling, “LUNAR: a Lightweight Underlay Network Ad-hoc Routing Protocol and Implementation,” in *Proc. of NEW2AN’04*, St. Petersburg, Russia, February, 2–6 2004.

[6] R. Draves, J. Padhye, and B. Zill, “The architecture of the Link Quality Source Routing Protocol.” Microsoft Research, Tech. Rep. MSR-TR-2004-57, 2004.

[7] T. Clausen and P. Jaquet, “Optimized Link State Routing Protocol (OLSR),” RFC 3626, October 2003. [Online]. Available: http://www.ietf.org/rfc/rfc3626.txt

[8] R. Ogier, F. Templin, and M. Lewis, “Topology Dissemination Based on Reverse-Path Forwarding (TBRPF),” RFC 3684, February 2004. [Online]. Available: http://www.ietf.org/rfc/rfc3684.txt

[9] M. Benzaid, P. Minet, K. Al Agha, C. Adjih, and G. Allard, “Integration of Mobile-IP and OLSR for a Universal Mobility,” *Wireless Networks*, vol. 10, no. 4, pp. 377–388, July 2004.

[10] U. Jönsson, F. Alriksson, T. Larsson, P. Johansson, and G. Maguire Jr., “MIPMANET - Mobile IP for Mobile Ad Hoc Networks,” in *Proc. of MobiHoc 2000*, Boston, MA, USA, August, 11 2000, pp. 75–85.

[11] P. Engelstad, A. Tønnesen, A. Hafslund, and G. Egeland, “Internet Connectivity for Multi-Homed Proactive Ad Hoc Networks,” in *Proc. of IEEE ICC’2004*, vol. 7, Paris, France, June, 20–24 2004, pp. 4050–4056.

[12] P. Engelstad and G. Egeland, “NAT-based Internet Connectivity for On Demand MANETs,” in *Proc. of WONS 2004*, Madonna di Campiglio, Italy, January, 18–23 2004, pp. 4050–4056.

[13] R. Droms, “Dynamic Host Configuration Protocol,” RFC 2131, March 1997. [Online]. Available: http://www.ietf.org/rfc/rfc2131.txt

[14] N. Vaidya, “Weak Duplicate Address Detection in Mobile Ad Hoc Networks,” in *Proc. of ACM MobiHoc 2002*, Lausanne, Switzerland, June, 9–11 2002, pp. 206–216.

[15] S. Nesargi and R. Prakash, “MANETconf: Configuration of Hosts in a Mobile Ad Hoc NEtwork,” in *Proc. of INFOCOM 2002*, vol. 2, New York, NY, June, 23–27 2002, pp. 1059–1068.

[16] K. Weniger and M. Zitterbart, “Address Autoconfiguration on Mobile Ad Hoc Networks: Current Approaches and Future Directions,” *IEEE Network*, vol. 18, no. 4, pp. 6–11, July/August 2004.

[17] S. Carl-Mitchell and J. Quarterman, “Using ARP to Implement Transparent Subnet Gateways,” RFC 1027, October 1987. [Online]. Available: http://www.ietf.org/rfc/rfc1027.txt

[18] D. Plummer, “An Ethernet Address Resolution Protocol,” RFC 826, November 1982. [Online]. Available: http://www.ietf.org/rfc/rfc0826.txt

[19] A. Tønnesen. (2004, December) Implementation of the OLSR specification (OLSR_UniK). Version 0.4.8. University of Oslo. [Online]. Available: http://www.olsr.org/

[20] E. Borgia, “Experimental evaluation of ad hoc routing protocols,” in *Proc. of IEEE PerCom 2005 Workshops*, Kauai Island, Hawaii, March, 8–12 2005.

[21] C. Gomez, D. Garcia, and J. Paradells, “Improving Performance of a Real Ad-hoc Network by Tuning OLSR Parameters,” in *Proc. of IEEE ISCC 2005*, Cartagena, Spain, June, 27–30 2005, pp. 16–21.