The Computer Journal Advance Access published May 11, 2007

doi:10.1093/comjnl/bxm013

A Layer-2 Framework for Interconnecting *Ad Hoc* Networks to Fixed Internet: Test-bed Implementation and Experimental Evaluation

E. ANCILLOTTI[1], R. BRUNO[2,*], M. CONTI[2], E. GREGORI[2] AND A. PINIZZOTTO[2]

[1]*Department of Information Engineering, University of Pisa, Italy*
[2]*IIT–National Research Council (CNR), Via G. Moruzzi 1, 56124 Pisa, Italy*
**Corresponding author: r.bruno@iit.cnr.it*

It is widely recognized that a prerequisite for the commercial penetration of the *ad hoc* networking technologies is the integration with existing wired/wireless infrastructure-based networks to provide an easy and transparent access to the Internet and its services. However, most of the existing solutions for enabling the interconnection between IPv4-based mobile *ad hoc* networks and the Internet are based on complex and inefficient mechanisms, as Mobile-IP and IP tunnelling. In this paper, we describe an alternative approach to build multi-hop and heterogeneous proactive *ad hoc* networks, which can be used as flexible and low-cost extensions of traditional wired local area networks (LANs). Our proposed architecture provides transparent global Internet connectivity and address autoconfiguration capabilities to mobile nodes without requiring configuration changes in the pre-existing wired LAN, and relying on basic layer-2 functionalities. We have prototyped the core components of this architecture for Optimized Link State Routing-based *ad hoc* networks and we have conducted several experiments comparing the throughput performance obtained using our scheme and a well-known alternative (network address translation) NAT-based solution. The experimental outcomes show that our proposed technique ensures higher perconnection throughputs than the NAT-based solution in the considered network scenarios.

Keywords: Ad hoc network, Internet access, OLSR, Experimental evaluation

Received 27 July 2006; revised 7 March 2007

1. INTRODUCTION

A mobile *ad hoc* network (MANET) is a collection of mobile nodes connected together over a wireless medium, which self-organizes into an autonomous multi-hop wireless network. Traditionally, MANETs have been considered as small-scale *standalone* 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 areas, battlefields, temporary networks, etc.). However, the recent advances in mobile and ubiquitous computing, and the development of inexpensive, portable devices are further extending the application fields of *ad hoc* networking. Indeed, it is now widely recognized that supporting an easy access to the Internet and its services is one of the most important features needed to foster the commercial penetration of the *ad hoc* networking technologies. In fact, mobile users are looking for multipurpose networking platforms in which cost is an issue and Internet access is a must. As a consequence, nowadays, multi-hop *ad hoc* networks do not appear as isolated self-configured networks, but rather emerge as a flexible and low-cost extension of wired infrastructure networks, coexisting with them. 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 *ad hoc* network integrated into classical wired/wireless infrastructure-based networks [1].

In this paper we concentrate on investigating how this networking paradigm can be applied to extend the range of

traditional wireless local area networks (WLANs) [2] over multiple radio hops, in order to provide a seamless and untethered mobility support for mobile/portables devices in the local area environment. Ensuring a seamless network coverage of a local environment using traditional infrastructure-based WLANs is a difficult challenge to address because of the several factors that impair the radio transmissions, as electromagnetic interference, fading, obstacles, etc. Integrating *ad hoc* networking in traditional WLAN technologies allows discovering and maintaining multi-hop wireless paths within the network, providing a flexible, robust and cost-effective solution to increase coverage areas. More precisely, we envisage a heterogeneous network in which wired and multi-hop wireless technologies transparently coexist and interoperate (as shown in Fig. 3). In this network, separated groups 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. In addition, the gateways use their wired interfaces also to communicate with static hosts residing in the wired LAN. The network resulting from the integration of the *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.

Three different categories of solutions have been proposed for enabling interconnection between *ad hoc* networks and the Internet. One approach requires the implementation of a network address translation (NAT) on each gateway. In this case, the mobile nodes do not need a globally routable IP address because the NAT gateway translates the source private IP address of the outgoing traffic with a public IP address, which is routable on the fixed Internet. An alternative approach relies on the design of techniques capable of automatically configuring a unique, topology-dependent and globally routable IP address for each mobile node visiting an *ad hoc* network. Finally, a third category of solutions assumes that a mobile IP foreign agent (MIP-FA) is implemented in the *ad hoc* nodes that act as Internet gateways. In this case, the mobile node needs a *permanent* and unique globally routable IP address (i.e. its home address), which is used during the registration procedures with the foreign agents (FAS) of the visiting *ad hoc* network.

This paper describes a simple, yet practical approach to logically extend a wired LAN by employing proactive *ad hoc* networking technologies in a way that is transparent for the wired nodes. The scope of our work is restricted to considering IPv4-based multi-hop *ad hoc* networks and we left to future studies the extension of our solution to IPv6-based MANETs. One of the main innovations of our solution is to rely only on basic address resolution protocol (ARP) capabilities [3] and standard IP routing rules. By positioning our architecture at the data link layer, we may avoid undesired and complex interactions with the IP protocol and provide global Internet connectivity in a very straightforward manner. In addition, we describe a distributed protocol for the address autoconfiguration of *ad hoc* nodes, which relies on dynamic host configuration protocol (DHCP) servers located in the wired part of the network and does not require that new *ad hoc* nodes have direct access to the DHCP servers. Using our scheme, mobile nodes can dynamically obtain a unique IP address that is topologically correct within the extended LAN. We have prototyped the main components of our architecture in a general and realistic test-bed using the Optimized Link State Routing (OLSR) protocol [4] as the *ad hoc* routing protocol. In this test-bed, we have conducted a large variety of experiments, comparing the throughput performance of Internet access provided by our proposed scheme and an alternative well-known NAT-based solution [5]. The experimental results show that in the considered network scenarios: (i) our scheme provides higher per-connection throughputs than the NAT-based solution; (ii) a limited node mobility does not cause permanent transport-layer session breaks; (iii) node mobility may induce drastic throughput degradations when using the NAT-based solution, while for the tested mobility patterns and traffic configurations our technique provides more efficient gateway handoffs; and (iv) the network performances can be significantly improved by properly setting the OLSR protocol parameters such as to increase route stability.

Recently, other schemes have been proposed to provide *ad hoc* support below the IP layer. For example, in [6] label switching was employed to put routing logic inside the wireless network card. More recently, the LUNAR [7] *ad hoc* routing framework and the mesh connectivity layer (MCL) [8] have been proposed. These solutions locate the *ad hoc* support between the data link layer and the network layer. This 'layer 2.5' is based on *virtual* interfaces that allow abstracting the design of *ad hoc* protocols from both the specific hardware components and the 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 significantly increases the system complexities and introduces additional header overheads. On the contrary, our proposed architecture exploits existing ARP capabilities, reducing implementation complexity, providing fully backward compatibility and ensuring minimal overheads.

The rest of this paper is organized as follows. Section 2 discusses the variety of architectural issues and design options that need to be considered to interconnect *ad hoc* networks to fixed IP networks. Section 3 introduces existing approaches to tackle the inter-networking of MANETs with the fixed Internet and reviews the most well-known solutions. The design principles and the protocol details of our proposed interconnected architecture are described in Section 4. Section 5 shows experimental results on the network performance in various test-bed configurations. Finally, Section 6 draws concluding remarks and discusses future work.

2. BASIC DESIGN CHALLENGES

The characteristics of the *ad hoc* networking differ substantially from the conventional IP architecture. Therefore, the interconnection of *ad hoc* networks to the fixed Internet brings up several issues regarding addressing, routing, mobility management and gateway selection. In this section, we discuss, in particular, the substantial conflicts between the routing and addressing architectures of MANETs and the fixed Internet.

One of the fundamental characteristics of the fixed Internet is the adoption of a hierarchical addressing architecture with the IP addresses divided into a network identifier and a host identifier. All the hosts in a certain network segment have to share the same network identifier (also called network prefix). Consequently, the IP address has a location-specific topological meaning and the conventional IP routing protocols can use one single route to address an entire IP subnet (i.e. a network consisting of hosts that share the same network prefix). Note that the use of structured addressing in the Internet has been a fundamental factor in enabling Internet scalability. On the contrary, in the traditional view of *ad hoc* networking, structured addresses are not required and IP addresses have been seen as unique identifiers without a specific topological meaning. Consequently, *ad hoc* networks are usually autonomous systems without hierarchy and organized by adopting a flat private address space. Thus, routing in *ad hoc* networks is typically performed using host-specific routes. It is evident that non-trivial issues arise to allow the interoperability between routing protocols adopting either network-specific routes or host-specific routes.

Another characteristic that distinguishes *ad hoc* routing and conventional IP routing is the use of multi-hop communications within the *ad hoc* network. This implies that hosts residing in the same network do not always share a common physical link. On the contrary, nodes in a fixed Internet sharing the same network prefix expect to have link-layer connectivity. Several basic mechanisms employed by conventional IP-based protocols (such as Mobile IP) heavily rely on this assumption. This means that these mechanisms should be modified and/or extended to be applied in multi-hop *ad hoc* networks.

Another problem that needs to be addressed to interconnect *ad hoc* networks to the fixed Internet is how to determine that an address is not present in the *ad hoc* domain. This is a trivial issue when using a structured addressing architecture, because it is always possible to decide whether a destination is located within the same network by simply looking at the destination's network prefix. Furthermore, in this case default routes can be used when no route exists to that destination. On the other hand, these features cannot be considered as implicitly granted by the *ad hoc* routing protocol. For these reasons, several *ad hoc* routing protocols implement specific route discovery mechanisms that are executed before deciding whether the destination is within the *ad hoc* network or not.

When the *ad hoc* node has somehow determined that the destined node is not present on the MANET, it has to detect the available gateways it can use to reach that destination. Thus, specific gateway discovery procedures and gateway selection algorithms should be designed. In addition, gateway selection is particularly critical when managing mobility inside the *ad hoc* network. In fact, frequent and unnecessary changes of selected gateways can introduce a significant burden to the routing protocol and drastically degrade the performance of transport-layer connections.

An issue that is strictly related to the gateway selection is how to ensure that the incoming traffic returning from the Internet is correctly routed to the gateway and then back to the originator *ad hoc* node. To allow this, the *ad hoc* node needs an IP address that is routable from the rest of the Internet. A variety of solutions can be devised to accomplish this, such as either implicit address translation at the gateway or some kind of explicit signalling between the *ad hoc* node and the gateway. However, at least the gateway must have a globally routable IP address because it resides on the edge between the *ad hoc* domain and the external Internet.

3. EXISTING SOLUTIONS FOR INTERCONNECTING MANETS TO FIXED INTERNET

In the following, we review the existing solutions for enabling the inter-networking of *ad hoc* and fixed networks, pointing out how they cope with the main challenges introduced in Section 2. Our survey is focused on the mechanisms that can be applied to IPv4-based MANETs because our proposed scheme is especially designed for IPv4 networks. For the sake of completeness, at the end of this section we briefly discuss the approaches proposed for IPv6 networks, providing references to the most consolidated proposals.

3.1. Mobile IP-based approaches

One of the possible approaches to provide Internet connectivity for *ad hoc* networks is based on the integration of Mobile IP[1] [9] with the *ad hoc* routing protocols. The reader is referred to the Mobile IP specification [9] for a description of the Mobile IP protocol.

The key idea behind the integration of Mobile IP and *ad hoc* networking is to set up an MIP-FA in the gateways that interconnect the *ad hoc* network with external networks, and to run an extended version of classical mobile IP in the MANET. This approach requires that each mobile node has a permanent and unique routable IP address (i.e. an 'home address'). When a mobile node visits an *ad hoc* network it adopts its home

[1]In this section, we consider only Mobile IPv4 and the IP version is omitted for brevity

address as a unique identifier, and exploits conventional *ad hoc* routing protocols to establish a multi-hop route with one of the FAs present in the MANET. After registering with one of the available MIP-FA gateways, the visiting node will be globally routable on the Internet. Although the basic design principles are simple, several problems arise when adapting the mobile IP to operate in multi-hop *ad hoc* networks. The first obstacle comes from the fact that standard mobile IP protocol assumes that FAs and visiting nodes have link-layer connectivity to rely on link-local broadcast for signalling and control. On the contrary, in multi-hop wireless networks broadcast transmissions consume a great amount of network resources (i.e. bandwidth and energy) because broadcast messages flood the whole *ad hoc* network [10]. In addition, the mobile IP protocol adopts a proactive approach for agent discovery and for executing movement detection and handoff, which is based on periodic broadcasting of agent advertisements. In some situations, this behaviour clashes with the *ad hoc* routing protocol operations, especially when they are carried out in an on-demand manner. Finally, since the addressing architecture in *ad hoc* networks is usually flat, specific techniques have to be designed to allow mobile nodes to decide whether a destination is located within the *ad hoc* network or in the Internet. In the remaining of this section, we review the different solutions that have been proposed to cope with these problems, both for proactive and reactive *ad hoc* networks, pointing out advantages and limitations of each of them.

Early research on MANETs focused on the design of proactive *ad hoc* routing protocols. Consequently, initial solutions for interconnecting *ad hoc* networks to the Internet described mechanisms for using Mobile IP on top of proactive routing. For instance, in [11] an initial design is presented to integrate standard IP routing to DSR [12], a source-based on demand *ad hoc* routing protocol. The basic principle is that each node participating in the same *ad hoc* network selects its home address from a common IP subnet. In this case, the gateway can be configured as a standard IP router between the MANET subnet and the external Internet. Mobile IP is then used to support the migration of mobile nodes from the Internet into and out of *ad hoc* networks. The idea is that mobile nodes piggyback Mobile IP AGENT SOLICITATION ON DSR ROUTE REQUESTS to register with the FA colocated with the gateway. In addition, when the mobile node wants to communicate with a node outside of the *ad hoc* network, the FA sends a *proxy* ROUTE REPLY listing itself as the last hop in the route to the intended destination.

An alternative, and more elaborate, solution to provide Internet connectivity for reactive *ad hoc* networks is described in [13], and it is known as MIPMANET. This scheme assumes that *ad hoc* nodes forward traffic using AODV [14], a table-driven on-demand routing algorithm. In contrast with the approach adopted in [11], the MIPMANET architecture does not require that the mobile nodes' home addresses belong to the same IP subnet. When a mobile node wants Internet access it simply register its arbitrary home address with one of the available FAs, and tunnels each packet to the FA with whom it is registered, which decapsulates the packets and forwards them to the destination. This tunnelling is used to emulate the concept of default routes into the on-demand *ad hoc* routing protocol. Since MIPMANET does not impose any network-prefix semantic within the *ad hoc* network, the sender must initiate a route discovery process using the AODV routing protocol to decide whether a destination is located within the *ad hoc* network or not. If the destination is not found within the *ad hoc* network, then the mobile node infers that the destination is in the wired Internet and tunnels the packets addressed to that destination to the FA. Furthermore, MIPMANET uses reverse tunnelling as defined in [15], establishing an IP tunnel both in the forward direction (from home agent address to FA care-of-address) and in the reverse direction. Figure 1 illustrates the fundamental components of the MIPMANET architecture. Concerning the FA agent discovery mechanism, the MIPMANET scheme adapts the Mobile IP protocol to operate in a more on-demand fashion allowing FAs to periodically unicast AGENT ADVERTISEMENT messages to registered nodes. In addition, MIPMANET utilizes a new algorithm, called MIPMANET Cell Switching (MMCS), to determine when mobile nodes should register with a new FA agent upon moving.

In [16] a solution similar to MIPMANET is presented to implement an MIP-FA on a gateway for *ad hoc* networks running the AODV routing protocol. However, the scheme proposed in [16] adopts a somehow simpler approach, limiting the use of IP tunnels inside the *ad hoc* network. More precisely, MIP-FA gateways periodically advertize their presence through AGENT ADVERTISEMENT messages, and each mobile node maintains a list containing the IP addresses of available FAs. A mobile node that wants Internet access can register with the closest FA one of the care-of-addresses announced

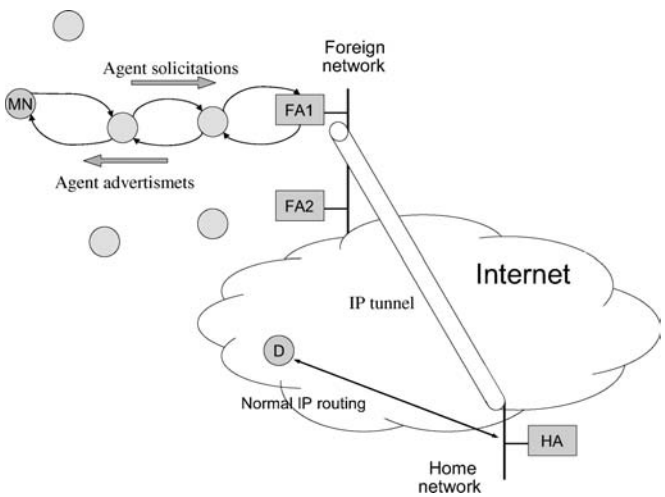

FIGURE 1: illustration of the MIPMANET solution

in the received AGENT ADVERTISEMENT messages, and then it updates the location information maintained by its HA. In addition, mobile nodes can proactively search for FAs by issuing route request (RREQ) packets addressing the *All Mobility Agents* multicast group [9]. When an FA receives an RREQ for a destination to which it does not have an explicit route (note that the FA has explicit routes only for mobile nodes registered with it) it replies with a special route reply (RREP), called FA-RREP, informing the sender that the destination can be reached through the gateway. In other words, the gateway generates *proxy* RREP packets on behalf of the nodes that might be present on the external Internet. Upon receiving an FA-RREP the source node does not use immediately this route, but waits for receiving normal RREP messages indicating that the destination node is located within the *ad hoc* network. Moreover, by properly setting the destination sequence numbers in the RREP message it is also possible to ensure that routes to nodes in the MANET will always have higher priority than routes established using FA-RREPs sent by gateways. The advantage of using FA-RREPs is that data packets can be transmitted to the FA using standard IP forwarding and it is not needed to use tunnelling within the *ad hoc* network.

The solutions described so far integrate reactive *ad hoc* routing protocols (DSR or AODV) with the Internet routing and the Mobile IP architecture. However, the basic design of many of the Mobile IP mechanisms, such as agent discovery, movement detection and reachability of he mobile node, follows proactive principles. Thus, it is opportune to find a trade-off between the on-demand operations of reactive *ad hoc* routing protocols and the overhead introduced by periodically broadcasting agent advertisements. For instance, MIPMANET proposed to use the MMCS algorithm [13] to detect and move to new FAs According to this algorithm, a mobile node should register with another FA if it is at least two hops closer to this new FA than to its previous FA, for at least two consecutive agent advertisements. In [17], a protocol that limits the flooding of agent advertisements in an n-hop neighbourhood using TTL scoping is described. In [18], Mobile IP is extended to manage multiple simultaneous connections with foreign networks. Based on the registered care-of-addresses, multiple paths can be used for packets to and from a mobile node. Thus, enhanced throughput and a more reliable connection can be achieved. An alternative scheme is proposed in [19]. In that work, some heuristics are described to classify the traffic load of the gateways, such as to avoid selecting congested route to gateways. In addition, to reduce the flooding overhead due to solicitations, optimized searching algorithms are used such as the Expanding Ring Search method [14]. However, some intrinsic limitations of on-demand routing protocols as the inability to support default routes and to easily determine that an address is not present on the MANET, cannot be overcome without relying on complex address allocation schemes or resource-consuming IP-in-IP encapsulation [20].

For these reasons, recently, the integration of Mobile IP with proactive *ad hoc* routing protocols has gathered a lot of attention in the research community. A hierarchical approach to accomplish this integration is proposed in [21]. More precisely, Mobile IP is used to support macro-mobility between different IP domains, while the OLSR *ad hoc* routing protocol is adopted to support micro-mobility inside the MANET environment. As in prior work, the architecture proposed in [21] contains a gateway implementing an MIPFA, allowing the OLSR-IP access network to be connected to the Internet. In addition, in this hierarchical architecture, special nodes, called OLSR *Base Stations*, have been introduced to reduce the number of global location updates performed by Mobile IP. These base stations have both wired and wireless interfaces and implement the OLSR protocol on both of them. When a mobile node enters an IPOLSR access network, it receives either periodic AGENT ADVERTISEMENT messages broadcasted by the gateways or unicast AGENT ADVERTISEMENT messages sent by the gateways in reply to the node's AGENT SOLICITATION. Once the mobile node is registered, it is attached to the OLSR-IP access network. This implies that the node has a host-specific entry in the routing table for each IP destination address known locally on the MANET, while all the traffic for external networks is forwarded along a possible default route out of the MANET through the gateway. The HNA messages (see Section 4.2.2 for a description of the OLSR protocol) issued by the gateway establish this binding between the gateway itself and the external networks. In such a way, it is not required to have an explicit procedure to determine if a destination is present or not in the MANET. Note also that the use of IP tunnelling is limited to the communications between FA and HA, but it does not occur within the *ad hoc* network.

Albeit, the considerable effort devoted to the design of solutions enabling an efficient integration of Mobile IP with the *ad hoc* routing protocols to provide a global mobility between Internet and MANETs, Mobile IP-based solutions have still a number of drawbacks. The first one is that in order to allow Mobile IP and *ad hoc* networking to cooperate it is necessary to introduce further complexities and sub-optimal operations in the implementation of both Mobile IP and *ad hoc* routing protocols. Probably, an integrated design of Mobile IP and *ad hoc* routing functionalities might be much more efficient, minimizing the overheads. In addition, Mobile IP was designed to handle mobility of devices in case of relatively infrequent mobility. Thus the overheads introduced to manage roaming and handoffs between FAs are a relevant issue in MANET environments, where handoff functions operating at the data link layer may be more suitable. Finally, when the technique of default routes is used to route the traffic from the mobile node to the closest gateway, the use of Mobile IP can easily lead to triangle routing. In fact, when the mobile node moves it can get closer to a gateway different from the one to which it is

currently registered. As a consequence, the forward traffic leaves the *ad hoc* network through one gateway while the return traffic enters the *ad hoc* network through the new MIP-FA gateway to which the mobile node is registered. To solve this problem the authors of [1] have proposed to use Mobile IP reverse tunnelling [13], or explicit tunnelling to one of the gateway instead of using default routes. Note that changing between two MIP-FA-based gateways do not cause a transport-layer session break because the mobile node has simply to register the new FA with its HA. As we will discuss in the following section, changing gateways in NAT-based solutions is much more critical.

3.2. NAT-based approaches

An alternative category of solutions to interconnect MANETs to the Internet is based on the implementation of NAT modules [22] on the gateways. Traditionally, basic NAT is used to allow hosts in a private network to transparently access the external Internet. More precisely, a router with a NAT module implemented on it, behaves as an address translator, which dynamically maps the set of non-routable private addresses (selected from a reserved address range) used internally in the local domain to a set of globally routable network addresses. A variant of basic NAT is the Network Address Port Translation, or NAPT [22], which translates private IP addresses to a single globally routable IP address using different transport layer ports. When implementing a NAT module in a gateway of the *ad hoc* network, this gateway translates the source IP address of outgoing packets with its IP address, 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* source node. Note that NAT-based approaches impose a limited addressing structure within the *ad hoc* network. In fact, the *ad hoc* network is identified by one or more network-prefixes designated to be used in private networks, while the host-specific part of the address is autonomously administered within the MANET.

One of the earliest implementation of this method for reactive *ad hoc* networks was developed by the Uppsala University for the AODV routing protocol [23]. In that implementation mobile nodes are not aware of the available gateways enhanced with NAT capabilities, and these gateways use proxy RREP messages to reply to RREPs destined for hosts on the Internet. The use of proxy RREP messages is clearly inspired by work done in [11, 16]. To reduce the complexity of the implementation, the AODVUU NAT solution assumes that all the addresses on the *ad hoc* network are allocated from the same private IP subnet. Thus, the gateways reply only to RREQ messages targeting destinations external to the *ad hoc* network. To avoid this limitation, it is possible to apply solutions similar to the ones defined in [16] for an MIP-FA-based gateway. More precisely, the NAT gateway can be allowed to reply to all the received RREQ messages generating a proxy RREP with the same sequence number of the received RREQ. Hence, a direct route to the designated destination not traversing the gateway will always have preference to the route announced by proxy RREP messages.

Solutions based on the use of proxy RREPs do not work correctly in multi-homed networks, i.e. when multiple gateways are present in the *ad hoc* network. Indeed, to avoid transport-layer session breaks it is necessary to ensure that each packet from the same session is routed over a specific gateway, since a NAT router translates both outgoing and incoming packets. However, it is difficult to control the gateway selection for an *ad hoc* node that is moving. To solve this problem, in [24] an alternative approach is described to provide Internet connectivity for on-demand routing protocols. First of all, the solution proposed in [24] defines a method for implementing gateway discovery using the AODV protocol. When a mobile node searches a route to its designated destination, it initially floods the network with normal RREQ messages. If the source node does not receive any reply, it issues a special RREQ message with a 'Gateway'-flag set. Upon receiving this type of RREQ messages the gateways are allowed to reply with proxy RREPs on behalf of external nodes. When the source node receives these replies, it becomes aware of the available gateways. Then, the *ad hoc* node selects one of these gateways according to some heuristic and it tunnels the packets addressed to external destinations to the selected gateway using, for example, IP-in-IP encapsulation [20]. Furthermore, the NAT gateway will also tunnel the incoming packets returning from the Internet to the source nodes. Figure 2 illustrates how the IP-in-IP encapsulation method works when tunnelling the packets for an external host via a NAT gateway.

As described in [5], NAT gateways can also be implemented in proactive *ad hoc* networks. Using proactive

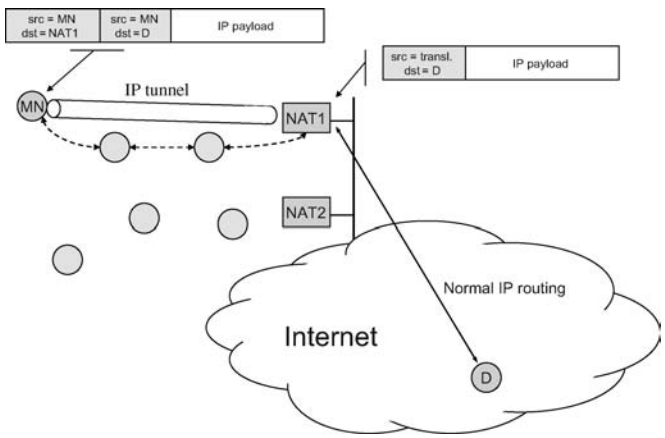

FIGURE 2: illustration of the tunnelling operations in NAT-based solutions [5, 24]

routing protocols, it is easy to advertize gateways within the *ad hoc* network, and default routes can be established between the mobile nodes and the closest gateway. However, instead of using default routes to send packets addressed to host located in the external Internet as proposed in [21], the solution proposed in [5] requires that each sender node establishes a tunnel with the selected gateway (e.g. using IP-in-IP encapsulation [20] or minimal IP encapsulation [25]) to deliver packets addressing external IP networks. Employing explicit tunnelling instead of default routes ensures that each packet of the same transport-layer session is consistently routed through the same gateway, even if the source node moves. In [5, 24], the problem of how enabling gateways adopting different mechanisms to cooperate while providing Internet connectivity is also addressed. For instance, in multi-homing scenario some gateways may be NAT-based, and other gateways may be MIP-FA-based. Again, a working solution for this issue is the use of explicit tunnelling that forces the packets generated from the same session to be routed through the gateway selected at the beginning of the session itself. This prevents the session break when the source node gets closer to another gateway. However, this method requires that special techniques be implemented to allow source nodes to discover the different capabilities of the available gateways. To this end, extensions of the *ad hoc* routing protocols have to be devised. Finally, it is worth pointing out that most of the schemes described in this section rely heavily on the use of IP tunnels. However, tunnelling introduces a fixed overhead in the tunnelled IP packet headers. For instance, in case of IP-in-IP encapsulation, 20 bytes of overhead are added in every outgoing packet. Experimental results presented in [5] indicate that in some situations the throughput obtained by a TCP session can suffer a 30% decrease. This clearly motivates the need of designing more efficient and lightweight solutions to provide Internet connectivity for *ad hoc* networks.

Although the NAT-based approach undoubtedly has advantages such as to allow the *ad hoc* network to continue to use a private address internally, as well as its simplicity, it also raises several concerns from the application layer standpoint. In particular, NAT is not very suitable for incoming connections and this fact causes significant difficulties for peer-to-peer applications. Recently, a few workarounds have been proposed to reduce the problems created by NAT, such as NAT-traversal techniques. However, a clear solution interoperable with existing NAT devices is still not available.

3.3. Proposals for IPv6-based MANETs

Most of the research aimed at combining IPv6 and *ad hoc* networking has focused on the design of mechanisms to configure globally routable IPv6 addresses within a MANET. In general, these approaches treat the *ad hoc* network as an IPv6 subnet. According to this view, gateway nodes present in the *ad hoc* network act as default routers receiving all the traffic destined for IP subnets outside the MANET. Consequently, in this category of solutions, gateway discovery mechanisms and address autoconfiguration techniques are strictly inter-dependent.

The standard IPv6 architecture includes a stateless address autoconfiguration technique [26], but it cannot be applied in multi-hop topologies. There have been numerous proposals to extend this autoconfiguration protocol to operate in a MANET considering multi-hop operations, network partitioning and merging (e.g. [27]). However, the proposals that have received more attention in the research community are the schemes that deal both with address autoconfiguration and gateway discovery. One solution is described in [28], which defines both proactive and reactive strategies to discover the gateways within the *ad hoc* network. The network prefix, which is distributed by these Internet gateways, can then be used for configuring a (typically globally) routable IPv6 address for each *ad hoc* node. In [28] it is also described how Mobile IPv6 can be modified to be used in *ad hoc* network. In particular, Internet gateway advertisements will be used to detect the node's movement instead of the neighbouring discovery mechanisms used in conventional Mobile IPv6. In addition, the mobile node uses the globally routable address acquired from the Internet gateway as its care-of-address. Hence, no FAs are needed in the *ad hoc* subnet. Finally, packets sent outside the MANET contain a routing header that includes the address of the default gateway. This is a sort of emulation of the tunnel towards the gateway used in the MIPMANET protocol to avoid address reconfigurations after gateway hand-offs. An alternative solution to interconnect IPv6-based MANETs to the fixed Internet is described in [29]. This scheme introduces the concept of 'prefix continuity'. More precisely, in the same MANET, multiple subnets (i.e. network prefixes) can be used, but the network identifiers are assigned to visiting nodes such that any node that selected a given prefix has at least one neighbour with the same prefix on its path to the selected gateway. This implies that the MANET is organized in clusters of hosts sharing the same network prefix. This network organization reduces the overheads introduced by flooding gateway advertisements.

For a comprehensive survey of the several approaches proposed for enabling Internet connectivity for IPv6-based MANETs, and a detailed description of the protocol operations, the interested reader is referred to [30] and references herein.

4. PROPOSED ARCHITECTURE

The basic design principle we adopted during the definition of our proposal was to provide transparent communications between static nodes, which uses traditional wired technologies; and mobile nodes, which uses more advanced *ad hoc* networking technologies, employing mechanisms that run

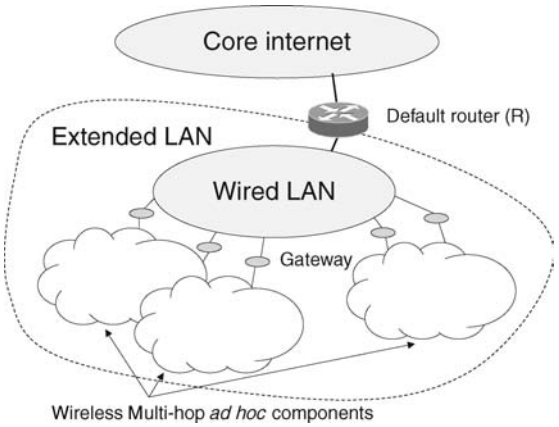

FIGURE 3: reference network architecture

below the IP layer[2]. As discussed in the introduction, in this work we address two major problems: address autoconfiguration and global Internet connectivity for IPv4-based *ad hoc* networks. However, before describing the details of our solutions, it is useful to illustrate the complete network architecture we propose for interconnecting heterogeneous *ad hoc* networks to the Internet. To this end, Fig. 3 depicts the reference network architecture we consider in this work.

As illustrated in the figure, we envision an extended LAN composed of a conventional LAN (the wired component) and several *ad hoc* clouds. In this network, mobile/portable nodes not in close proximity to the fixed networking infrastructure establish multi-hop wireless links to communicate with each other (e.g. using the IEEE 802.11 technology [2]) using an *ad hoc* routing protocol. The wired LAN and the *ad hoc* components are interconnected using gateways, which are special nodes provided with both wired and wireless interfaces. We also assume that the *ad hoc* routing protocol is running on both the gateways' interfaces. This implies that separate (i.e. not in direct radio visibility) *ad hoc* clouds are not disconnected because an *ad hoc* node can exchange routing messages with any other *ad hoc* node in the extended LAN also through the wired LAN. As explained in later sections, this architectural design allows transparent support for node mobility and facilitates the Intranet communications.

In our architecture, multi-homing is permitted, i.e. multiple gateways can be located within the same *ad hoc* component. Consequently, specific mechanisms are required to support the hand-off between gateways without transport-layer connection breaks. In general, between pairs of gateways in radio visibility of each other, two direct links can be established, both wired and wireless. The choice between the two links is demanded by the routing protocol. However, we can assume that the wired link has a lower link cost than the wireless link because wired technologies are still more reliable and have higher capacity than standard wireless technologies. As a consequence, a routing protocol selecting least-cost paths will always give preference to the wired path for inter-gateway communications. In Section 4.2, we will explain the implications of this assumption.

Concerning the addressing architecture of our network, we assume that the extended LAN is a single address space Namely, all the nodes in the extended LAN, both *ad hoc* nodes and static ones, have an IP address with the same network identifier. To indicate the network identifier we use the standard notation IP_S/L, in which IP_S indicates the network prefix, and L the network mask length. For instance, in our test-bed, we used $IP_S/L = X.Y.96.0/22$. It is worth pointing out that our solution does not impose any addressing hierarchy within the extended LAN, and both *ad hoc* and wired nodes may have an arbitrary IP address belonging to the IP_S/L subnet. This implies that the wired nodes are not aware of the *ad hoc* hosts and vice versa.

Standard IP routing is used to connect the extended LAN to the core Internet. Regarding the *ad hoc* routing protocol, our scheme is specifically designed for being integrated with proactive routing protocols. Examples of these type or routing protocols for MANETs are the OLSR protocol [4] or the Topology Dissemination Based on Reverse-Path Forwarding (TBRF) routing protocol [32]. The motivation behind this design choice is that proactive routing protocols usually support gateway advertisements, allowing the gateways to use special routing messages to set up specific default routes in the *ad hoc* network. In addition, proactive routing protocols, adopting classical link state approaches, build 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.

To build and make this extended LAN, operational we have designed three main mechanisms:

- An address autoconfiguration protocol for the *ad hoc* nodes that takes advantage of the DHCP servers located in the wired LAN, which does not require that mobile nodes are aware of the identity/location of these DHCP servers;
- An adaptive proxy-ARP capability, which allows the gateways to intercept packets generated by the wired nodes and addressing the *ad hoc* nodes;
- An automatic procedure to set the proper network specific routing entries needed by the *ad hoc* nodes to correctly forward their traffic to the external networks.

The following sections describe the operations performed by the aforementioned techniques and how the *ad hoc*

[2]An initial version of our proposal, as well as preliminary experimental results, is presented in [31].

components are transparently integrated into the wired infrastructure.

4.1. Address autoconfiguration

A prerequisite for proper routing is that all nodes are configured with a unique address. Thus, various protocols have been proposed in the literature for the purpose of address autoconfiguration in MANETs. Generally speaking, 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 on *stateful approaches* maintain either centralized or distributed address allocation tables, and they execute distributed algorithms to establish a consensus among all the nodes in the network on the new IP address, before assigning it. Protocols proposed in [33] and [34] are well-known examples of the former and latter approach, respectively. A limitation of most of the early solutions is that they are designed to work in *stand-alone* MANET. To address this problem, the Internet Engineering Task force (IETF) has established the *AUTOCONF* working group with the main purpose of standardizing mechanisms to be used by *ad hoc* nodes for configuring unique local and/or globally routable IPv6 addresses[3] In fact, it is now evident that using a unified strategy to select a global node address, to route the packets and to access the Internet may be beneficial, because complexities and overheads are reduced.

In this work, we propose a simple but effective protocol to assign a globally routable IPv4 address to the mobile nodes of the extended LAN by taking advantage of the easy access to the fixed infrastructure. In general, in a wired LAN unconfigured hosts may use the Dynamic Host Configuration Protocol [36] to query centralized servers to obtain IP configuration parameters (i.e. a unique IP address, a common netmask and, eventually, a default gateway). However, this solution is not straightforwardly applicable to *ad hoc* networks because the node running the DHCP server may not be permanently reachable by all nodes. In addition the legacy DHCP server discovery procedure requires link-level connectivity between unconfigured nodes and the DHCP servers. To overcome these limitations, our scheme assumes that DHCP servers are located only in the wired LAN and employs a novel mechanism to allow unconfigured nodes to contact the DCHP servers through multi-hop paths. To achieve our goal we partially reuse some mechanisms from the MANETconf protocol described in [34]. In particular, in MANETconf a node requesting an address first searches for already configured nodes and selects one of its neighbours as initiator of the configuration process. However, MANETconf can be used only to allocate unique and private addresses to nodes in a stand-alone MANET. In our scheme, a mobile node not yet associated with the extended LAN executes a preliminary message handshake to discover reachable and already configured *ad hoc* nodes. Then, the unconfigured node that wants to join the *ad hoc* component elects one of the discovered neighbours as DHCP relay agent. A DHCP relay is an entity that is capable of relaying DHCP_DISCOVER broadcasts from a LAN which does not include a DHCP server to a network which does have one [36]. The proposed address autoconfiguration protocol is basically an extension of the functionalities of the legacy DHCP relay agents. More precisely, as illustrated in Fig. 4, an unconfigured node *N* broadcasts special messages, called NEIGHBOUR_REQ, to discover other nodes that are within its radio visibility and which can interconnect him to the *ad hoc* component. To make the protocol more robust against message losses and network dynamics, node *N* can periodically generate new NEIGHBOUR_REQ messages scanning each channel and operating mode (e.g. 802.1 *a*/*b*/*g*) supported by its interface. When a node that is already part of the *ad hoc* network correctly receives a NEIGHBOUR_REQ message, it discovers the physical address of the node *N* and it can unicast a NEIGHBOUR_REPLY message to node *N*. From the received NEIGHBOUR_REPLY messages, node *N* can build a list of available DHCP relay agents, and it will select one of them according to some heuristic (e.g. the one with the best signal quality, or the last one to reply). In the example shown in Fig. 4, node *N* has selected node *H*2 as its proxy DHCP relay agent. Node *H*2 is informed about this choice through a CONF_REQ message. Node *H*2 acknowledges the correct reception of this request sending a CONF_ACK message to node *N*. After that, node *H*2 activates its *proxy* DHCP relay agent, which initiates the process of address assignment on behalf of node *N* using the standard DHCP protocol. Note that node *H*2 can contact the DHCP servers located in the wired LAN using unicast DHCP

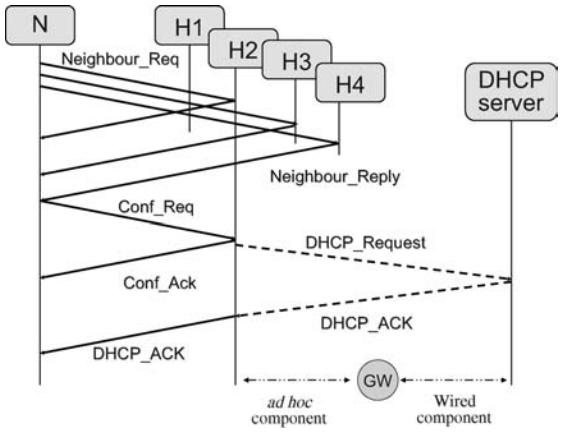

FIGURE 4: message exchanges during the address autoconfiguration

[3]For a general overview of the main approaches for address autoconfiguration in MANET, and the description of the most well-know protocols the reader is referred to [35]. In addition, draft specifications from the Autoconf WG can be found at http://tools.ietforg/wg/autoconf/

control messages, because node *H*2 is already part of the *ad hoc* component. The DCHP messages generated by node *H*2 are routed through one of the gateways according to the mechanisms that will be described in the following sections. The DHCP server receiving the request, will answer to the DHCP relay agent with a DHCP_ACK, containing the IP configuration parameters for the new node *N*. The configuration process is concluded when the DHCP relay forwards the DHCP_ACK message to the initial node *N*, which can now join the network. After joining the *ad hoc* component, node *N* may also turn itself into a DHCP relay agent for the DHCP server from which it received the IP configuration parameters, in order to support the configuration process of new mobile nodes. Finally, it is worth noting that our scheme does not need any initialization procedure because the gateways are directly connected to the wired LAN and can broadcast a DHCP_DISCOVER message to locate available servers. Hence, the first mobile node to enter the *ad hoc* network may find at least one gateway capable of initiating the illustrated configuration process.

A limitation of the proposed autoconfiguration protocol is that the wired LAN should be permanently reachable by the *ad hoc* nodes in order to permit renewing the address lease with the DHCP servers. However, in the considered network scenario this limitation may not be considered problematic because it is reasonable to assume that the network is not highly dynamic. In fact, we envision the extended LAN will be used mostly as a flexible and cost-effective extension of the fixed networking infrastructure in enterprise buildings or campus facilities. In these contexts, users are semi-static or nomadic and are interested in having a continuous access to Internet and its centralized services (e.g. web browsing, access to centralized data repository, etc.).

4.2. Interconnecting *ad hoc* nodes to the fixed Internet

The basic assumptions of our architecture are the following: (i) a proactive *ad hoc* routing protocol is used in the *ad hoc* components. This implies that *ad hoc* nodes' routing tables are populated with entries specifying the next-hop neighbour to forward a message to any other *ad hoc* node in the extended LAN; and (ii) all the hosts in the extended LAN share the same IPv4 network prefix. As previously introduced, the key mechanisms we propose to interconnect the *ad hoc* components (Fig. 3) to the wired LAN and the external Internet rely on the use of network-specific routes and some extended functionalities of the ARP protocol. For clarity, in the following we separately describe how communications are established and maintained between *ad hoc* nodes and hosts in the wired LAN (i.e. Intranet communications) or hosts in the external Internet (i.e. communications routed through the default router *R* shown in Fig. 3). However, before describing the proposed mechanism it is useful to give a short description of the ARP and OLSR protocols, whose functionalities will be extensively exploited in our proposal.

4.2.1. Overview of the ARP protocol

IP-based applications identify 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. the MAC address. The ARP protocol [3] is 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 the hosts attached to the same physical network. This packet contains the destination's 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 APR_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. In particular, in [37] it has been proposed that *the proxy ARP* mechanism makes it possible to construct 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. Hence, when host A sends IP packets for host B, they arrive to the router, which will forward such packets to host B.

4.2.2. Overview of the OLSR routing protocol

The OLSR protocol [4] is an optimization of the classical link-state algorithm tailored to mobile *ad hoc* networks. More precisely, being a proactive routing protocol, OLSR 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. More precisely, each node independently selects its MPR among its one-hop neighbours such as to ensure that all its two-hop neighbours receive the broadcast messages retransmitted by these selected relays. The link state reports, which are generated periodically by MPRs, are called TOPOLOGY CONTROL (TC) messages. These TC messages are flooded to all the nodes in the network, but only the MPRs are allowed to forward the control messages received from other nodes, in order to reduce the number of retransmissions needed to cover the entire network.

OLSR employs a neighbour discovery procedure based on HELLO messages. The HELLO packets contain the list of

neighbours known to the node and their link status. Thus, HELLO messages allow each node to discover its one-hop and two-hop neighbours, which are required during the MPR selection procedure. The neighbourhood information and the topology information are updated periodically, and they enable each node to locally compute the least-cost routes to any possible destination in the *ad hoc* network, using the Dijkstra's shortest path algorithm. This routing table is recomputed whenever there is a change in either the neighbourhood information or the topology information.

In order to enable 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. Consequently, 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 conventional default route 0.0.0.0/0, will receive each packet destined to IP addresses without a known route on the local *ad hoc* network.

4.2.3. Intranet connectivity

As discussed in Section 2, the main problem to solve and support the Intranet connectivity is to guarantee that each *ad hoc* node can identify whether the destination node is within the *ad hoc* part or the wired part of the network, and vice versa. To explain how our scheme addresses this issue let us initially analyse the case of an *ad hoc* node *N* that is to required communicate with a wired host *H*. To perform its routing decisions, normally node *N* has three types of routing information in its routing table. First, the node *N*'s routing table contains host-specific entries to reach any other *ad hoc* node, which are inserted by the proactive *ad hoc* routing protocol. Second, there is an entry that is automatically inserted by the TCP/IP protocol stack at the network boot, which provides the local reachability on its wireless interface. Namely, since the node *N*'s IP address belongs to the IP subnet identified by the IPS/*L* network/mask pair, then *N*'s routing table contains a generic entry that indicates that all the IP addresses matching this network prefix may be directly reached through the wireless interface if a more precise routing entry is not known. Finally, node *N*'s routing table contains the generic routing entry 0.0.0.0/0, which is the default route advertised by the gateways. When the *ad hoc* node *N* wants to send a packet addressed to a wired node *H*, node *N* checks its routing table. Since node *N* does not have a host-specific routing entry for node *H*'s IP address, it believes that node *H* is directly reachable on its wireless interface, and it will issue an ARP_REQUEST message targeting node *H*'s physical address. This ARP_REQUEST fails because node *H* is not directly reachable on node *N*'s wireless interface. To solve this inconsistency, node *N* needs a specific mechanism to discover that node *H* can be reached only through a gateway, although it shares with node *N* the same network prefix. To this end, we exploit the properties of the longest-matching rules used by the standard IP routing. Specifically, we configure the gateways such as to advertise two additional and more precise network specific routes, which announce the reachability of the wired LAN through the gateways' wired interfaces. These network-specific routes are addressing the $\{IP_{S_{Low}}\}/(L+1)$ and $\{IP_{S_{High}}\}/(L+1)$ subnetworks, i.e. the two disjoint IP subnets whose union is equal to IP_S/L[4]. The use of these additional network-specific routes, which are more precise than the route providing local reachability to IPS/*L*, ensures that each *ad hoc* node will forward the traffic to its closest gateway to communicate with any host on the local wired LAN.

The use of the two network-specific routes previously specified is a simple but effective way to guarantee correct routing of egress traffic from the *ad hoc* components. However, they may cause an inconsistent behaviour in the gateways. In fact, each gateway can directly communicate with a wired host *H* through its wired interface. On the other hand, each gateway also receives HNA messages sent by other gateways, setting up the additional routing entries advertized in these messages. Hence, 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 the routing entries targeting the $\{IP_{S_{Low}}\}/(L+1)$ and $\{IP_{S_{High}}\}/(L+1)$ subnets, instead of the entry indicating the local reachability of the host *H* on the gateway's wired interface. The effect is that the IP packet will loop among the gateways until the TTL expires, without reaching the correct destination *H*. To resolve this problem we again exploit the properties of the longest-matching rules used by the standard IP routing. Specifically, each gateway automatically configures two additional routing entries bound to its wired interface. These two additional entries have the same network/mask as the ones announced in the HNA messages (i.e. $\{IP_{S_{Low}}\}/(L+1)$ and $\{IP_{S_{High}}\}/(L+1)$), but with lower metric. Hence, when a gateway wants to communicate with a host in the wired LAN, it will always give preference to its wired interface.

Let us now consider the reverse direction, i.e. a wired host *H* that wants to communicate with an *ad hoc* node *N*. Since node *H* is running standard IP routing, normally its routing table will have only two types of routing entries. First, there is an entry that is automatically inserted by the TCP/IP protocol stack at the network boot, which provides the local reachability on its wired interface. Second, there is a generic entry related to the default router (node *R* in Fig. 3) and used to forward the packets addressing external IP subnets.

[4]To clarify this concept, let us assume that IPs/L = *X.Y*.96.0/22. Then, this network prefix can be split into the two smaller subnets $\{IP_{S_{Low}}\}/(L+1)$ = *X.Y*.98.0/23 $\{IP_{S_{High}}\}/(L+1)$ = *X.Y*.98.0/23. Note that this procedure is always possible for L < 32.

Hence, node *H* has no sufficient routing information to decide whether the destination is in the *ad hoc* or wired part of the network. As stated previously our objective is to design a solution that does not require modifications of routing behaviours in the preexisting wired part of the network. To achieve this goal, we implemented in each gateway an enhanced proxy ARP server that masquerades the IP addresses of all the *ad hoc* nodes reachable through the gateway's wireless interface. When node *H* wants to communicate with node *N*, it issues a conventional ARP_REQUEST message targeting node *N*'s physical address because host *H* believes that node *N* is locally reachable on its wired interface. When a gateway receives this ARP_REQUEST searching for an IP address that is reachable through its wireless interface, the gateway publishes its MAC address. As a consequence, that gateway intercepts all the packets sent by node *H* to node *N*. The intercepted traffic is then forwarded to the designated destination inside the *ad hoc* network using the *ad hoc* routing protocol.

The proxy ARP mechanism we have designed differs from the standard proxy ARP functionality defined in the RFC 1027 [37], because it is capable of adapting to the topology changes. More specifically, each proxy ARP server will proactively check the gateway's routing table to publish on the gateway's wired interface only the IP addresses related to host-specific routing entries bounded to the gateway's wireless interface (this condition ensures that the gateway is the closest one to the *ad hoc* node). When there is a change in the routing table due to node mobility, the proxy ARP reactively updates the list of IP addresses published on the gateway's wired interface. This ensures a transparent hand-off between different gateways, as explained in detail in Section 4.2.5. Finally, it is worth pointing out that there are no inter-dependencies between the OLSR protocol and the proxy ARP servers running on the gateways. In fact, the OLSR protocol maintains the gateway's routing table, which is independently read by the local proxy ARP server to build its list of masqueraded IP addresses. Legacy operations of both OLSR and ARP protocols are not affected by the proxy ARP server actions.

The last aspect that needs to be discussed is the existence of some network configurations where asymmetric routing may occur, i.e. the forward path is different from the return path. Let us consider the case in which node *N* discovers two gateways, say *GW*1 and *GW*2, which are at the same distance (in terms of hops) from node *N*. 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 the wired node *H* for the *ad hoc* 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 *GW*1 is the default gateway for node *N*, but *GW*2 has sent the last ARP reply to node *H*. In this case, node *H* sends the traffic destined to node *N* to *GW*2, which routes it to node *N*. On the other hand, node *N* sends packets destined to node *H* to *GW*1, which forwards them to node *H*. It is important to note that asymmetric paths are not necessarily by themselves a problem. Indeed, both node *N* and node *H* correctly receive and send their packets. In addition, the asymmetric routing occurs only in symmetric topologies. Thus, it is reasonable to assume that in this local environment both paths are characterized by similar delays.

4.2.4. Internet connectivity

Providing Internet connectivity for the *ad hoc* nodes is now intuitive since Internet connectivity can be considered as a special case of the Intranet connectivity explained above. The only additional requirement is that the gateways know the default router's IP address. However, the default router for the LAN is one of the IP configuration parameters that are provided in the DHCP_ACK messages used to configure both wired hosts and *ad hoc* nodes. Thus, when an *ad hoc* node wants to send a packet addressing external IP subnets, it will simply forward the packet to the gateway. Then, the gateway will deliver the packet to the default router using the same mechanisms adopted to communicate with any wired host in the LAN. Similarly, the incoming traffic received from the Internet and targeting the IP address of an *ad hoc* node, will be forwarded by the default router to the gateway that operates the proxy ARP for that IP address. Note that assuming a single public address space for the extended LAN permits a global reachability of the *ad hoc* nodes from nodes located in the external Internet, avoiding the difficulties typical of NAT-based solutions.

4.2.5. Support for gateway hand-offs

In general, solutions to support Internet connectivity for *ad hoc* networks using default routes may experience transport-layer session breaks when the default routes change, depending on the network dynamics. To avoid transport-layer session breaks, in [24] it was proposed to replace default routes with explicit tunnelling between the mobile nodes and the gateways. However, this significantly complicates the implementation and introduces relevant overheads as shown in our experiments (see Section 5). On the contrary, in our architecture the mobility is supported in a transparent way for the higher protocol layers. In fact, when an *ad hoc* node moves and gets closer to a different gateway, there is only a change in the ARP caches of the session endpoint located in the wired LAN, which does not cause the break of active transport sessions. To better clarify the actions occurring during an hand-off, let us consider an *ad hoc* node *N* with a TCP connection open with a wired node *H*, and using gateway *GW*1 as a default route to reach the wired LAN. This implies that the node *H*'s ARP cache contains a mapping between the node *N*'s IP address and the gateway *GW*1's MAC address. When node *N* moves and gets closer to a different gateway *GW*2 it updates its routing table and uses *GW*2 as a new default gateway to reach the

wired LAN. Simultaneously, *GW*2's routing table also changes because the next hop for node *N* switches from *GW*2's wired interface to *GW*2's wireless interface. As a consequence, the proxy ARP running on *GW*2 inserts node *N*'s IP address in the list of IP addresses the gateway *GW*2 publishes. In addition, it generates a GRATUITOUS ARP on the *GW*2's wired interface for node *N*'s IP address. This GRATUITOUS ARP updates the ARP tables in all the wired hosts that have a stale ARP entry for the node *N*'s IP address, which was mapped with the MAC address of *GW*1's wired interface. After this update the traffic destined to and/or originated from node *N* is correctly routed only through gateway *GW*2.

5. IMPLEMENTATION AND EXPERIMENTAL RESULTS

To validate the correctness of the proposed mechanisms and to evaluate the system performance, we have deployed a small-scale test-bed with two gateways and five nodes operating in static or quasi-static configurations. In this test-bed we have prototyped the core functionalities of our architecture. In particular, we have developed the software components described in Section 4.2 to support the inter-networking with the fixed Internet, while we have left to work out in future the implementation and testing of the address autoconfiguration scheme described in Section 4.1. In our test-bed we have also implemented the well-consolidated solution described in [5] to integrate NAT gateways with MANETs running OLSR routing protocol. Comparative tests have been conducted both in static and low-mobility configurations.

Regarding the system hardware configuration, our testbed consists of seven *IBM R-50* laptops with *Intel Pro-Wireless 2200* as integrated wireless card. All nodes use a Linux 2.6.12 kernel and run the OLSR_Unik implementation in version 0.4.7, which is fully compliant with the RFC 3626 and also implements additional modules to support explicit tunnelling between *ad hoc* nodes and gateways. The *ad hoc* nodes are connected through IEEE 802.11 *b* wireless links, transmitting at a maximum fixed rate of 11 Mbps. To generate asymptotic TCP traffic we used the *iperf* tool (http://dast.nlanr.net/Projects/Iperf/), while the saturated UDP traffic was generated with the *MGEN* tool. (http://cs.itd.nrl.navy.mil/work/mgen/) Different from other studies [5, 21], in which the network topology was only emulated 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 in Pisa. All the numerical results shown in the following are the average of five trials, each trial consisting of 5-min bulk TCP or UDP transfer.

5.1. Path life characteristics

A preliminary set of experiments was conducted to gather a better understanding of the OLSR behaviour, and the performance trade-offs between protocol overheads and responsiveness to network dynamics. The results of these tests are fundamental to isolate performance limitations depending on routing inefficiencies from the overheads introduced by the interconnection between the MANET and the fixed Internet. For these reasons, in the following we analyse how the OLSR parameters' setting influences the network performance and the path life in particular. The path life, whose formal definition will be introduced later in this section, is a measure of the routing protocol ability to maintain reliable and up-to-date topological information. Ideally, the routing protocol should be able to promptly react to link failures caused by radio link problems and node mobility, such as to eventually find alternative optimal routes. To this end, the OLSR routing protocol defines a set of procedures to monitor the link quality and to identify link and route changes, such as link sensing, neighbour detection and topology discovery. It is intuitive to note that the efficiency of these mechanisms has a significant influence on the route stability, the maintenance of end-to-end connectivity and the prompt reaction to topology modifications. Therefore, initially we investigated more in depth the impact of the routing protocol parameters on the performance of *ad hoc* networks, such as to identify an 'optimal' parameter setting to be used during the tests on the Internet access performance.

First of all, we can notice that the OLSR protocol periodically generates routing control packets in order to refresh the topology information. The behaviours of the various OLSR procedures are therefore regulated by a set of parameters that establish the timing for the OLSR operations. The default constant values for these parameters are defined in the OLSR RFC [4]. More precisely, each node generates, with a period equal to *HELLO_Interval*, HELLO messages to perform link sensing. The information provided in HELLO messages is considered valid for a *NEIGHB_HOLD_TIME*. Furthermore, periodic link reports, the TC messages, are generated by the MPRs. The validity time for TC message information is the *TOP_HOLD_TIME*, while the repetition period is the *TC_Interval*. Finally, each gateway, being connected to external networks, generates HNA messages, providing information on the reachability of non-OLSR networks. By analogy with previous parameters, *HNA_HOLD_TIME* and *HNA_Interval* are the validity time and repetition period of HNA messages, respectively. It is intuitive to observe that we may enhance the routing reactivity to topological changes by reducing the maximum time interval between periodic control message transmissions. However, the effect of using different parameter settings from the default ones has not been clearly quantified in the literature.

An additional feature of the OLSR protocol, which may impact the end-to-end connectivity characteristics, is the *link hysteresis* process [4]. The link hysteresis is a procedure designed to make the link sensing more robust against bursty losses or transient connectivity between nodes.

More precisely, the OLSR protocol computes for each link between a node and its neighbours a value, called *link_quality*, which measures the reliability of that link. A link is considered 'bad' if it allows HELLO messages to pass through it sometimes but not very often. An established link, i.e. a symmetric and reliable link, is considered pending and not usable or communications when its *link quality* goes below a fixed threshold, known as *HYST_THRESHOLD_LOW*. Note that a pending link is not considered broken because the link properties are still updated for each received HELLO message. On the contrary, a link is considered as lost when its validity time expires. In this case, the link is purged from the neighbourhood list. On the other hand, a pending link is promoted to the established status only when its *link_quality* goes above a fixed threshold known as *HYST_THRESHOLD_UP*. It is quite obvious that it should be $HYST\_THRESHOLD\_UP \geq HYST\_THRESHOLD\_LOW$. Figure 5 illustrates the link hysteresis behaviour. The diagram points out that when $HYST\_THRESHOLD\_LOW < link\_quality < HYST\_THRESHOLD\_UP$ the link status remains unchanged.

A key implementation requirement for the link hysteresis is the availability of an appropriate measure of the *link_quality*. If some measure of the signal/noise level on a received message is available (e.g. as a link layer notification), then it can be used as an estimation of the *link_quality* index (after being normalized to the range [0,1]). The OLSR specification [4] describes an alternative algorithm to estimate the *link_quality*, which does not require the use of link-layer information. This algorithm monitors the number of lost OLSR messages, Then, the exponentially smoothed moving average of the OLSR-packet transmission success rate is adopted as a measure of the *link_quality*. Formally, every time an OLSR message is correctly received $link\_quality = (1 - HYST\_SCALING) \cdot link\_quality + HYST\_SCALING$, where the *HYST_SCALING* value is the smoothing factor of the estimator, which is a number fixed between 0 and 1. When an OLSR message is lost, the instability rule [4] is applied, that is $link\_quality = (1 - HYST\_SCALING) \cdot link\_quality$. Note that the status of a new discovered link is initially set pending and its *link_quality* value is fixed to *HYST_SCALING*.

The behaviour of the hysteresis strategy is clearly determined by the specific setting of the algorithm parameters, and in particular by the memory size of the *link_quality* estimator and the threshold values. The OLSR specification suggests as default configuration $HYST\_THRESHOLD\_LOW = 0.3$ and $HYST\_THRESHOLD\_UP = 0.8$, and it adopts $HYST\_SCALING = 0.5$ as scaling factor. According to these values, even a perfect link (i.e. a link with $link\_quality = 1$) will be purged from the routing tables when two consecutive OLSR control 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 validate our intuitions and to investigate the influence of the duration of repetition periods on the path life characteristics, we performed a set of experiments considering various OLSR parameter settings. The network layout we used in our tests is depicted in Fig. 6. It is a chain topology consisting of five wireless links. The distances between the *ad hoc* nodes are set up in such a way to form a 5-hop chain topology with non-volatile wireless links. Node *GW* is the gateway node attached to the wired LAN. Although derived in a specific network scenario, we believe that our findings are applicable in generic situations because the chain topology is one of the most critical network scenarios for the OLSR protocol, which has been designed particularly for large and dense networks.

We tested the effect of using four different OLSR configurations, which are listed in Table 1. Our goal was to validate our intuitions on the over-pessimistic behaviour of the hysteresys process (the *set*1 configuration uses default OLSR parameters as in the *std* configuration, but the link hysteresys is disabled). In addition, we wanted to quantify the impact of reducing the time interval for periodic control message transmissions, and

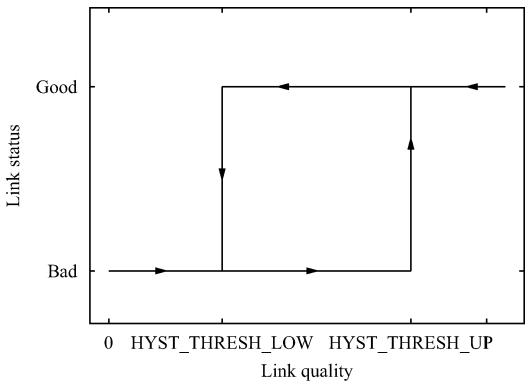

FIGURE 5: illustration of the hysteresis process

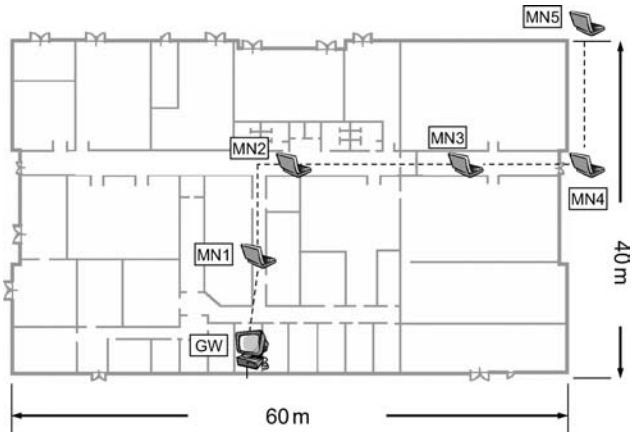

FIGURE 6: network layout used to conduct tests in static conditions

TABLE 1: OLSR parameter configurations

OLSR parameters	*std*	*set*1	*set*2	*set*3
HELLO_INTERVAL (s)	2	2	1	0.5
NEIGHB_HOLD_TIME (s)	6	6	6	6
TC_INTERVAL (s)	5	5	2.5	1.25
TOP_HOLD_TIME (s)	15	15	15	15
HNA_INTERVAL (s)	5	5	2.5	1.25
HNA_HOLD_TIME (s)	15	15	15	15
Hysteresis	Yes	No	No	No

to evaluate the trade-off between improved protocol reactivity and increased protocol overheads. Thus, in *set*2 and *set*3 configurations all the repetition intervals are twice and four times shorter than the default ones, respectively. Note that the validity times have not been changed from the default values indicated in [4]. As reference scenario we consider a network without data traffic, where the OLSR protocol is running using a default setting (hereafter indicated as *std** configuration).Due to space limits, in the following figures we show the experimental results related to a 3-hop TCP connection activated from node *MN*3 to node *GW* but similar results were also obtained for different path lengths.

The first performance index we measured during the tests was the OLSR overhead, defined as the average amount of OLSR traffic generated per unit time by each node (expressed in terms of base pairs). As Fig. 7 illustrates, the routing protocol overheads increase by reducing the generation periods of control traffic. It is evident that by halving the repetition period of OLSR messages we almost double the total routing overhead. However, the total overhead is negligible because it is always lower than 2 kbps. From the shown results, we notice that some nodes (i.e. *MN*1 and *MN*2) generate more control traffic than others. This is due to the fact that in the OLSR protocol only the MPRs generate link state reports, and in our experiments nodes *MN*1 and *MN*2 act as MPRs for the other nodes. Node *MN*3 generates the least OLSR control traffic because it is the end-point of the chain and it sends only HELLO messages. Node *GW* produces more routing traffic than node *MN*3 because it is the gateway node and it also sends HNA messages. On the other hand, it is less intuitive to explain why there is a slight reduction of protocol overheads when activating the TCP connection (*std* configuration) with respect to a network without data traffic (the *std** configuration). To provide clear reasons for this phenomenon, we should consider that each HELLO message contains the list of sending node's single-hop and two-hop neighbours, while each TC message contains the list of all the network links. Hence, the TC and HELLO message sizes depend on the number of links each node discovers. As we will better explain later, using the *std* configuration instead of the *std** increases the number of link timeouts suffered from the OLSR routing. Consequently, in the *std* case the routing messages deliver less topological information than in the *std** case. This explains the reduction in the generated overheads.

Figure 7 shows the amount of control messages produced by the routing protocol. However, not all the sent messages will be correctly received by the nodes' neighbours due to collisions, radio problems, and so on. Therefore, to understand the routing behaviours it is fundamental to evaluate the loss probability. In particular, we measured the loss probability of routing traffic in terms of the percentage of OLSR control messages that a node has sent but that its neighbours have not received. Figure 8 shows the average loss probability for OLSR traffic experienced by each node in the network. From the experimental results we observe that in the presence of heavy-loaded traffic the loss probability can be up to 20%. To explain these values we should note that the OLSR control traffic is encapsulated into UDP packets that are sent as broadcast frames. It is well recognized that the transmission of

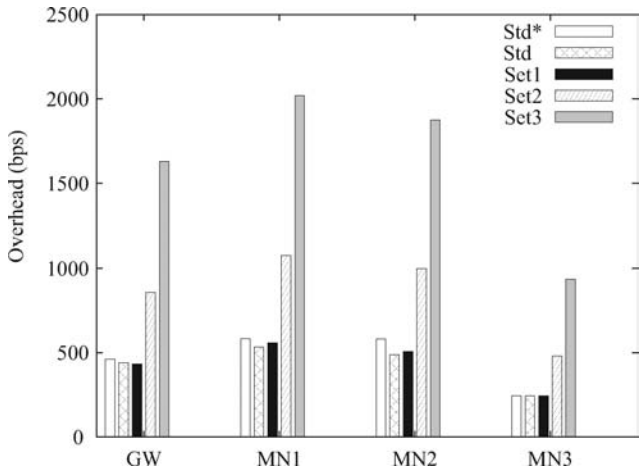

FIGURE 7: per-node OLSR protocol overheads (TCP case)

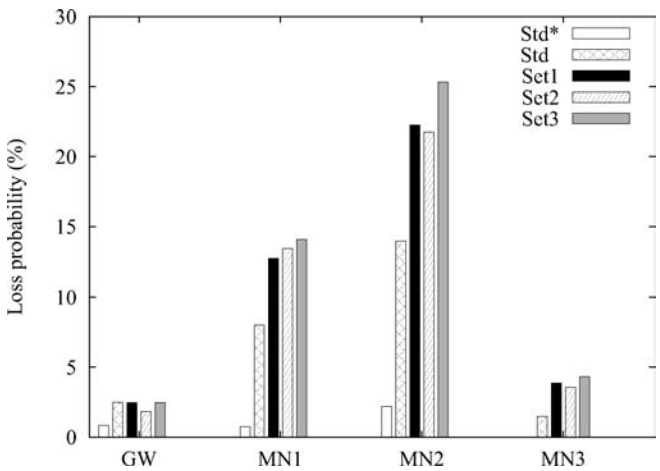

FIGURE 8: loss probability of OLSR control traffic (TCP case)

broadcast packets is unreliable on wireless channel. However, if we consider the *std** case, the loss probability is very small and always lower than 2%. It is safe to assume that this low number of losses is mainly due to channel noise and the lack of layer-2 retransmissions for broadcast frames. On the other hand, as soon as we introduce data traffic in the network the loss probability of OLSR packets experiences an upsurge. This can be explained by considering the increase of the collision probability in a heavy loaded network. While the MAC layer retransmit unicast frames to recover from congestion situations, broadcast frames are vulnerable to the collision events. Thus, the presence of data traffic inevitably degrades the performance of the OLSR routing protocol and its ability to efficiently distribute topology updates in case of radio problems or node mobility. Note that the loss probability of OLSR messages measured on node *MN*2 is higher than the loss probability measured on node *MN*1. By inspecting the experiment traces we discovered that this difference was due to the fact that link between node *MN*3 and *MN*2 was less reliable than the link between node *MN*2 and *MN*1. As a consequence the OLSR messages, which are not protected by layer-2 retransmissions, sent by node *MN*3 to node *MN*2 were more frequently subjected to channel losses than the ones sent by node *MN*2 to node *MN*1.

To quantify the degradation of the routing protocol performance we introduce the concept of path life. More precisely, this index measures the portion of time during which the source has a valid route to its intended destination. Figure 9 shows the path life of the route between node *MN*3 and node *GW*. When the OLSR routing is running without the interference of data traffic, the default parameter setting provides a 99% path life. However, the presence of unicast traffic generated by a single persistent TCP connection reduces the routing protocol reliability to 80%. The OLSR messages are generated by increasing the frequency. We rapidly counter-balance the negative impact of unicast traffic and channel noise on broadcast routing messages. From the experimental results, we observe that there is no gain in reducing more than four times the repetition intervals because the *set*3 configuration is sufficient to ensure a 100% path life between *MN*3 and *GW* nodes in static configurations.

We have replicated the same tests substituting the TCP traffic with asymptotic constant-bit-rate (CBR) UDP traffic. Our goal was to distinguish between the influence, if any, of flow-controlled elastic traffic (TCP) and unresponsive inelastic traffic (UDP) on the OLSR routing performance. From the experimental results we observe similar trends but with a general decrease of routing performance.

In particular, Fig. 10 illustrates the loss probability experienced by each node in the network in the same configurations used during the TCP case. From the experimental results we can notice that the loss probabilities are higher for the central nodes when using UDP traffic instead of TCP traffic, with an increase from a maximum loss probability of 20% to 35%. To explain this increase in the loss probability we can observe that CBR traffic does not regulate its sending rate to limit network congestion. Thus, even when the routing protocol suffers a link failure, the UDP flow continues to generate packets at the same rate. On the contrary, TCP traffic reduces its sending rate when node *MN*3 looses its route to node *GW*. This is beneficial for the routing control packets, which have more chances to be correctly received.

It is intuitive to note that this further reduction in the percentage of OLSR control packets that are successfully distributed to neighbours, will negatively impact the ability of the routing protocol to maintain a stable and reliable end-to-end connectivity. To validate our observations, in Fig. 11 we report the path life of the route between node *MN*3 and node *GW* when an asymptotic CBR UDP flow is delivering UDP traffic from node *MN*3 towards 30 node *GW*, saturating the wireless links. Concerning the *std* configuration there is a decrease from 80% to 35% of path life changing TCP with

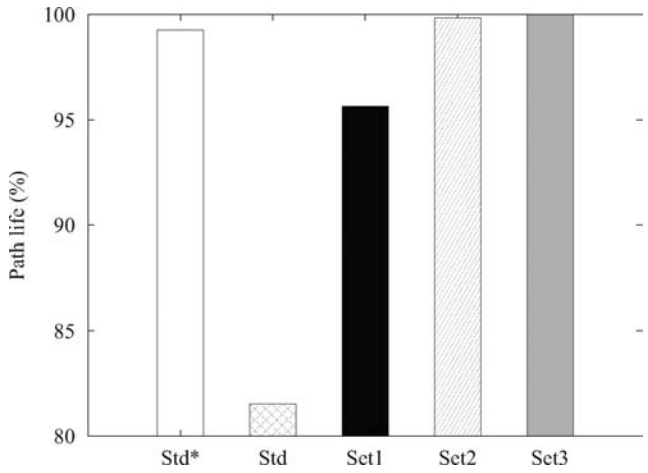

FIGURE 9: path life of the route between node *MN*3 and node *GW* (TCP case)

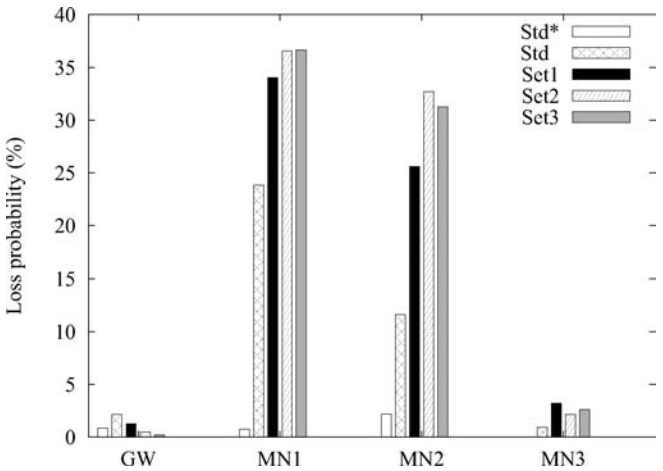

FIGURE 10: loss probability of OLSR control traffic (UDP case)

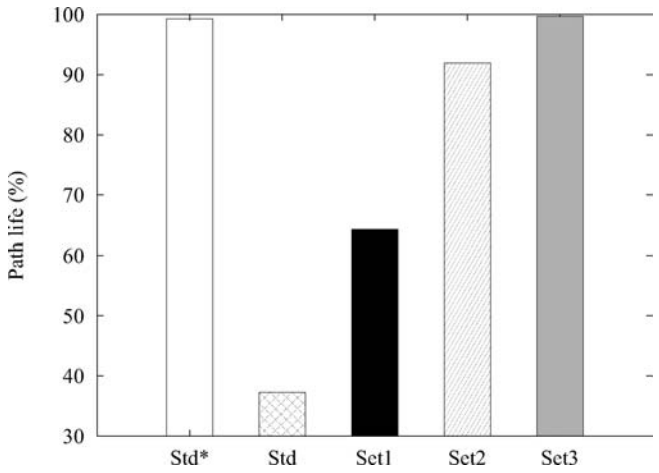

FIGURE 11: path life of the route between node *MN*3 and node *GW* (UDP case)

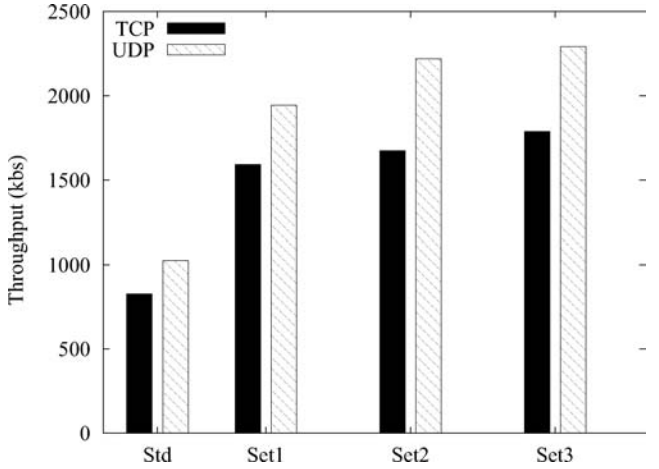

FIGURE 12: comparison of TCP and UDP throughputs for a 3-hop chain

UDP, while for the *set*1 configuration the decrease is from 95% to 65%. It is necessary to employ at least the *set*2 configuration to observe reliable routing behaviours.

To summarize our findings, in Fig. 12 we compare the TCP and UDP throughput for the various parameter settings considered in the experiments. First, we can note that the UDP throughput is always greater than the TCP throughput. This is obviously due to the additional overheads introduced by the TCP return traffic, which consists of TCP ACK packets. It is also straightforward to note that the improvement in the path life stability leads to a corresponding increase in the throughput performance. However, there is no additional gain by further increasing the repetition frequency of OLSR messages beyond the *set*3 configurations because the throughput curves flatten out. A further observation derived from the shown results relates to the different behaviour of TCP and UDP traffic due to the use of flow-control mechanisms in TCP flows. In particular, UDP flows utilize the channel resources in proportion to the path life. For this reason, when the path life is 35% (*std* setting) the UDP throughput is about 35% of the throughput obtained when path life is 100% (*set*3 setting). On the contrary, TCP behaviour is complicated by the use of flow-control that reduces the TCP sending rate when there is a packet loss. As a consequence, when path life is 80% (*std* setting) the TCP connections experience losses and the maximum retransmission timeout increases up to 16 s. This indicates that long time intervals separate consecutive retransmissions when the end-to-end connectivity is broken. Hence, even if the path is reestablished by the routing protocol the TCP flow may not be aware of this because it has to wait for the retransmission timer expiration before retransmitting the last sent segment. Note that the increase of path life from 80% to 95% (*set*1 setting) reduces the maximum TCP retransmission timeouts to 4.5 s. This implies that TCP connections can utilize, more efficiently, the available path without introducing useless delays between retransmissions. Finally, using the *set*3 parameter setting the route instability is almost null and the maximum TCP retransmission timeout is negligible (less than 0.3 s).

If not otherwise stated, in the subsequent sections, in which we compare the efficiency of our proposed solution for interconnecting *ad hoc* networks to fixed Internet with respect to the NAT-based solution defined in [5], we will configure the OLSR protocol using the *set*3 parameter setting.

5.2. Performance constraints of Internet access

To measure the performance limits of Internet access we conducted experiments in the network layout depicted in Fig. 6. However, different from the tests performed in Section 5.1, the final destination of the data traffic is not the *GW* node, but a server located in the wired part of the extended LAN. Our goal is to verify that our scheme introduces less overhead than a NAT-based scheme using explicit tunnelling between *ad hoc* nodes and gateways. The network performances are measured in terms of the throughput obtained by TCP and UDP traffic. The differences between the per-connection throughput measured using our proposed scheme and the NAT-based solution specified in [5] are used to quantify the protocol overheads.

The first set of experiments is aimed at evaluating the impact on the per-connection throughput of the number of wireless links traversed in the *ad hoc* domain before reaching the gateway. In these tests the IP packet size is constant and equal to 1500 bytes. Figure 13 compares the average throughput of a single TCP flow measured using our scheme (indicated with the label 'ARP-based' in the plots) or the one proposed in [5] (indicated with the label 'NAT-based' in the plots) versus the number of wireless hops needed to reach the gateway. Figure 14 reports the results obtained in similar configurations considering UDP flows. The parenthesis

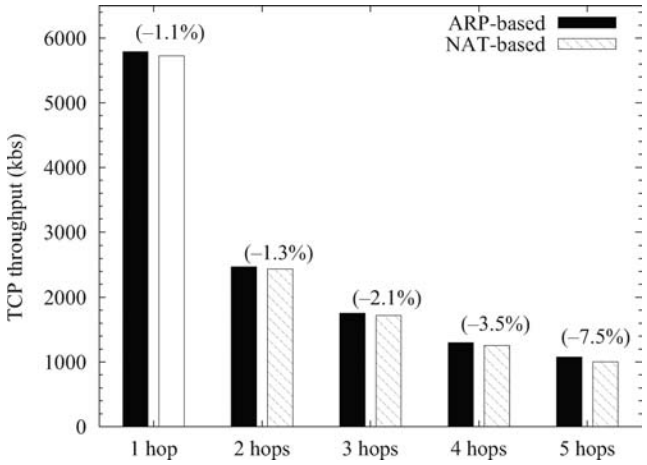

FIGURE 13: comparison of TCP throughputs versus the number of hops

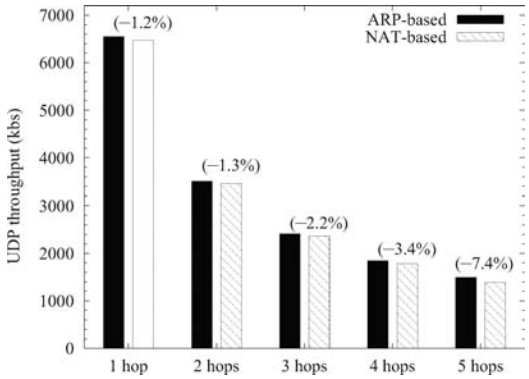

FIGURE 14: comparison of UDP throughputs versus the number of hops

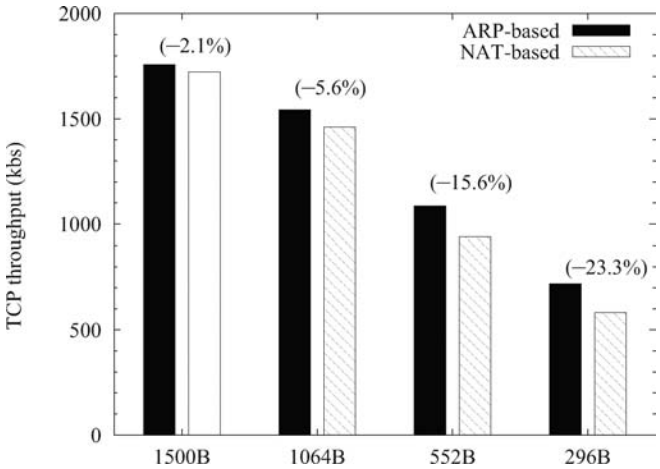

FIGURE 15: comparison of TCP throughputs versus the IP packet size

shows the percentage difference between the throughput measured in the 'NAT-based' case and the one measured in the 'ARP-based' case for each network configuration.

Several useful considerations can be drawn from the results shown. First, our scheme guarantees a higher per-connection throughout in all the considered network scenarios. This can be easily explained by noting that in the NAT-based scheme, the IP tunnel established between the sender node and the gateway uses IP-in-IP encapsulation. The additional IP header (20 bytes) can appear to be a small overhead when compared with the overall packet size. In fact, on a one-hop connection the throughput degradations are less than 2%. However, this overhead is replicated on all the links traversed by the packets. Hence, the more hops needed to reach the gateway, the higher is the throughput decrease. In fact, for a 5-hop connection the throughput measured in the NAT-based case is about 7% less than in the ARP-based scheme. Another interesting observation derived from the results shown is that in both schemes the UDP throughput decrease is almost proportional to the number of hops (i.e. the throughput of a n-hop UDP flow is about n times lower than the throughput of a one-hop UDP flow), while for the TCP flows the throughput reduction is greater. This is due to the self-interference between the TCP data packets and the return TCP acknowledgements, which are small packets reducing the channel capacity. It is worth pointing out that our experiments were conducted at the ground floor of the CNR building, where no access points were installed. However, it is still possible that the experimental measures would be affected by external interferences (e.g. access points located at higher floors of CNR building or nearby buildings, employees walking in the corridors, etc.). Hence, to guarantee homogenous channel conditions between experiments, we interleaved ARP-based tests with NAT-based tests.

In the previous experiments we considered an IP packet size equal to 1500 bytes, and we measure the maximum throughput achievable in each network configuration. However, it is widely recognized that in typical Internet traffic, the IP payload size is often smaller than the Ethernet maximum transmission unit [38]. It is intuitive to note that the overheads introduced by the IP-in-IP encapsulation needed to establish a tunnel between the sender node and the gateway degrades the throughput performance when the packet size decreases. To quantify this throughput reduction we conducted a second set of experiments measuring the throughput obtained by a 3-hop TCP and UDP connection versus the IP packet size, and we compared the results obtained with the ARP-based scheme and the NAT-based scheme. Figure 15 shows the throughput measured for a 3-hop TCP flow, while Fig. 16 shows the results in the UDP case. The results shown clearly indicate that the additional IP header added to the original packet is a significant overhead for small IP packets. For instance, when the IP packet size is 296 bytes, the throughput obtained using the NAT-based scheme is about 25% lower than the one measured in our scheme.

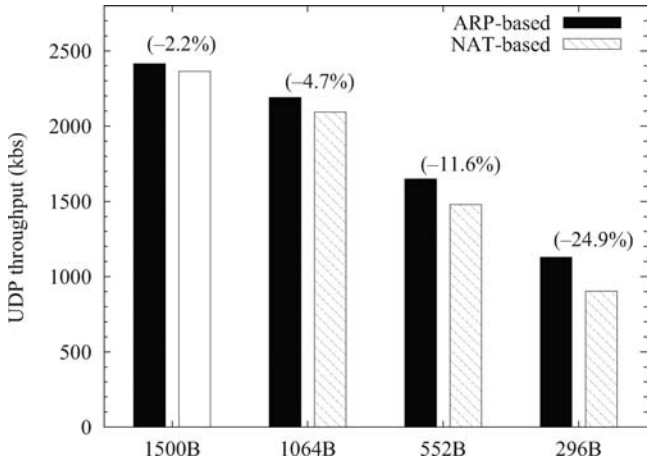

FIGURE 16: comparison of UDP throughputs versus the IP packet size

5.3. Performance constraints with gateway hand-offs

To test the efficiency of gateway hand-offs in a multi-homed network configuration we considered the network layout illustrated in Fig. 17. Specifically, we studied the system performance in low mobility/semi-static configurations, in which a single node roams between multiple gateways. Although the considered network scenario is not representative of high mobility conditions it is a typical example of multihomed network configurations because in the same *ad hoc* cloud the mobile node can reach two different gateways. A similar network configuration was also considered in [5].

The mobility test begins with the mobile node *MN*4 in position P1, where it is in radio visibility of gateway *GW*1. During the test, node *MN*4 has a TCP (or UDP) flow established with a host *H* in the wired LAN. Using the NAT-based scheme, when node *MN*4 is in position P1 it establishes a tunnel with gateway *GW*1, and this tunnel is permanently maintained throughout the experiment, independent of the *MN*4's position. On the contrary, using the ARP-based scheme, when node *MN*4 is in position P1 it sets up a default route to *GW*1 to reach the external fixed network, but the default gateway may change according to node mobility pattern. After 50 s it moves to position P2. The time required to move from one position to the next one is always 10 s. On location P2, node *MN*4 reaches the gateway *GW*1 through *MN*1, i.e. using a 2-hop wireless path. After other 50 s node *MN*4 moves to position P3. In this location gateway *GW*2 is two hops away, whereas gateway *GW*1 is three, hops away. Consequently, using our scheme, node *MN*4 switches to gateway *GW*2 to forward traffic addressing wired hosts. Moreover, the new default gateway *GW*2 begins to act as proxy ARP for the mobile node. On the contrary, using the NAT-based scheme, node *MN*4 continues to use gateway *GW*1, which is at a distance of three hops from node *MN*4. Finally, after other 50 s, host *MN*4 moves to position P4, where it is in radio visibility of *GW*2. However, the NAT-based technique forces node *MN*4 to tunnel its traffic towards gateway *GW*1 that is 3-hop away. Then, this movement pattern is repeated on the way back to position P1.

Figure 18 shows the throughput obtained by node *MN*4 in case of TCP traffic, while Fig. 19 shows the throughput obtained by node *MN*4 in case of UDP traffic. The experimental results confirm that both our scheme and the NAT-based solution may avoid a permanent TCP (or UDP) session break when the sender moves slowly. However, in our scheme this is achieved by supporting a transparent hand-off between the gateways *GW*1 and *GW*2, which does not require node *MN*4's address reconfiguration. On the contrary, the NAT-based solution described in [5] achieves this result by ensuring that the packets sent by node *MN*4 are always forwarded through gateway *GW*1, independent of the node *MN*4's position. In a particular network scenario we

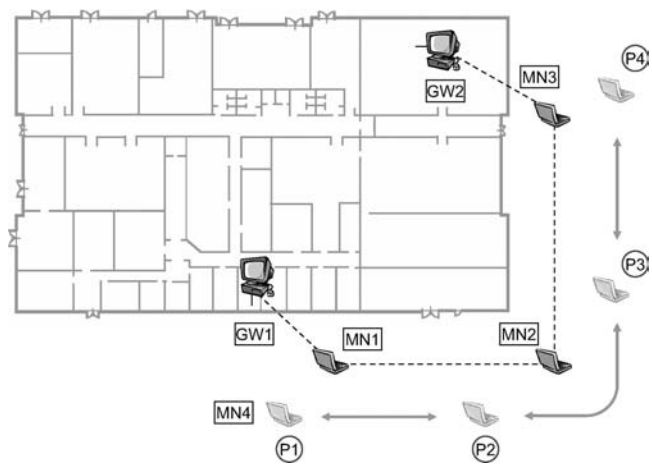

FIGURE 17: network layout used to conduct tests with node mobility

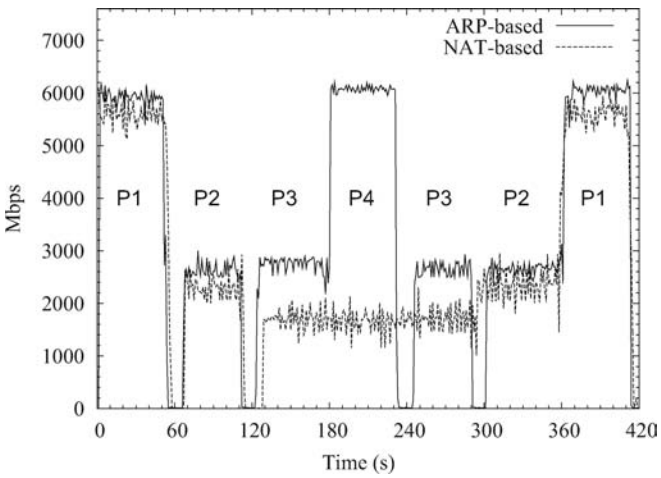

FIGURE 18: comparison of TCP throughputs when node *MN*4 moves

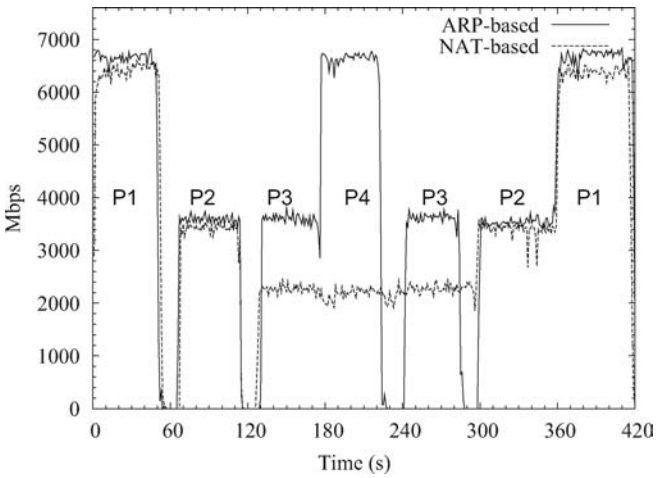

FIGURE 19: comparison of UDP throughputs when node *MN*4 moves

tested, this may be inefficient, because in position P3 and P4, node *MN*4 is closer to gateway *GW*2 than to gateway *GW*1 and it could use a shorter route to reach the wired LAN. For instance, in position P4 node *MN*4, using the NAT-based scheme, obtains a throughput that is three times lower than the one measured using our proposed solution.

Another interesting observation that can be derived from the experimental results regards the duration of the temporary connection breaks that may be caused by node movements and gateway handoffs. In particular, Figs 18 and 19 show that a TCP or UDP connection may be temporarily unable to successfully deliver packets for intervals close to 15 s when there is a change in the node *MN*4's routing table. The analysis of the causes of these throughput holes is quite complex because there is the interplay of different factors. Clearly, a fundamental role is played by the OLSR routing. For instance, let us consider the movement of node *MN*4 from position P1 to position P2. In this case, node *MN*4 continues to use gateway *GW*1 to reach the wired LAN, but now the gateway can be reached only through the node *MN*1. The routing protocol may quickly discover that the gateway *GW*1 is now reachable using a 2-hop route, but it will not immediately invalidate the old 1-hop route to *GW*1. In fact, node *MN*4 continues to consider valid and usable the shorter 1-hop route to *GW*1 until the validity time of the direct link to *GW*1 does not expire (the link timeout is 6 s by default [4]). After this timer expiration, node *MN*4 has to rebuild its routing table, and this may require a few seconds. On the contrary, when node *MN*4 moves from position P2 to position P1 we do not observe a temporary loss of connectivity. This can be explained by noting that, after this movement node *MN*4 gets closer to gateway *GW*1 (from two hops to one hop). According to the OLSR routing algorithm, when a shorter route is discovered, the routing table is immediately updated and recomputed. Similar reasoning can be used to explain routing behaviour during node *MN*4's movements between the other positions. Note that in our scheme the movement from position P2 to position P3 induces the change of default gateway for node *MN*4, while in the NAT-based scheme it causes the use of a 3-hop route instead of a 2-hop route to reach gateway *GW*1.

A second factor that affects the duration of throughput holes is the TCP flow control. In fact, when there is a link breakage, a TCP flow may suffer packet losses, increasing the retransmission timeouts. Hence, the delay introduced between two consecutive TCP retransmissions contributes to increase the interval during which no packets are sent on the channel. In other words, the correct route may be available but the TCP flow does not send packets because it is waiting for the expiration of the retransmission timer. Finally, a third aspect that influences the duration of throughput holes is the queuing delay. In fact, when there is a route change, the mobile node has to issue a new ARP_REQUEST to determine the mapping between the IP address and the physical address of the new neighbour. However, in case of asymptotic UDP traffic a large number of packets may be queued in the interface transmission buffer and the transmission of the ARP message may experience significant delays.

5.4. Lessons learned from the test-bed

Several lessons can be learned from the experiences gathered during the test-bed implementation and the analysis of experimental results. Our first observation refers to the trade-offs related to the use of IP tunnels within multi-hop *ad hoc* environments. This technique is adopted in many existing solutions for enabling interconnection between MANETs and the Internet. In fact, the use of IP tunnels is beneficial to provide transparent support of multi-homing and gateway hand-offs. On the other hand, IP tunnelling introduces evident inefficiencies in multi-hop environments because it impedes the use of the shortest paths available to reach the external networks. In multi-hop environments this may lead to significant throughput degradations.

Another relevant practical aspect that should be considered when comparing different solutions for enabling interconnectivity between Internet and multi-homed MANETs is the type of protocol overheads needed to support gateway hand-offs. In particular, existing solutions usually require the address reconfiguration when the mobile node changes the default gateway. This address reconfiguration requires time and may contribute to increase the duration of temporary session breaks. Mobile-IP based solutions clearly introduce higher overheads than NAT-based solutions because, upon moving, the mobile node should register the new care of-address with its home agent that may be far from the MANET. To reduce this heavy burden, the solution proposed in [21] introduces micromobility techniques. Our scheme does not require address reconfigurations and the gateway hand-off causes only an

update in the lists of IP addresses masqueraded by the proxy ARP servers running on the gateway.

Finally, the protocol complexity should also be taken into account when evaluating the feasibility of a proposal. For instance, solutions employing overlay networks of underlay layers below IP may introduce unacceptable implementation complexities. In addition, limiting the modifications to conventional IP mechanisms helps to easily implement the proposed solution under different environments and platforms.

6. CONCLUSION

This paper presented a practical and lightweight architecture to logically extend traditional wired LANs using multi-hop *ad hoc* networking technologies. The proposed architecture uses existing *ad hoc* routing protocols and can transparently interoperate with the fixed Internet infrastructure. We have shown that a simple approach exploiting proxy ARP servers and basic properties of the longest-matching rules used by standard IP routing is sufficient for establishing a heterogeneous network that appears to the external Internet as a single IP subnet. The protocol changes are quite limited and restricted to the gateway nodes. The experiments conducted in a prototyped system show that the proposed architectural design achieves an efficient use of network resources and it proves to be better than an alternative NAT-based scheme for the particular network topologies and traffic conditions we have tested.

We believe that there are several related aspects that are worth being further investigated in future work.

- The gateway selection procedure implicitly relies on the *ad hoc* routing protocol. In the case of OLSR, it is accomplished using shortest-path criteria. However, in a multihoming scenario, several gateways can exist, which may be implemented using different technologies and may have different capabilities. Thus, there could be many benefits in designing cooperative heuristics to select gateways such as to obtain load balancing within the *ad hoc* network, or more efficient handovers.
- In this work we have considered basic IP services, i.e. unicast routing and dynamic address allocation. However, more sophisticated functionalities, such as multicast and QoS management, have been developed for the Internet. Therefore, our proposed architecture should be extended to facilitate the integration of these additional capabilities.
- The address allocation scheme described in this paper allows the exploitation of DHCP servers to assign IP addresses that are topologically correct in the entire extended LAN. However, a detailed evaluation of the efficiency of our proposal and a comparative study with other auto-configuration scheme is required. In addition, we intend to explore how to extend our solution to deal with the typical problems that may arise due to node mobility, such as message losses, and network partitions and mergings.

REFERENCES

[1] Bruno, R., Conti, M. and Gregori, E. (2005) Mesh networks: commodity multihop ad hoc Networks. *IEEE Commu. Mag.*, **43**, 123–131.

[2] 802.11b (2001) Standard for Wireless LAN medium access control (MAC) and Physical layer (PHY) specifications. Amendment 2: higher-speed physical layer (PHY) extension in the 2.4 GHz band. IEEE.

[3] Plummer, D. (1982) *An ethernet address resolution protocol*. IETF RFC 826. http://www.ietf.org/rfc/rfc826.txt.

[4] Clausen, T. and Jaquet, P. (2003) *Optimized link state routing protocol (OLSR)*. IETF RFC 3626. http://www.ietf.org/rfc/rfc3626.txt.

[5] Engelstad, P., Tønnesen, A., Hafslund, A. and Egeland, G. (2004) Internet Connectivity for Multi-Homed Proactive ad hoc Networks. *Proc. of IEEE ICC'2004*, Paris, France, June 20–24, 4050–4056. IEEE Computer Society Press.

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

[7] Tschuding, C., Gold, R., Rensfelt, O. and Wibling, O. (2004) LUNAR: a Lightweight Underlay Network Ad-hoc Routing Protocol and Implementation. *Proc. of NEW2AN'04*, St. Petersburg, Russia, February 2–6, pp. 1059–1068. IEEE Computer Society Press.

[8] Draves, R., Padhye, J. and Zill, B. (2004) *The architecture of the Link Quality Source Routing Protocol*. Technical Report MSR-TR-2004-57. Microsoft Research, Redmond, WA.

[9] Perkins, C. (2002) *IP mobility support for IPv4*. IETF RFC 3344. http://www.ietf.org/rfc/rfc3344.txt.

[10] Ni, S., Tseng, Y., Chen, Y. and Sheu, J. (2002) The broadcast storm problem in a mobile ad hoc network. *Wireless Netw.*, **8**, 153–167.

[11] Broch, J., Maltz, D. and Johnson, D. (1999) Supporting hierarchy and heterogenous interfaces in multi-hop wireless ad hoc networks. Parallel Architectures, Algorithms, and networks. *Proc. of I-SPAN'99*. Fourth International Symposium on June 23–25, pp. 370–375.

[12] Johnson, D., Hu, Y. and Maltz, D. (2007) *The Dynamic Source Routing Protocol (DSR) for Mobile Ad Hoc Networks for IPv4*. IETF RFC 4728. http://www.ietf.org/rfc/rfc4728.txt.

[13] Jönsson, U., Alriksson, F., Larsson, T., Johansson, P. and Maguire Jr. G. (2000) MIPMANET – Mobile IP for Mobile Ad Hoc Networks. *Proc. of ACM MobiHoc 2000*, Boston, MA, USA, August 11, 75–85. ACM Press.

[14] Perkins, C., Belding-Royer, E. and Das, S. (2003) *Ad hoc On-Demand Distance Vector (AODV) Routing*. IETF RFC 3561. http://www.ietf.org/rfc/rfc3561.txt.

[15] Montenegro, G. (1998) *Reverse Tunneling for Mobile IP*. IETF RFC 2344. http://www.ietf.org/rfc/rfc2344.txt.

[16] Sun, Y., Belding-Royer, E. and Perkins, C. (2002) Internet connectivity for ad hoc mobile networks. *Int. J. Wireless Inf. Netw.*, **9**, 75–88. Special issue on 'Mobile Ad hoc Networks: Standards, Research, Application'.

[17] Ratanchandani, P. and Kravets, R. (2003) A Hybrid Approach to Internet Connectivity for Mobile Ad hoc Networks. *Proc. of IEEE WCNC 2003*, New Orleans, USA, March 16–20, pp. 1522–1527. IEEE Computer Society Press.

[18] Ahlund, C. and Zaslavsky, A. (2003) Integration of Ad Hoc Network and IP Network Capabilities for Mobile Hosts. *Proc. of ICT 2003*, February 23–March 1, pp. 482–489. IEEE Computer Society Press.

[19] Brannstrom, R., Ahlund, C. and Zaslavsky, A. (2005) Maintaining Gateway Connectivity in Multi-hop Ad hoc Networks. *Proc. of IEEE LCN 2005*, Sydney, Australia, November 15–17, 682–689. IEEE Computer Society Press.

[20] Perkins, C. (1996) *IP Encapsulation within IP*. IETF RFC 2003. http://www.ietf.org/rfc/rfc2003.txt.

[21] Benzaid, M., Minet, P., Al Agha, K., Adjih, C. and Allard, G. (2004) Integration of mobile-IP and OLSR for a universal mobility. *Wireless Net.*, **10**, 377–388.

[22] Srisuresh, P. and Holdrege, M. (1999) *IP Network Address Translator (NAT) Terminology and Considerations*. IETF RFC 2663. http://www.ietf.org/rfc/rfc2663.txt.

[23] AODV-UU Implementation. Version 0.6. http://core.it.uu.se/core/index.php/AODV-UU.

[24] Engelstad, P. and Egeland, G. (2004) NAT-based Internet Connectivity for On Demand MANETs. *Proc. of WONS 2004*, Madonna di Campiglio, Italy, January 18–23, Lecture Notes in Computer Science, pp. 4050–4056. Springer-Verlag.

[25] Perkins, C. (1996) *Minimal Encapsulation within IP*. IETF RFC 2004. http://www.ietf.org/rfc/rfc2004.txt.

[26] Thomson, S. and Narten, T. (1998) *IPv6 Stateless Address Autoconfiguration*. IETF RFC 2462. http://www.ietf.org/rfc/rfc2462.txt.

[27] Fan, Z. (2003) IPv6 Stateless Address Autoconfiguration in ad hoc Networks. *Proc. of PWC'03*, Venice, Italy, September 23–25, Lecture Notes in Computer Science, 665–678. Springer-Verlag.

[28] Wakikawa, R., Malinen, J., Perkins, C., Nilsson, A. and Tuominen, A. (2006) *Global connectvity for IPv6 mobile ad hoc networks*. Internet Draft.

[29] Jelger, C., Noel, T. and Frey, A. (2004) *Gateway and address autoconfiguration for IPv6 ad hoc networks*. Internet Draft.

[30] Ruiz, P., Ros, F. and Gomez-Skarmeta, A. (2005) Internet connectivity for mobile ad hoc networks: solutions and challenges. *IEEE Commun. Mag.*, **43**, 118–125.

[31] Ancillotti, E., Bruno, R., Conti, M., Gregori, E. and Pinizzotto, A. (2006) A Layer-2 Architecture for Interconnecting Multi-hop Hybrid Ad Hoc Networks to the Internet. *Proc. of WONS 2006*, Les Menuires, France, January 8–20, pp. 87–96. IFIP.

[32] Ogier, R., Templin, F. and Lewis, M. (2004) *Topology Dissemination Based on Reverse-Path Forwarding (TBRPF)*. IETF RFC 3684. http://www.ietf.org/rfc/rfc3684.txt.

[33] Vaidya, N. (2002) Weak Duplicate Address Detection in Mobile Ad Hoc Networks. *Proc. of ACM MobiHoc 2002*, Lausanne, Switzerland, June 9–11, 206–216. ACM Press.

[34] Nesargi, S. and Prakash, R. (2002) MANETconf: Configuration of Hosts in a Mobile Ad Hoc NEtwork. *Proc. of INFOCOM 2002*, New York, NY, June 23–27, 1059–1068. IEEE Computer Society Press.

[35] Weniger, K. and Zitterbart, M. (2004) Address autoconfiguration on mobile ad hoc networks: current approaches and future directions. *IEEE Netw.*, **18**, 6–11.

[36] Droms, R. (1997) *Dynamic Host Configuration Protocol*. IETF RFC 2131. http://www.ietf.org/rfc/rfc2131.txt.

[37] Carl-Mitchell, S. and Quarterman, J. (1987) *Using ARP to Implement Transparent Subnet Gateways*. IETF RFC 1027. http://www.ietf.org/rfc/rfc1027.txt.

[38] Crovella, M. and Bestavros, A. (1997) Self similarity in world wide web traffic: evidence and possible causes. *IEEE/ACM Trans. on Netw.*, **5**, 835–846.