Building an IP-based Community Wireless Mesh Network: Assessment of PACMAN as an IP Address Autoconfiguration Protocol

Carlos J. Bernardos [a,*] Maria Calderon [a] Ignacio Soto [a]
Ana Beatriz Solana [a] Kilian Weniger

[a] *Departamento de Ingeniería Telemática*
Universidad Carlos III de Madrid

Abstract

Wireless Mesh Networks are experiencing rapid progress and inspiring numerous applications in different scenarios, due to features such as autoconfiguration, self-healing, connectivity coverage extension and support for dynamic topologies. These particular characteristics make Wireless Mesh Networks an appropriate architectural basis for the design of easy-to-deploy community or neighbourhood networks. One of the main challenges in building a community network using mesh networks is the minimisation of user intervention in the IP address configuration of the network nodes. In this paper we first consider the process of building an IP-based mesh network using typical residential routers, exploring the options for the configuration of their wireless interfaces. Then we focus on IP address autoconfiguration, identifying the specific requirements for community mesh networks and analysing the applicability of existing solutions. As a result of that analysis, we select PACMAN, an efficient distributed address autoconfiguration mechanism originally designed for ad-hoc networks, and we perform an experimental study – using off-the-shelf routers and assuming worst-case scenarios – analysing its behaviour as an IP address autoconfiguration mechanism for community Wireless Mesh Networks. The results of the conducted assessment show that PACMAN meets all the identified requirements of the community scenario.

Key words: Community networks, Wireless Mesh Networks, Experimental evaluation, PACMAN

1 Introduction

Wireless Mesh Networks (WMNs) have emerged as a key technology for next-generation wireless networking [1], [2]. WMNs can have two types of nodes: mesh routers and mesh clients. Mesh routers – which present minimal or no mobility – constitute the backbone of the WMN, and some of them may have gateway functionality to connect the WMN with external networks (e.g., the Internet). Both mesh routers and mesh clients can forward packets on behalf of other nodes.

WMNs are dynamically self-organised and self-configured, with the nodes in the network automatically establishing a multi-hop ad-hoc network and maintaining the mesh connectivity. Autoconfiguration is an important feature from a deployment perspective, avoiding the need for manual intervention. Another interesting feature is its capability for self-healing, that is, the WMN is able to autonomously react to address a harmful, unexpected situation without the need for user intervention. Self-configuration and self-healing are two key features required to build WMNs that are both easy-to-deploy and robust.

There exist diverse application scenarios for WMNs, resulting in different WMN architectures. A WMN can consist of only mesh clients – commonly referred to as a Client WMN – only mesh routers – a Backbone WMN – or a combination of mesh routers and mesh clients – a Hybrid WMN [2]. One of the most promising application scenarios of *Backbone WMNs* today is what is known as *community networks*, where several users in a building or in a neighbourhood set up a WMN to communicate among themselves and share a number of access links (typically DSL or cable) to the Internet.

The community scenario demands a set of features that are naturally provided by a Wireless Mesh Network, namely:

- *Self-configuring and self-healing capabilities.* A community network should be able to bootstrap with little or no user intervention and to recover from certain failures.
- *Decentralised and unmanaged nature.* A community network should not rely

* Corresponding author.
Email addresses: cjbc@it.uc3m.es (Carlos J. Bernardos), maria@it.uc3m.es (Maria Calderon), isoto@it.uc3m.es (Ignacio Soto), anabeatriz.solana@alumnos.uc3m.es (Ana Beatriz Solana), kilian.weniger@googlemail.com (Kilian Weniger).
[1] The research of UC3M authors leading to these results has received funding from the European Community's Seventh Framework Programme (FP7/2007-2013) under grant agreement n^o 214994 (CARMEN project).
[2] The work of UC3M authors was also partially supported by the Spanish Government under the POSEIDON (TSI2006-12507-C03-01) project.

on any centralised entity that might potentially become a single point of failure. Since the devices that form the mesh network belong to different users, assuming a common management authority is not feasible.
- *Radio coverage extension ability.* The use of a multi-hop wireless network facilitates connectivity at locations where there is no Internet access infrastructure available.

A community Wireless Mesh Network consists of a set of fixed mesh routers providing connectivity to clients, and therefore it can be considered a Backbone WMN. This type of WMN is probably the most efficient and easy-to-deploy WMN, since it is not affected by routers' mobility and energy consumption constraints, because mesh routers are expected to be connected to a reliable power source at a fixed location.

In this paper, we first study how wireless interfaces of mesh routers can be configured in order to create community WMNs, highlighting the advantages and disadvantages of each possible configuration scheme. Then we select the best one from a deployment point of view, taking into consideration existing technologies and currently available devices on the market. The resulting architecture is used as the basis for our study of the IP autoconfiguration mechanisms in community WMNs.

Then, we identify and analyse the requirements that an IP address autoconfiguration solution aimed at a community WMN should meet. The results of this study are used in a subsequent analysis of the applicability of existing solutions [3], [4] – proposed within the area of Mobile Ad-hoc Networks (MANETs) – to the community scenario. One of these proposals is PACMAN (Passive Autoconfiguration for Mobile Ad-hoc Networks) [5]. PACMAN has all the features required for an efficient address autoconfiguration in the community WMN scenario. In particular, PACMAN is distributed, adapts to dynamic topologies, introduces very low protocol overhead, provides self-healing capabilities, and works in IPv4 networks [3].

Since PACMAN meets all the identified requirements for a solution aimed at working in a community WMN, we have performed an experimental analysis of a real-life implementation of PACMAN. Although a lot of effort has been devoted to propose solutions for the IP address autoconfiguration issue, little experimentation has been done with these protocols. Consequently, there is a need for analysis of the behaviour of these kinds of mechanisms in real test-beds in order to get a better insight into their behaviour. The main goal of this experimental evaluation is to analyse how a *real* autoconfiguration solution performs under different conditions in the community WMN scenario. All the experiments have been performed using off-the-shelf residential routers [4] which accurately represent real deployment environments.

[3] In this paper we focus on community networks that should be easily deployed nowadays. Therefore, we only consider IPv4 address autoconfiguration mechanisms.
[4] Linksys WRT54GSv4.

The rest of the article is organised as follows. In Section 2 we provide background information regarding community Wireless Mesh Networks, and analyse two key aspects that must be considered in their deployment: how to properly configure the wireless interfaces of the mesh routers and legacy clients to create a community WMN, and how to manage the IP address space used within the network. Section 3 tackles the IP address autoconfiguration, by first introducing a set of key features that should be provided, and then analysing whether existing proposed solutions meet the identified requirements or not. Section 3 also describes in detail the PACMAN protocol. Next, Section 4 is devoted to an experimental evaluation of PACMAN using off-the-shelf routers. Finally, we summarise the conclusions of our work in Section 5.

2 Wireless Mesh solutions for Community scenarios

It is not clear when the concept of community Wireless Mesh Networking appeared for the first time, since WMNs are closely related to Mobile Ad-Hoc Networks (MANETs). It is however clear that the area of mesh networking is now receiving quite a lot of attention, not only from the research community (e.g., Microsoft Self-Organizing Neighborhood Wireless Mesh Networks [5], Champaign-Urbana Community Wireless Network [6], roofnet [7]), but also from users and companies that are already building the first community mesh networks (e.g., Meraki [8], Open-Mesh [9]). In order to deploy usable community WMNs, there are many challenges that need to be tackled, such as routing, self-configuration and healing, radio planning, capacity handling, etc.

In this section we describe in detail the scenarios for community mesh networks looking at the configuration in layer 2 and how to manage the addressing at layer 3. The resulting architecture is the basis of the study of IP autoconfiguration requirements and solutions carried out in this article.

We assume a community scenario like the one depicted in Figure 1. The current model to provide Internet access from homes consists of individual users having their own access router that is equipped with an Internet interface (through xDSL, cable, etc.) and an interface to connect with user devices (e.g., laptops). This latter one is typically a wireless IEEE 802.11.

A WMN solution allows increasing the flexibility and functionality of the previous scenario. With a WMN, the mesh routers can connect among themselves, improv-

[5] https://research.microsoft.com/mesh/
[6] http://www.cuwireless.net/
[7] http://pdos.csail.mit.edu/roofnet/doku.php
[8] http://meraki.com/oursolution/mesh/
[9] http://open-mesh.com/

ing the communication inside the community, and enabling the sharing of Internet access links among its users. In fact, with this solution we do not need an access link (xDSL, cable, ...) per router/home, since available links in different homes can be shared by all community users.

2.1 Layer-2 architectures to create community WMNs

One important design consideration in community WMNs is the configuration of the involved layer-2 technologies, since this aspect has an impact on the type of devices (hardware) required to set up the WMN, the efficiency in the use of radio resources and the resulting layer-2 topologies. A deployment requirement is that the hardware complexity for the WMN scenario should not increase significantly in comparison with the current home Internet access scenario. So, we assume that only IEEE 802.11 technology will be used for communications inside the WMN. In addition, all or part of the mesh routers will be connected to the Internet using some technology such as xDSL or cable. Depending on the number and configuration mode of the IEEE 802.11 wireless interfaces we have identified the following four main backbone mesh deployment options:

(1) Mesh routers equipped with only one wireless interface, operating in ad-hoc mode. Wireless interfaces of both mesh routers and conventional end devices are configured to operate in ad-hoc mode. We should note that even with this approach, we do not want end-user devices to take part in the mesh routing operations, and therefore an additional mechanism is required at layer-3 to allow end-user devices to identify and configure a mesh router as their Internet gateway. The main advantage of this approach is the reduced cost of the mesh routers, since they only are required to have one wireless interface. In fact, this allows the use of currently available access routers for residential applications. On the other hand, the drawback is that the radio resources are used inefficiently, because only one of the available radio channels can be used in the WMN.

(2) Mesh routers equipped with two wireless interfaces, one operating in ad-hoc mode and the other in infrastructure mode. In this case, mesh routers configure one wireless interface in infrastructure mode, as an Access Point [10] (AP) serving conventional clients that might attach to it, while the other wireless interface is configured in ad-hoc mode to be part of the community WMN. This approach does not restrict the possible mesh topologies, but as in the previous case, it comes at the price of suboptimal use of the available radio channels.

(3) Mesh routers equipped with two wireless interfaces, both operating – preferably in two different non-overlapping channels – in infrastructure mode. As

[10] Another analogous configuration – easier to achieve from the point of view of today's available devices in the market – is to use a router with one wireless interface, and one wired interface, to which a simple Access Point (in bridged mode) is connected.

before, one interface is configured in AP mode to provide connectivity to other devices – both conventional clients and mesh routers – while the other is configured in station (STA) mode, to connect to other mesh routers. This approach provides better use of available radio channels, while limiting the flexibility of the community network (i.e. the number of potential network topologies is restricted by the fact that a wireless interface configured in STA mode cannot be simultaneously connected to more than one AP). It is also worthwhile mentioning that a WMN configured in this way would likely require some layer-2 autoconfiguration mechanisms to setup optimal – or at least efficient – mesh network topologies.

(4) Mesh routers equipped with more than two wireless interfaces, one configured in infrastructure mode and the others in ad-hoc mode. The interface working in infrastructure mode is configured as an AP to provide connectivity to conventional user devices. The rest of the wireless interfaces working in ad-hoc mode are used for connections to other mesh routers. Having more than one ad-hoc interface allows the creation of links in different channels achieving a more efficient use of the radio spectrum, while still being able to connect any pair of mesh routers by configuring them with a common channel in one of their interfaces. The advantage of this solution compared with the previous one is a better flexibility in the creation of mesh topologies. The disadvantage is an increase in hardware requirements. This solution and the previous one share the disadvantage of requiring a complex configuration for setting up the layer-2 topology.

Those previous configurations that involve the use of more than one wireless interface can be achieved using a recent solution offered by some commercial products allowing the creation of more than one virtual interface from just one network card. For example, this can be used to have one STA and one AP using the same wireless physical interface. These solutions represent a trade-off between efficiency and cost, and do not change the conclusions of the analysis in this section.

In this article we have selected the second deployment option, since it provides a reasonable trade-off between network topology flexibility and use of radio resources, while keeping layer-2 configuration complexity low – which is an important concern in this scenario–. It is easy to build community networks of this type today using for example Linksys WRT54GSv4 devices and additional Access Points (if it is required to provide wireless access to conventional clients). No particular layer-2 configuration mechanism is needed to set up a mesh topology, since the routers will be able to communicate with any other mesh routers within their radio coverage using the ad-hoc interface.

2.2 IP address space management

Once a layer-2 mesh topology is available, we have to consider the management of the IP address space in the mesh. We basically need IP addresses for:

(1) the user devices, that connect to a mesh router to obtain network access.
(2) the interfaces used by the mesh routers to communicate among them. Mesh routers – forming the backbone WMN – use these addresses and run a routing protocol – probably an ad-hoc routing protocol – to enable the communication among them.
(3) the communications with devices outside the mesh (i.e. on the Internet). These pose the need for globally reachable addresses.

Globally reachable addresses will be provided by the Internet Service Providers, one per each access link to Internet. But we cannot expect to have global addresses for covering the other needs of the scenario. A solution to solve this issue is to use the IPv4 private address space.

One possible approach is the utilisation of the same IPv4 address space both for the user devices and for the mesh routers interfaces (points 1 and 2 above). However, this presents the disadvantage of making the user devices' addresses configuration dependent on a community-wide address space management. Such a management would require coordination at the community network level for the configuration of the IPv4 address of a user device.

A better approach is to separate the end-user devices private address space from the mesh routers address space (see Figure 2), that is, use two different address spaces. The IP addresses of the end-user devices can be configured locally with the support of each mesh router, by running a DHCP server. This is a straightforward solution because it is the currently deployed approach for single-hop scenarios (i.e. a gateway providing IP connectivity to directly attached clients). Besides, it has the important additional advantage of not requiring any changes in the end-user devices. Every mesh router must run a Network Address Translator (NAT) to translate from the private addresses used by the conventional IP devices attached to it, to the private addresses used in the WMN. In order to configure the IP addresses used in the backbone, an IP address autoconfiguration mechanism is required, to ensure that there are no duplicated addresses in the backbone mesh. Consequently, both address spaces are managed independently and the IP addresses of the end hosts do not affect the address autoconfiguration of the mesh routers in the WMN.

Finally, a mesh router with an access link (e.g., DSL or cable) to an external network (i.e. an Internet Gateway – IGW), will have a NAT functionality performing the following translations (see Figure 2):

(1) from the end-user devices IP private addresses to the backbone WMN private

IP address. This type of translation is performed by all mesh routers, including those that do not have a direct connection to the Internet.
(2) from the WMN IP private addresses to the public IP address configured in the mesh router (assigned by its Internet Service Provider – ISP), and
(3) from the end-user devices (conventional IP terminals) IP private addresses to the mesh router public IP address (assigned by its ISP).

Translation 3 is the one performed by most residential gateways nowadays, whereas the first two are specific to the community WMN scenario. Translation 1 takes place when traffic from a device attached to the mesh router is routed towards its destination through the WMN (i.e. either the IGW functionality for this traffic is performed by another mesh router within the community network, or the traffic is intended for a node locally attached to the same community network). Translation 2 is performed when the mesh router is acting as an IGW for IP traffic from another node within the community mesh network.

In this scenario, the remaining configuration challenge is to provide mesh routers with the IPv4 addresses required to communicate among themselves, ensuring the uniqueness of the configured private addresses. This must be done through an automatic procedure requiring little (if any) user intervention.

3 IP address autoconfiguration for community WMNs

This section focuses on the problem of IP address autoconfiguration for community WMNs, using as a reference the community mesh scenario defined in the previous section, both in terms of layer 2 configuration and IP address management. We identify the requirements for an IP address autoconfiguration mechanism, review existing proposals, and select a candidate solution meeting all the requirements of our scenario.

3.1 IP address autoconfiguration required features

We have identified the following key features that should be taken into consideration when designing/evaluating an IP address autoconfiguration mechanism for community WMNs:

(1) **Support for dynamic topology.** In general, community WMNs have a dynamic topology, since the routers can be connected or disconnected unexpectedly (i.e. the owner/administrator switches nodes off/on), or new nodes are added/removed.
(2) **Self-healing.** A community WMN should be able to autonomously react and

solve harmful, unexpected problems without the need for user intervention. This is a key feature in order to build robust WMNs. In the particular case of IP address autoconfiguration schemes, the network should be able to detect and solve duplicated addresses (i.e. two nodes using the same IP address). These conflicts could appear due to two main reasons:

(a) Network merging. Even with an IP address autoconfiguration mechanism to ensure that each mesh router initially autoconfigures a different IP address, this uniqueness needs to be continuously checked during the WMN lifetime, since it might happen that two isolated networks join to form a single one (this situation is commonly referred in ad-hoc literature as network merging). To illustrate an example of WMN merging, we might think of a community network formed by equipment belonging to several neighbours of a 10-stories building. In this scenario, depending on the availability of the neighbours' routers, it is possible that several isolated WMNs networks are formed (e.g., a WMN cloud formed by routers on 1st to 5th floor and another one formed by routers on 7th to 10th floor). These isolated networks may merge if a router on the 6th floor is switched on, and it could happen that the two initially isolated networks had some common IP addresses configured, resulting in an address conflict after the merging.

(b) User misconfiguration. Address conflicts might also appear as a consequence of manual configuration mistakes. In an environment so open and unmanaged as a community network scenario, it is not unlikely that a user decides to manually configure its own router. The user may choose an IP address that is already in use in the WMN, and therefore the autoconfiguration mechanism used by the WMN routers should detect the address duplication and fix it (by changing the address of the WMN router that is running the autoconfiguration protocol).

(3) **Scalability.** The scalability with respect to configuration time (and also protocol overhead) when the number of nodes increases is an important concern. Community WMNs topologies range – in terms of dimension and number of nodes – from small to large (i.e. from several tens to hundreds of nodes).

(4) **Low overhead.** An IP address autoconfiguration solution may use some control signalling (e.g., message flooding). Given the wireless nature of community WMNs, this protocol overhead may have a significant impact on the performance. Thus, low protocol overhead is considered a key feature of the IP address autoconfiguration protocol. Processing overhead should be kept reasonably low, since protocol operations are implemented in mesh routers that should be low-cost devices, although not necessarily extremely limited devices.

3.2 Applicability of existing solutions

In this section, we describe and briefly analyse some existing IP address autoconfiguration proposals that could be applied to the community WMN scenario.

Since WMNs and MANETs share several key characteristics, some of the solutions proposed for IP address autoconfiguration in the field of MANETs may be also applicable to community WMNs. There is a plethora of existing proposals of MANET IP address autoconfiguration solutions [4], but not all of them are suitable for community scenarios. For example, a significant number of the proposed solutions so far only support IPv6, which is unacceptable for community WMNs nowadays. Even the IETF AUTOCONF Working Group, chartered in 2005 to tackle the problem of IP address autoconfiguration for MANETs, is only aiming at standardising IPv6 mechanisms.

We next review existing IPv4 address autoconfiguration solutions, analysing the capabilities they provide and their basic operation. There are solutions (such as [6], [7], [8]) that require a node to perform a particular procedure – called *pre-service* Non-Unique Address Detection [9] – before configuring a new IP address on one interface, to ensure that a candidate address (that is typically chosen randomly from a known pool) is not being used by other nodes within the same network. Most commonly, pre-service Non-Unique Address Detection mechanisms consist in sending some messages asking if the candidate address is in use or not, and waiting for a potential reply (if such a reply is not received, that is interpreted by the sending node as a hint that the candidate address is not being used by any node of the network and therefore the candidate IP address can be assigned to one of its interfaces). This kind of solution presents several disadvantages, specially when applied to community WMNs, since it requires additional signalling (that might be significant depending on the scenario) and it makes use of timeouts while message delays cannot be bounded in an ad-hoc network (even if it is possible, determining the delays is non-trivial).

On the other hand, there are some solutions (such as [10], [11], [12]) that ensure (to a certain extent) that addresses are unique when they are assigned to an interface. This can be done by using other means, such as statistical properties or use of disjoint address pools, etc. [11] .

Ensuring that IP addresses are unique at the moment of their assignment is not enough for all WMN scenarios, and in particular it is not for community WMNs. As we have already mentioned, an IP address conflict might appear, for example, as a result of a network merging or a user misconfiguration. Because of that, mechanisms that detect and solve duplicated IP addresses not only initially, but in a con-

[11] Meraki for example uses the following addressing scheme: nodes configure IP addresses that are the static hash of the MAC address onto the entire 10.0.0.0/8 private network.

tinuous way, are also needed. These mechanisms are commonly referred to as *in-service* Non-Unique Address Detection. There are basically two main ways of performing in-service Non-Unique address detection: actively – for example by means of periodic messaging [8] – or passively, by means of detecting address conflicts from routing protocol anomalies. Solutions intended for community WMNs can benefit from the use of passive in-service Non-Unique Address Detection mechanisms in order to save wireless bandwidth.

Another important characteristic is the centralisation degree of the solutions. Some solutions may assume the existence of a centralised infrastructure/entity or assign a special role to certain nodes [11], while others can be completely distributed, not relying on any special node/infrastructure to carry out the autoconfiguration task. Since a community WMN is clearly an decentralised and unmanaged environment, it seems more appropriate to make use of a distributed solution.

In conclusion, an IP address autoconfiguration solution intended to be deployed in a community WMN should have the following features:

- *IPv4 support:* since community networks are meant to be easily deployed nowadays, an IP autoconfiguration solution must be able to provide IPv4 addresses.
- *In-service Non-Unique Address Detection:* community networks must be able to self-heal from any potential address conflict that might appear, for example because of network merging or user misconfiguration. Therefore, solutions only performing pre-service Non-Unique Address Detection are not suitable for the community scenario.
- *Passive nature:* due to the scarce wireless bandwidth that is likely to be available in community WMNs, it is better to minimise bandwidth waste due to the use of active signalling to detect IP address conflicts.
- *Distributed nature:* since community networks are clearly decentralised and unmanaged, an IP address autoconfiguration solution must not rely nor assume the existence of any kind of centralised infrastructure.

As it is described in Section 3.3, PACMAN fulfils all these four requirements, making it an appropriate candidate protocol for community scenarios.

3.3 Passive Autoconfiguration for Mobile Ad-hoc Networks (PACMAN)

PACMAN [5] is a fully distributed address autoconfiguration mechanism for ad-hoc networks that aims to guarantee unique IP [12] addresses in the network even in the presence of network merging. It uses cross-layer information from ongoing routing protocol traffic. The basic idea is that a router joining the mesh network assigns an address to itself by randomly picking one from the set of yet unassigned

[12] Although we focus on IPv4, PACMAN works both for IPv4 and IPv6.

addresses according to the router's local knowledge, and relying on the Passive Duplicate Address Detection (PDAD) concept to detect conflicts originating from this optimistic address assignment or from network merging. The mesh router may learn about already assigned addresses by monitoring the routing protocol traffic or by requesting a list of addresses that are known to be assigned in the network from a neighbour router.

The components of PACMAN are the following. A routing protocol packet parser that extracts information from incoming routing protocol packets and hands them to other PACMAN components for further processing. Since PACMAN is routing protocol dependent, the protocol parser is itself modular to support different routing protocols.

An address assignment component that selects an IP address using a probabilistic algorithm. It also maintains an allocation table containing addresses that are already assigned to other mesh routers. The assignment component considers the allocation table to minimise the conflict probability. The table is passively updated based on incoming routing protocol packets.

A Passive Duplicate Address Detection (PDAD) component that detects potential address conflicts, e.g., occurring after two networks merged. A difficulty for the passive detection of address conflicts based on routing protocol packets is that a mesh router typically also receives routing protocol packets that contain the router's own address, e.g., packets that were forwarded by other mesh routers and originated by the receiver. Hence, if a router receives a routing protocol packet containing its own address, it is difficult to figure out whether this address is unique and used by the receiving router only or whether it is duplicate and use by another router in the mesh network as well.

PDAD is a core functionality of PACMAN and defines a set of rather simple algorithms that allows mesh routers to detect address conflicts in the network based on routing protocol anomalies. The basic idea of PDAD is to exploit the fact that some protocol events occur in case of duplicate addresses in the network, but (almost) never in case of unique addresses. PDAD does not send any control packets. Instead, each mesh router analyses incoming routing protocol packets for anomalies and detects a conflict, if the packet contains a duplicate address.

A specific combination of algorithms is used to detect all conflicts in the network running a specific routing protocol. More than ten PDAD algorithms are proposed in [13] and [5], which in combination are able to detect conflicts in MANETs running various routing protocols, in particular Optimized Link State Routing (OLSR), Ad-hoc On-Demand Distance Vector Routing (AODV), and Fisheye State Routing (FSR).

An example of a PDAD algorithm is the PDAD-Neighbourhood History (NH). The basic idea of this algorithm is to exploit the bidirectionality property of link-states

in link-state routing protocols like OLSR. If a mesh router receives a routing protocol packet with its own address as part of the set of bidirectional link-states of the originator, the originator must have been a neighbour of this router recently. Otherwise, another mesh router has the same address and the address is duplicated in the network. This algorithm requires that all routers have to record their recent neighbourhood history in an NH table.

Another example of a PDAD algorithm is the PDAD-sequence number (SN) algorithm, which uses sequence numbers in the routing protocol packets to detect duplicate addresses in the network. In most routing protocols, each mesh router originating routing protocol packets uses a sequence number only once (except for sequence number wrap-arounds) and each node increments its own and only its own internal sequence number counter. Under these assumptions, if a router receives a routing protocol packet originating from its own address and with a sequence number higher than its internal sequence number counter, the originator must be another router in the mesh network which has the same address as the receiver.

In case a mesh router detects a conflict of another router's address, the conflict resolution component notifies the respective router, which can then change its address to resolve the conflict.

PACMAN meets all the requirements in the community WMN scenario: it provides an efficient distributed IPv4 address autoconfiguration mechanism, able to cope with the sources of dynamism in this environment (addition/deletion of nodes, network merging), scalable with the number of routers, that provides self-healing capabilities against misconfiguration by users or network merging, and that has both very low protocol and reasonable low processing overhead.

There are other proposed IP address autoconfiguration mechanisms that follow a passive approach, such as [14] and [15]. Since these solutions are based on the same passive approach, it is expected that they could also be applicable to the community scenario. In this paper, we have chosen PACMAN as the solution to be evaluated because it was a pioneer solution among passive approaches, and because there was an open source implementation available. This software could be modified to be run in our community WMN test-bed setup – using off-the-shelf routers, and then used in our experimental evaluation.

4 Experimental evaluation

In this section we present the results of an experimental evaluation of PACMAN as IP address autoconfiguration mechanism for community networks – running OLSR as routing protocol within the community mesh network –, and using low cost off-the-shelf devices.

4.1 Experimental setup

The PACMAN version used in this experimental evaluation is an open source implementation for Linux [13] . It implements PDAD for OLSR and parser modules for multiple OLSR routing protocol implementations. This allows the use of PACMAN with an unmodified UniK OLSR [14] routing daemon. The PDAD module intercepts received routing protocol packets using the Linux netfilter hooks.

To perform our experiments we built a test-bed composed of 30 Linksys WRT54GSv4 routers. This small residential router is equipped with a 200 Mhz processor, an IEEE 802.11g WLAN interface and an IEEE 802.3 Ethernet interface connected to a VLAN capable 5-port switch. This is a very popular low-cost router, which provides a suitable platform for creating and testing community WMNs, since its firmware is released under the GNU GPL and so it can be easily modified [15] .

In the experiments, we made use of one of the wired interfaces of the router to perform management operations, such as local time synchronisation of all the routers, remote execution of tests and results retrieval for off-line processing. This avoids the impact of these operations on the network interfaces being autoconfigured by PACMAN during the experiments.

4.2 Experimental results

4.2.1 Single-hop

We first analyse the time required by a community WMN to be globally configured when it is initially bootstrapped (this is the most stressful case that can be considered in a real-life scenario, since all the involved nodes are activated and try to configure their IP addresses at the same time). The convergence time of the network after bootstrapping is the time required by the last node in the network to configure a unique IP address.

In this first set of experiments, we used the scenario shown in Figure 3, which involves a variable number of nodes (from 2 to 30), while keeping the number of IP addresses that are available for use fixed (the 192.168.0.0/27 pool [16]). In all the

[13] It can be obtained from `http://pacman-autoconf.sourceforget.net/`. Our work was performed with `pacman` v1.32.
[14] `http://www.olsr.org/`. Our work was performed with `olsrd` v0.4.9-1, configured as proposed in the OLSR specification [16].
[15] For these tests, we used the open source *OpenWRT* WhiteRussian RC 3 distribution (available at `http://www.openwrt.org/`).
[16] With this address pool size, the number of valid IP addresses is equal to the maximum number of devices that we might have on the network: 30. This is obviously the worst-case

experiments described in this article, a minimum of 30 executions were performed for each test, in order to obtain statistically meaningful results.

Convergence time results are illustrated in Figure 4. We observe that the number of nodes has an impact on the results, showing an increase in convergence time as the number of nodes gets larger. This is an expected result, mainly because the probability of two or more nodes randomly choosing the same IP address increases with the number of nodes, since in our test-bed the number of available IP addresses is fixed. The worst-case situation is that of 30 nodes, with no free IP address available after the convergence of the network, but even in that case, the convergence time is about 12 seconds (this is the time required by the last node in the network to configure a unique IP address). In addition to this, the average time required by a node to configure a unique IP address was also measured (this is basically the elapsed time that a node waits until it obtains IP connectivity), being about 300 milliseconds when the network consists of 2 nodes and 2.5 seconds for the case of 30 nodes (see Figure 5).

A mesh router running PACMAN may try different addresses before getting a non-duplicated one that can be used to gain IP connectivity. Figure 6 shows the average number of IP addresses that a node tries before getting a unique one. We observe that – on average – a node needs less than two attempts to get a valid IP address. We also measured the average maximum number of configurations attempts of the IP address that a node does (see Figure 7), and we observe that this number is close to 6 for the worst-case scenario (30 nodes with only 30 IP addresses available for the whole community network).

Related to the previous two metrics, there is a third performance metric that might have an impact on the overall scalability of a WMN, namely the fraction of nodes that require to be reconfigured before a steady state is reached. This metric reflects how stable the autoconfiguration mechanism is (see Figure 8). As expected, the probability of a node to reconfigure its IP address is related to the address collision probability, which depends on the number of nodes and the available address pool size.

It is important to highlight that all these tests have been conducted considering a bootstrapping scenario in which all the participant nodes boot at the same time. This is obviously a worst-case scenario, that reflects how the solution performs and scales under extreme conditions. During the steady operation of an already configured community WMN, the most common situation involving any change on the IP autoconfiguration, will consist of WMN routers joining and leaving (e.g., because a mesh router is switched on/off by its owner). Thus, the previously analysed results are worse than those that would be obtained when nodes just occasionally join and leave the network.

possible scenario.

4.2.2 Multiple-hop

Besides analysing how PACMAN performs when the number of nodes (and the relative ratio of nodes divided by available IP addresses) increases, it is also important to evaluate how it behaves as the size of the network – in terms of its diameter (i.e. number of hops) – is augmented.

The deployment of an experimental real-life test-bed to perform such an evaluation would require a very large physical area, in order to ensure that multiple-hops are used to communicate several mesh routers. Because of that, we adopted the following approach:

(1) A two-hop wireless set-up. This scenario basically involves two end mesh routers, initially configured with the same IP address (192.168.0.1). They cannot reach each other directly, but through a third router within their radio coverage. Using this 3-node WMN (see Figure 9), we measured the time required to solve the initial IP address conflict after bootstrapping the network (this time includes the time OLSR needs to bootstrap the network). Again, we are analysing a worst-case scenario, to actually evaluate the usability of PACMAN under stressing conditions. The results show that the time required by PACMAN to detect and solve an IP address conflict in this scenario is about 4.5 seconds. The same experiment using a single-hop set-up (that is, the two nodes are directly reachable without traversing any intermediate node) results in a conflict resolution time close to 2.6 seconds.

(2) Due to the large area that would be required to perform experiments involving several real wireless hops, we could not replicate the previous experiments in a test-bed involving more than 2 hops. In order to gather some qualitative insight about the behaviour of PACMAN in multi-hop environments with more than 2 hops, we set up a test-bed like the one shown in Figure 10, where two different 1-hop wireless mesh clouds are interconnected by means of a set of wired-connected routers (these routers use IP addresses from a different address space than the wireless mesh routers within each cloud). It should be noted that this scenario differs from the one considered in this article for community WMNs in several ways: it requires mesh routing protocols to run on multiple interfaces (for example, this may have an impact on the OLSR performance), wired links are used (therefore not suffering from the typical radio issues) and PACMAN cannot be run on the intermediate hops, due to a limitation on the software implementation used [17] . Despite these differences, conducted tests provide us with some initial results. The goal of these experiments is twofold: first, by performing the tests, it is possible to check the

[17] By not running PACMAN in all the nodes, some of the PDAD algorithms defined to detect IP address conflict cannot be used, and basically only the conflicting nodes would become aware of address conflicts, since intermediate hops are not running PACMAN. This adversely impacts the measured performance.

correct operation of PACMAN in relatively large (in terms of diameter) networks. Second, we get some results that, given the aforementioned experiment limitations – such as the impediment of running PACMAN on all nodes – can be considered as worst-case scenario results. Using this set-up, we measured the time required by two nodes initially configured with the same IP address, to detect and fix that address conflict, when the WMN is bootstrapped. This experiment was repeated several times, increasing the number of intermediate hops. The results show that less than 20 seconds are required to detect and solve the initial IP address conflict within WMNs with a diameter of up to 20 intermediate hops.

4.2.3 Network Merging

In this section we experimentally analyse how PACMAN performs – in terms of recovery time – on situations of network mergers. We considered a scenario consisting of two independently formed and configured WMNs which are isolated from each another (see Figure 11). These two unconnected clouds (composed of 14 and 15 nodes) are then merged by introducing a new node that is within radio coverage of both clouds. The same IP address pool (192.168.0.0/27) was used in all the nodes of the scenario and we forced one node at each isolated network to have the same IP address configured (192.168.0.1) so we ensured that an address conflict always occurred when the two networks merged.

The correct behaviour of PACMAN was checked under this extreme scenario, composed of 30 nodes after network merging, while the total amount of available addresses is also 30. Therefore, no IP address will remain available after the merging and this forces the network nodes to change several times the IP address they are trying to configure to avoid duplication. Results indicate that that the time required to completely configure the network (that is, no duplicated address used by any node) is around 100 seconds (on average, each mesh router needs 12 seconds to be configured with a unique IP address). This long delay is caused by the fact that the analysed scenario severely limits the number of available IP addresses. Results also show that about 33% of the nodes changed their IP address before getting a unique one, and that some of them had to change it more than twice before succeeding.

Based on these results, we conclude that PACMAN provides good self-healing capabilities, being able to recover from massive IP address duplications even in extreme scenarios.

5 Conclusion and future work

The design of appropriate IP address autoconfiguration mechanisms is the first step required to allow community WMNs to became a reality. There exist a plethora of proposals that tackle the autoconfiguration problem from the classical point of view of ad-hoc networking. It is important to revisit this problem from the particular perspective of community WMNs, paying special attention to those features that are critical for this kind of environment.

In this article we have analysed the deployment of WMNs using current residential routers. In this context, we have investigated the ability of PACMAN – a mechanism developed for IP autoconfiguration in ad-hoc networks – to satisfy the requirements that an IP address autoconfiguration protocol for community WMNs should meet. Our investigation was performed based on experiments using a real life test-bed that have given us insight into the behaviour of PACMAN under extreme conditions (e.g., during bootstrapping of the network, when the available address space is small relative to the number of nodes in the network, etc.), using resource-limited off-the-shelf devices. The obtained results show that PACMAN provides self-healing capabilities, while supporting dynamic topologies and keeping the protocol overhead very low (almost null, due to its passive nature). The protocol has been shown to scale well in our experiments. Although our test-bed did not involve a large number of devices, given that we conducted the tests under extreme conditions in terms of available IP addresses and that PACMAN presents a very low overhead, we are confident that the solution will also work in larger deployments.

More extensive experiments, including a test-bed with real users and more complex mesh topologies, are the focus of our future research.

Acknowledgements

The authors would like to thank Pablo Serrano, Alberto García-Martínez and Jose Felix Kukielka for their helpful comments that contributed to the improvement of this paper.

References

[1] S. Faccin, C. Wijting, J. Kneckt, A. Damle, Mesh WLAN Networks: concept and System Design, IEEE Wireless Communications 13 (2) (2006) 10–17.

[2] I. Akyildiz, X. Wang, W. Wang, Wireless mesh networks: a survey, Computer Networks 47 (4) (2005) 445–487.

[3] Y. Sun, E. Belding-Royer, A study of dynamic addressing techniques in mobile ad hoc networks, Wireless Communications and Mobile Computing 4 (3) (2004) 315–329.

[4] C. J. Bernardos, M. Calderon, H. Moustafa, Survey of IP address auto-configuration mechanisms for MANETs, Internet Engineering Task Force, draft-bernardos-manet-autoconf-survey-04.txt (work-in-progress) (November 2008).

[5] K. Weniger, PACMAN: passive autoconfiguration for mobile ad hoc networks, IEEE Journal on Selected Areas in Communications 23 (3) (2005) 507–519.

[6] C. E. Perkins, J. Malinen, R. Wakikawa, E. M. Belding-Royer, Y. Sun, IP Address Autoconfiguration for Ad Hoc Networks, Internet Engineering Task Force, draft-ietf-manet-autoconf-01.txt (work-in-progress) (November 2001).

[7] N. H. Vaidya, Weak Duplicate Address Detection in Mobile Ad Hoc Networks, in: MOBIHOC'02, 2002, pp. 206–216.

[8] J. Jeong, J. Park, H. Kim, D. Kim, Ad Hoc IP Address Autoconfiguration, Internet Engineering Task Force, draft-jeong-adhoc-ip-addr-autoconf-06.txt (work-in-progress) (January 2006).

[9] H. Moustafa, C. J. Bernardos, M. Calderon, Evaluation Considerations for IP Autoconfiguration Mechanisms in MANETs, Internet Engineering Task Force, draft-bernardos-autoconf-evaluation-considerations-03.txt (work-in-progress) (November 2008).

[10] H. Zhou, L. M. Ni, M. W. Mutka, Prophet Address Allocation for Large Scale MANETs, in: Proceedings of INFOCOM 2003, 2003.

[11] A. Misra, S. Das, A. McAuley, S. Das, T. Technol, Autoconfiguration, registration, and mobility management forpervasive computing, Personal Communications, IEEE 8 (4) (2001) 24–31.

[12] M. Mohsin, R. Prakash, IP address assignment in a Mobile Ad Hoc Network, in: Proceedings of MILCOM 2002, Vol. 2, 2002.

[13] K. Weniger, Passive duplicate address detection in mobile ad hoc networks, in: Proceedings of the IEEE Wireless Communications and Networking (WCNC), 2003.

[14] E. Baccelli, OLSR Passive Duplicate Address Detection, Internet Engineering Task Force, draft-clausen-olsr-passive-dad-00.txt (work-in-progress) (July 2005).

[15] K. Mase, C. Adjih, No Overhead Autoconfiguration OLSR, Internet Engineering Task Force, draft-mase-manet-autoconf-noaolsr-01.txt (work-in-progress) (April 2006).

[16] T. Clausen, P. Jacquet, Optimized Link State Routing Protocol (OLSR), Internet Engineering Task Force, RFC 3626 (Experimental) (October 2003).

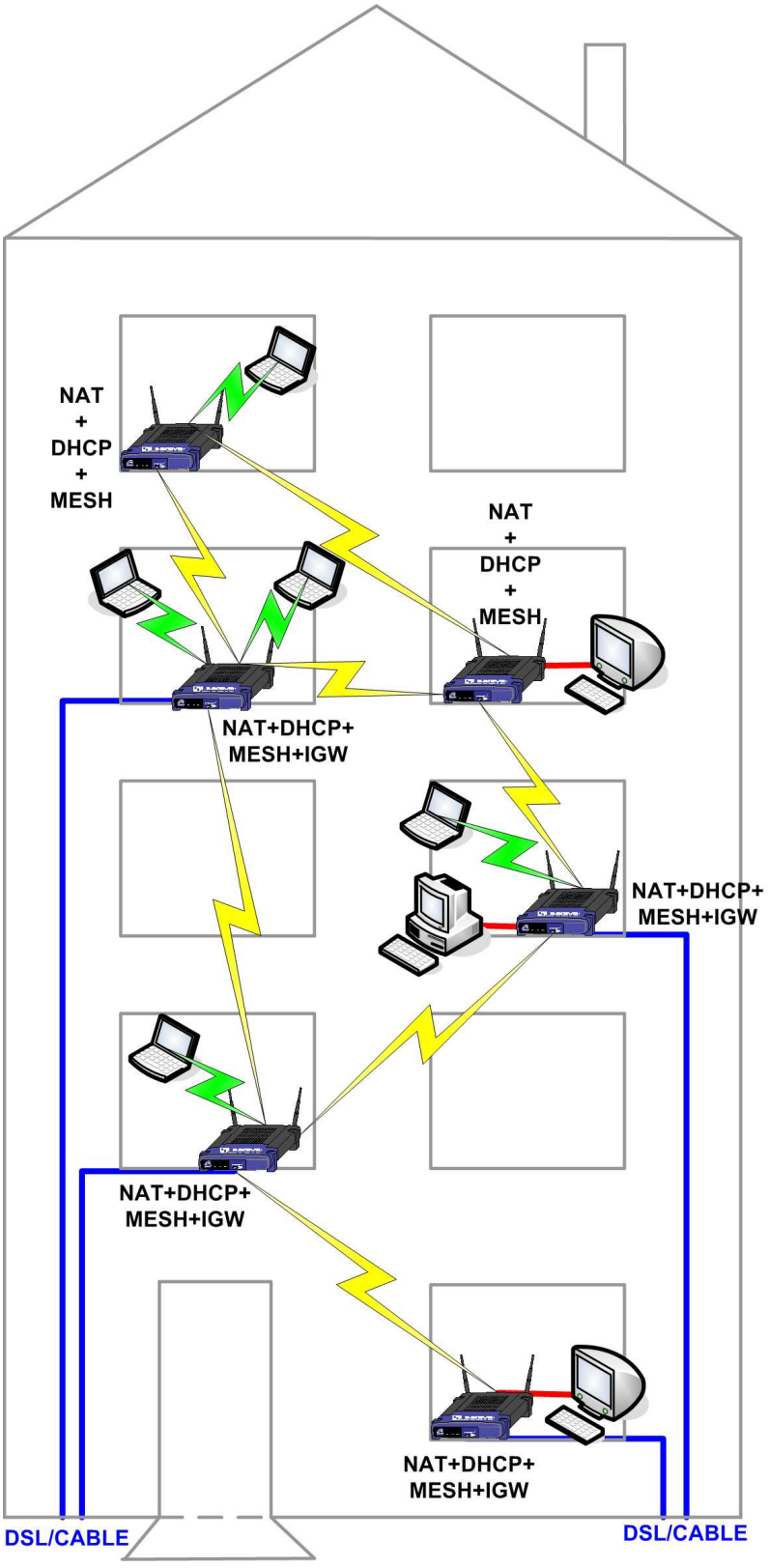

Figure 1. Community Wireless Mesh scenario

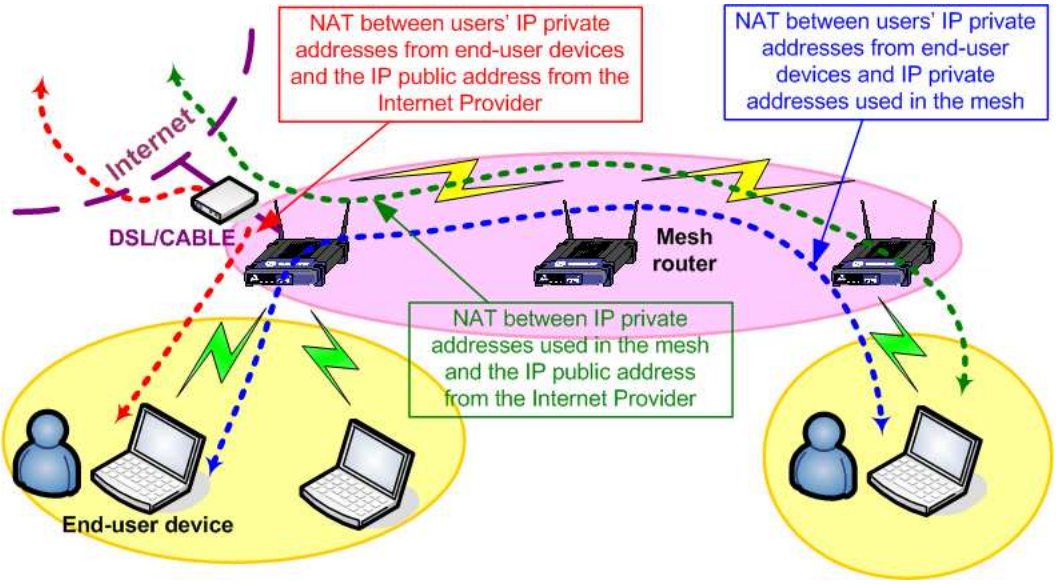

Figure 2. Community WMN IP addressing approach

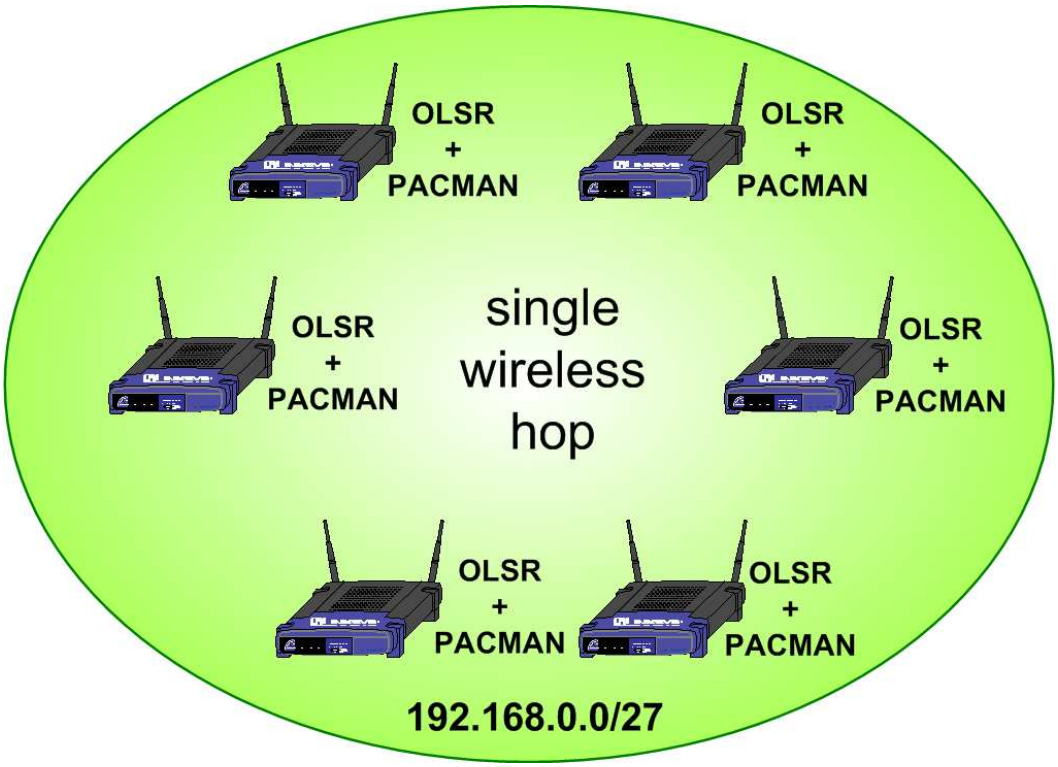

Figure 3. Single-hop scenario

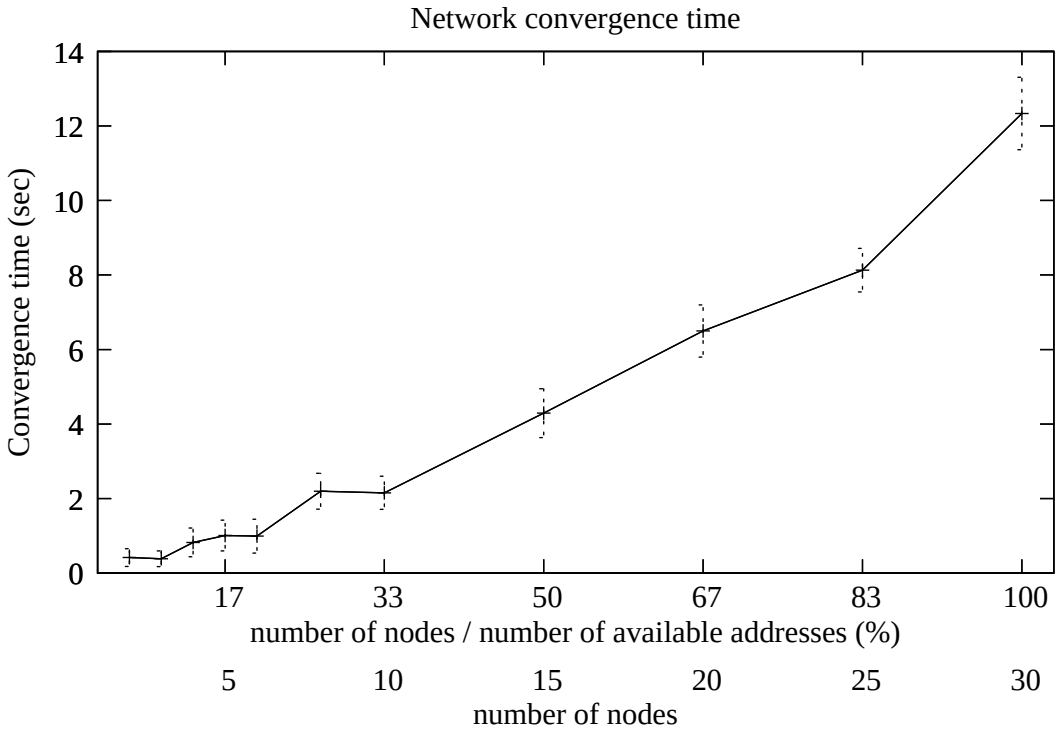

Figure 4. Average network autoconfiguration convergence time (single-hop bootstrapping scenario)

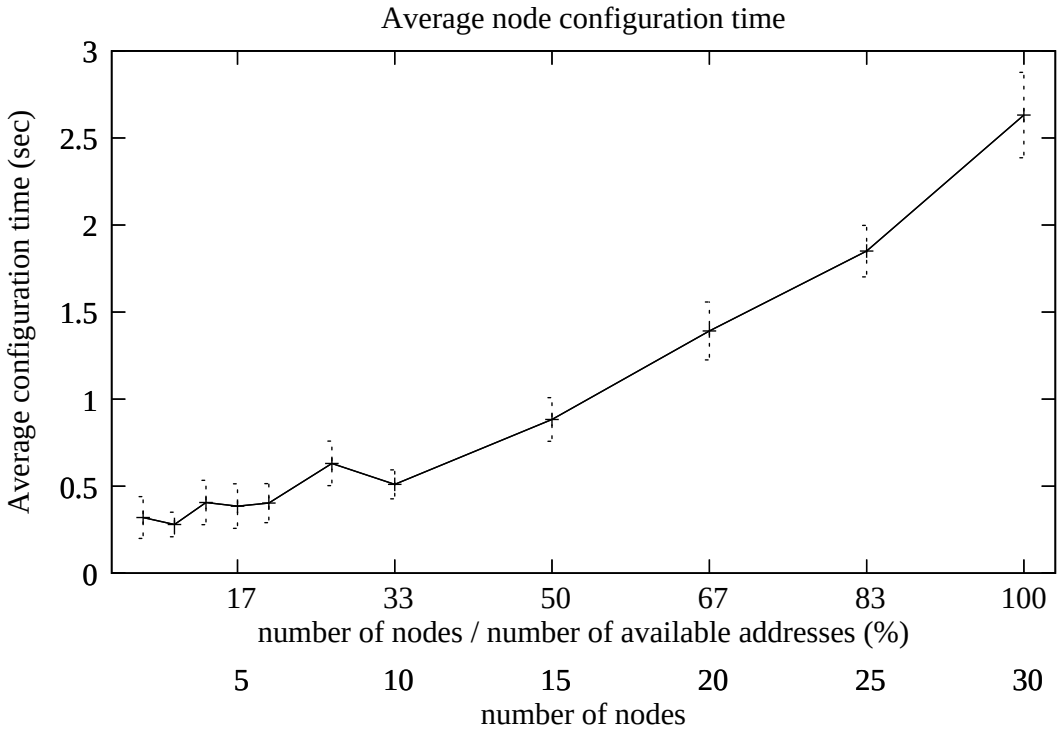

Figure 5. Average node configuration time (single-hop bootstrapping scenario)

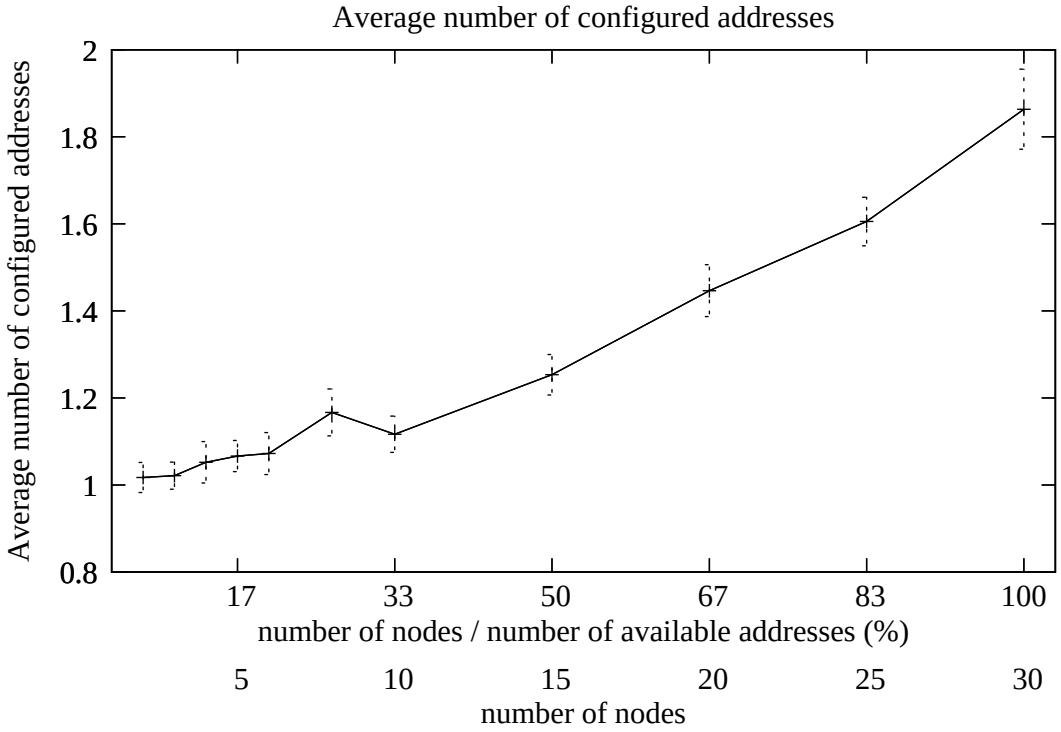

Figure 6. Average number of configured IP addresses (single-hop bootstrapping scenario)

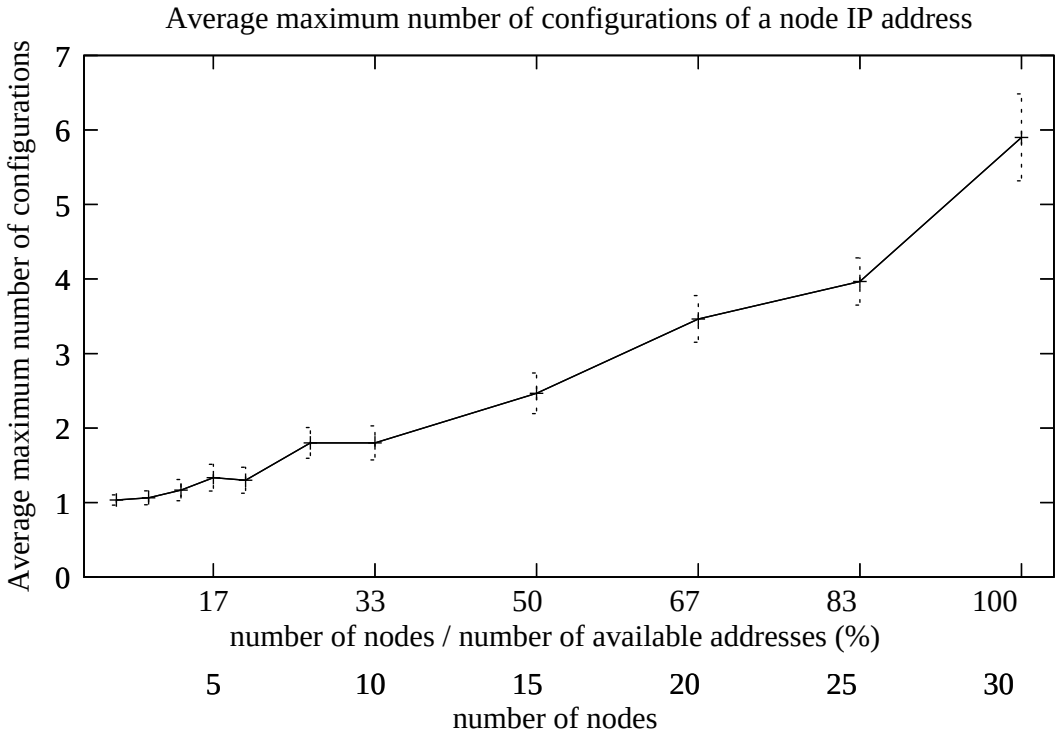

Figure 7. Average maximum number of configurations of the IP address of a node (single-hop bootstrapping scenario)

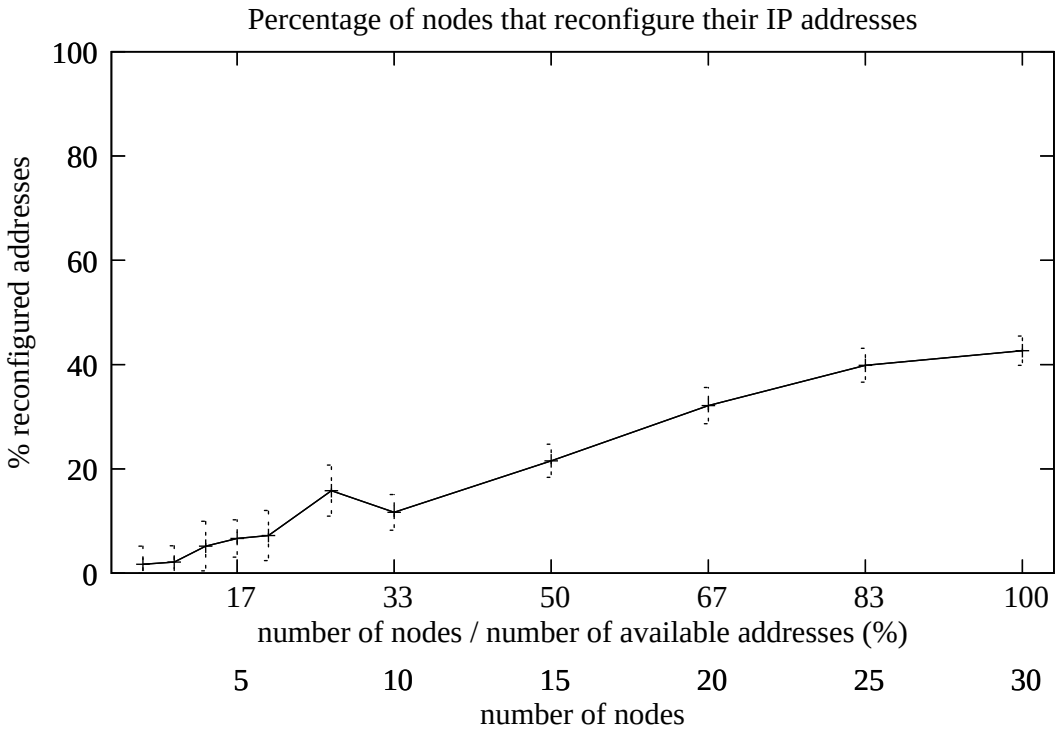

Figure 8. Percentage of nodes that needed to reconfigure their IP address (single-hop boot-strapping scenario)

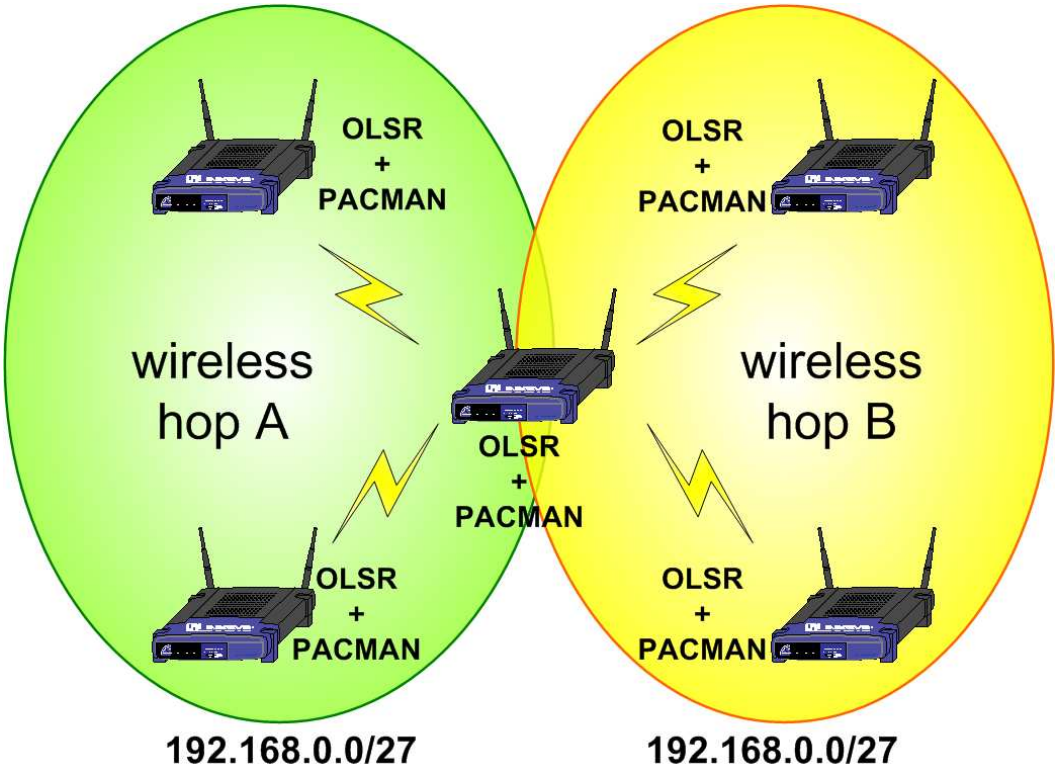

Figure 9. 2 wireless hops scenario

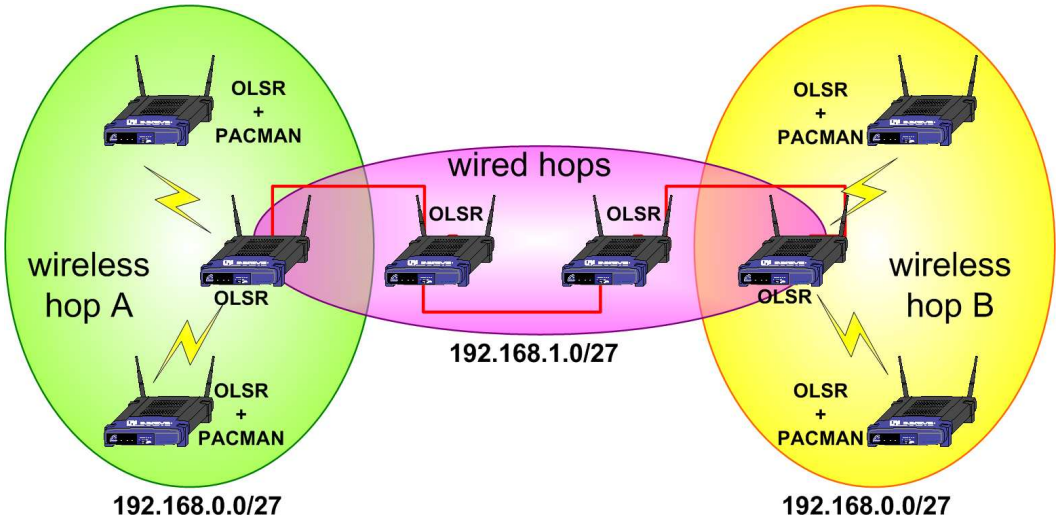

Figure 10. Multiple-hop scenario

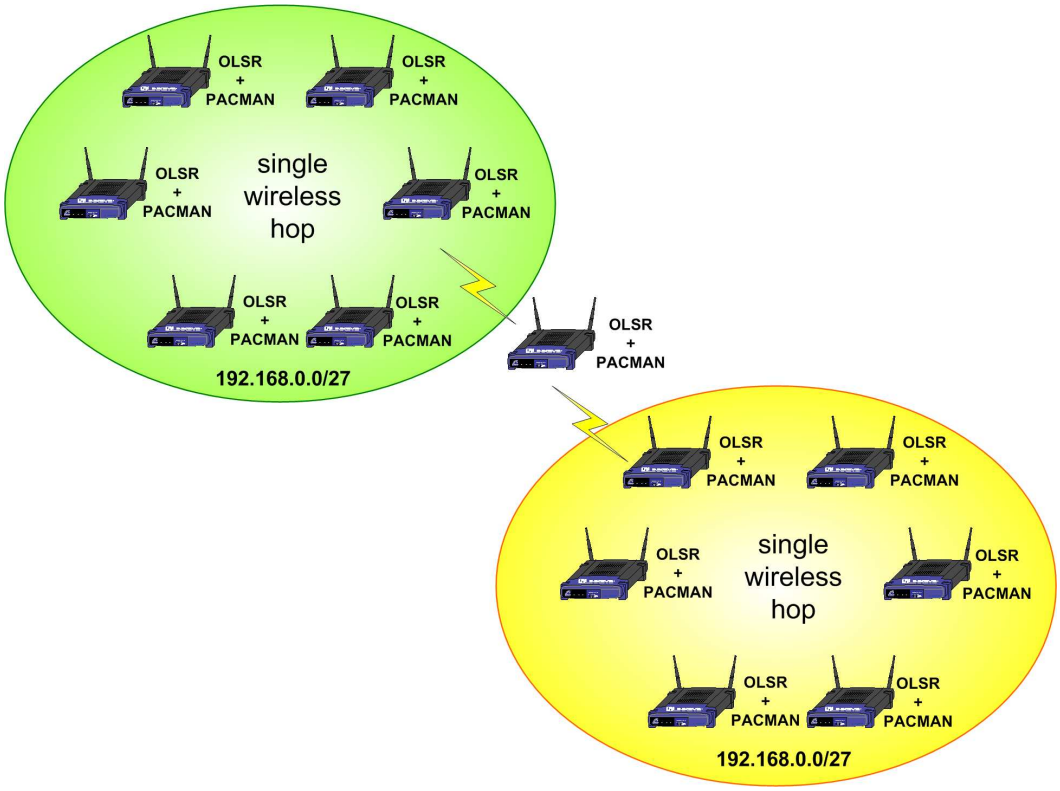

Figure 11. Merging of two networks