Seamless MANET autoconfiguration through enhanced 802.11 beaconing

M. José Villanueva, Carlos T. Calafate[†], Álvaro Torres, José Cano, Juan-Carlos Cano and Pietro Manzoni

Department of Computing Engineering
Universitat Politècnica de València
Camino de Vera S/N, 46022 Valencia, Spain
[†]*Contact phone: +34 963877007*
[†]*Contact fax: +34 963877579*
[†]*Contact e-mail: calafate@disca.upv.es*

Abstract

The deployment of mobile ad-hoc networks involves several configuration steps, which complicate research efforts and hinder user interest. This problem prompts for new approaches offering full autoconfiguration of terminals at the different network layers involved. In this paper we propose a novel solution for the autoconfiguration of IEEE 802.11 based MANETs that relies on SSID parameter embedding. Our solution allows users to join an existing MANET without resorting to any additional technology, and even in the presence of encrypted communications. Experimental testbed results using a real implementation of the proposed solution show that it introduces significant improvements compared to other existing solutions, allowing nearby stations to be configured in about two seconds, and also enabling multi-hop dissemination of configuration data to take place quickly and efficiently.

Keywords: IEEE 802.11; MANET autoconfiguration; SSID; bootstrap problem.

1 Introduction

Mobile Ad Hoc Networks (MANETs) [7] are a networking paradigm where terminals communicate wirelessly and in a multi-hop fashion, not requiring any infrastructure of support. Their characteristics in terms of flexibility and cost have made them the candidate technology for different applications such as rescue and military scenarios [18], information dissemination in vehicular networks [9], multimedia databases [24], or even video communication between peers [11].

However, even after a decade of research efforts, their ease-of-use is still quite low, making it a technology only accessible for experts.

One of the main reasons hindering a large scale deployment of these networks is the initial configuration phase [16]. For a MANET to be fully operational all stations must be configured using compatible layer-2 and layer-3 parameters. In particular, if the IEEE 802.11 technology [6] is adopted for the physical (PHY) and medium access control (MAC) layers, there are some basic parameters that all terminals must share. The common PHY parameters are mainly the modulation type, the frequency, and the synchronization timestamp; notice that these parameters are typically set automatically by the wireless interface without user intervention. Concerning MAC parameters, stations must share: (i) the service set identifier (SSID), (ii) the power-saving mode, (iii) the encryption mode, and (iv) the encryption key. Notice that the power-saving mode and the encryption mode are usually detected automatically, while the other two parameters must be set manually by the user. At layer-3 several parameters must also be set: (i) the IP version used - IPv4 or IPv6 -, (ii) the station's IP address, (iii) the network mask, (iv) the routing protocol used, and (v) the gateway to the Internet. Concerning the latter two parameters, notice that routing protocols are essential in MANETs to make multi-hop communication possible [12, 8]; additionally, these protocols usually offer gateway information either automatically or upon user request.

The aforementioned list of parameters evidence the complexity in configuring MANET stations. Also, since MANETs lack any sort of centralized server to handle configuration, all terminals involved must share this task in a distributed manner. Overall, we consider that there are mainly two barriers preventing distributed node configuration to be effective: on one hand, a wireless link must be established to share all the configuration parameters required to configure the wireless link itself; on the other hand, the fact that wireless communications are easy to intercept typically requires encryption to be adopted, which further complicates the configuration process if the encryption key itself is one of the configuration parameters required. Notice that in both cases we have a variant of the bootstrapping problem, which is typically complex to solve. Up to now this problem has remained mostly untackled by the research community, and no real alternative to manual setup has been found.

In this paper we propose a solution that is able to solve the MANET autoconfiguration problem described above in a very efficient and straightforward manner, setting up all the different parameters associated with the network layers involved in the process (i.e., PHY, MAC and network layers). Our solution assumes that the IEEE 802.11 technology is used, and thus relies on the only unencrypted piece of information that a user can modify at layer-2, the SSID, to accomplish the goals set. Since the SSID is embedded into beacon frames, periodically broadcasted by all MANET participants, high efficiency is achieved with no cost in terms of additional network traffic.

The paper is organized as follows: in the next section we refer to some related works in this research field. Section 3 briefly introduces BlueWi [16], one of the few autoconfiguration approaches available in the literature addressing both layer-2 and layer-3 requirements. An overview of the proposed solution is then presented in section 4. Section 5 offers details about an actual implementation of our proposal on a GNU/Linux platform. In section 6 we offer some performance results obtained in a real-life testbed. A comparison between our solution and BlueWi is then performed in section 7. Finally, in section 8, we present our conclusions along with some guidelines for future work.

2 Related works

In the literature we can find several proposals that focus on the IP address assignment problem in MANETs. Mohsin and Prakash [10] propose a proactive scheme for dynamic allocation of IP addresses in MANETs. Their solution uses the concept of binary split, and takes into consideration issues like network partitioning and merging, as well as abrupt departure of nodes from the system. Weniger [21] presents PACMAN, a novel approach for efficient and distributed address autoconfiguration of mobile ad hoc networks. Special features of PACMAN are the support for frequent network partitioning and merging, and very low protocol overhead. This is accomplished by using cross-layer information derived from ongoing routing protocol traffic, e.g., address conflicts are detected in a passive manner based on anomalies in routing protocol traffic. Sheu et al. [17] propose a scheme to assign IP addresses to newly-joined nodes. In their proposal some nodes are selected as coordinators, which are organized in a tree topology by exchanging *hello* messages. New nodes are able to obtain an IP address by listening to the exchanged *hello* messages and contacting the closest coordinator.

More proposals on this topic are addressed in the survey by Weniger and Zitterbart [22], which illustrates the different approaches for solving the IP address autoconfiguration problem in MANETs, highlighting the major challenges involved.

The main drawback of all the aforementioned proposals is that, for a fully functional MANET to be created, IP address assignment is not the only problem to solve. Thus, a solution offering full configuration of the different network protocols involved, both layer-2 and layer-3, is required.

One of the few works in the literature offering full MANET configuration is the solution proposed by Reyes et al. [16], which relies on Bluetooth to deliver the different configuration parameters required to setup an IEEE 802.11 based MANET. We describe this solution in more detail in the next section, since it will be used for comparison against our own.

In this work we propose a solution offering full MANET configuration that is decentralized, does not require any additional technology besides IEEE 802.11 itself, and does not introduce any extra traffic overhead into the network. Our solution is novel since it addresses both layer-2 and layer-3 configuration (which very few do), while avoiding the limitations of other related works in this field.

3 The BlueWi approach

BlueWi [16] is a solution that relies on Bluetooth [1] wireless interfaces to automate the MANET autoconfiguration process. This solution assumes that all nodes attempting to join the MANET are endowed with both a Wi-Fi and a Bluetooth interface to perform all the required tasks.

Initially, one of the nodes must act as a BlueWi server. This server will register the autoconfiguration service to make it available to all nodes. The rest of Bluetooth devices will function as clients, searching for that service so as to retrieve the MANET configuration parameters, and automatically applying that configuration afterwards. Figure 1 describes this process in more detail.

Every station that wants to join the MANET must first connect to the configuration server via Bluetooth, possibly competing with other stations also waiting to be configured. To do that, stations must perform an *inquiry* action to discover nearby Bluetooth devices. Afterwards they must check the different devices found sequentially until the server offering the desired service (i.e., the *MANET_Autoconf* service) is found. Stations can then establish an L2CAP or RFCOMM connection with the server and download the desired configuration parameters.

The configuration server must make sure it is visible by other devices, and listen to the appropriate L2CAP or RFCOMM port for incoming connections. When a client successfully establishes a connection with the server and requests the configuration data, the server must generate an XML file with all the required information and send it to the client. This XML file will contain all the necessary information for that station to successfully join the MANET. The configuration parameters include the station's IP address and mask, the routing protocol used (e.g. DSR, AODV, OLSR) and all the information required to configure the Wi-Fi interface (SSID, channel, etc.). By allowing the server to determine the IP address of each client we are able to avoid duplicated IP addresses.

Notice that, in the BlueWi solution, the Bluetooth interface is merely used to retrieve the configuration parameters required to join the MANET, while the Wi-Fi interface will allow the station to participate actively in the MANET immediately after the parameters have been received. Thus, after a client station receives its configuration data, it must switch automatically to the Wi-Fi mode. This means that the Bluetooth interface is disconnected and the Wi-Fi interface is activated, allowing to reduce to a minimum the interference between Bluetooth and Wi-Fi technologies.

When the Wi-Fi card is enabled the client station can then proceed to apply the new configuration settings. By doing so it will automatically join the MANET, being able to communicate with other mobile stations that have also configured themselves previously.

Overall, we consider that, despite this solution is able to address both layer-2 and layer-3 configuration requirements, it suffers from some limitations such as requiring all nodes to be endowed with a Bluetooth wireless card, and being centralized, thus suffering from scalability limitations.

4 Overview of the proposed solution

In the field of Wireless Local Area Networks (WLANs), the IEEE 802.11 standard has gained much popularity over the past few years. In fact, its presence is now nearly ubiquitous, although most of the networks are access protected.

The deployment of mobile ad-hoc networks (MANETs) also relies mostly on the IEEE 802.11 standard for the physical and MAC layers. However, differently from WLANs, the lack of access points or any sort of centralized management entity complicates the configuration process for terminals attempting to join the network. In particular, the users must be able to achieve a successful configuration in terms of both layer-2 and layer-3 parameters to enable communication. The characteristics of MANETs - i.e. variable topology, short-lived, decentralized - further complicate the configuration process since the network participants and the different layer-2 parameters may change frequently. Additionally, the support for multi-hop communications requires the same MANET routing protocol to be running on all network nodes.

Due to all the aforementioned issues, the startup of a MANET involves a complex and time-consuming configuration process that may even hinder scalability. Hence, we seek a solution that makes the configuration of MANET stations as simple as possible, so that even those users that are not experts in wireless networking may join and participate in the MANET in a quick, transparent and satisfactory manner.

The envisioned decentralized configuration solution takes into consideration that 802.11 is the technology of choice for most of the MANETs created and that, even when communications are encrypted, beacon frames are not. Thus, our proposal relies on beacon frames as potential carriers of the vital information that allows a station to gain awareness of critical configuration parameters.

By analysing the structure of an IEEE 802.11 beacon frame (see figure 2) we may observe that all the frame fields are automatically set by the 802.11 MAC layer without user intervention, except for the SSID field. This field is set by the user and carries the network's name, having a maximum size of 32 bytes according to the 802.11 standard.

In our proposed autoconfiguration system the SSID will be used not only to include the network's name, but also to inform stations about configuration details which will allow them to be transparently configured. Such duality is not expected to cause any drawback since most SSIDs in use are characterized by a low byte count. To justify this statement we have taken a large database including about 8 million samples corresponding to the top 1000 SSIDs used worldwide [23], and then plotted the cumulative distribution function for this 1000 SSID sizes. The result of this analysis is presented in figure 3. We can see that 92% of the SSIDs in use have a length between 4 and 9 characters, being very large sizes (>16) quite scarce and lacking any additional benefits. Thus, we consider that limiting the SSID to a smaller size would not represent any significant limitation, especially when targeting ad-hoc networks where the SSID must be defined every time a new network is created.

Taking the previous analysis into consideration, we propose a strategy to

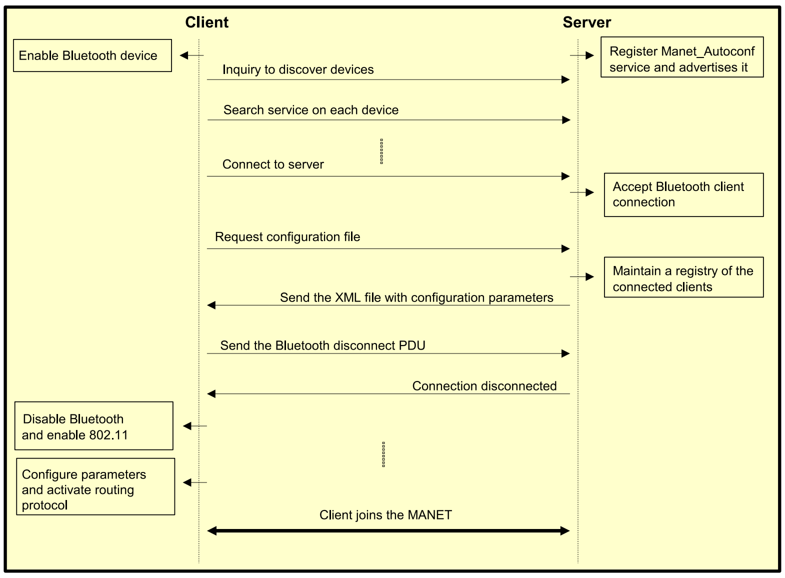
Client
Server
Enable Bluetooth device
Register Manet_Autoconf service and advertises it
Inquiry to discover devices
Search service on each device
Connect to server
Accept Bluetooth client connection
Request configuration file
Maintain a registry of the connected clients
Send the XML file with configuration parameters
Send the Bluetooth disconnect PDU
Connection disconnected
Disable Bluetooth and enable 802.11
Configure parameters and activate routing protocol
Client joins the MANET

Figure 1: BlueWi client/server interaction: message interchange diagram.

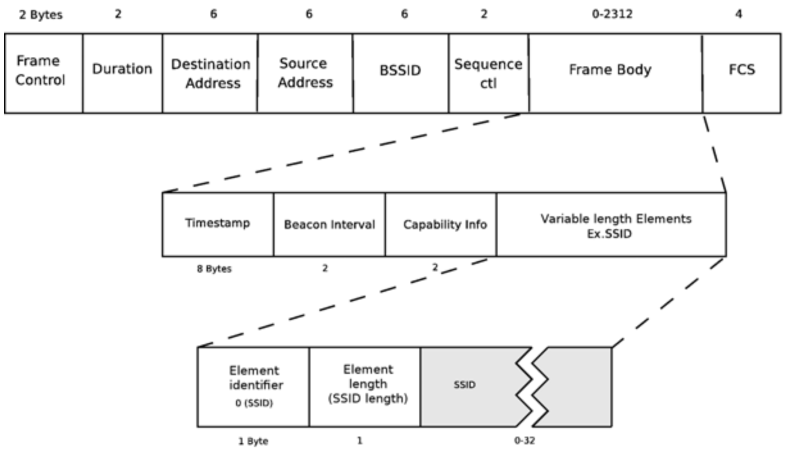

Figure 2: Details of an IEEE 802.11 beacon frame

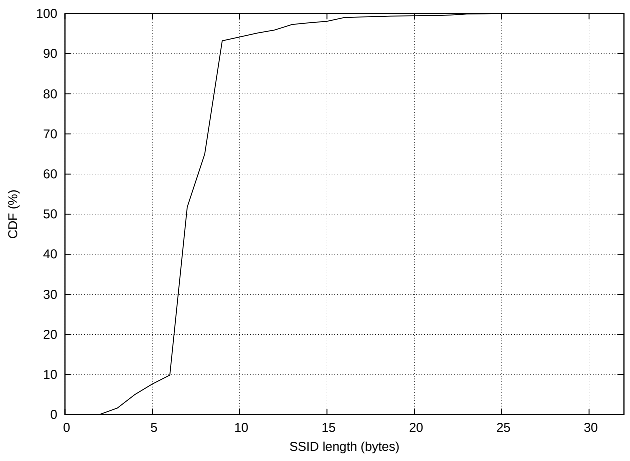

Figure 3: Cumulative distribution function for the SSID set analysed.

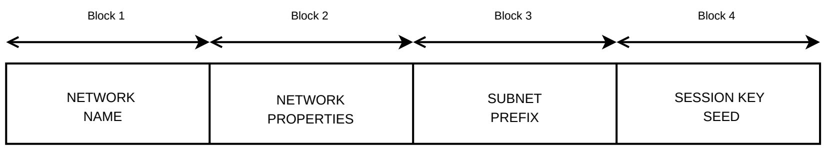

Figure 4: Generic SSID partitioning strategy.

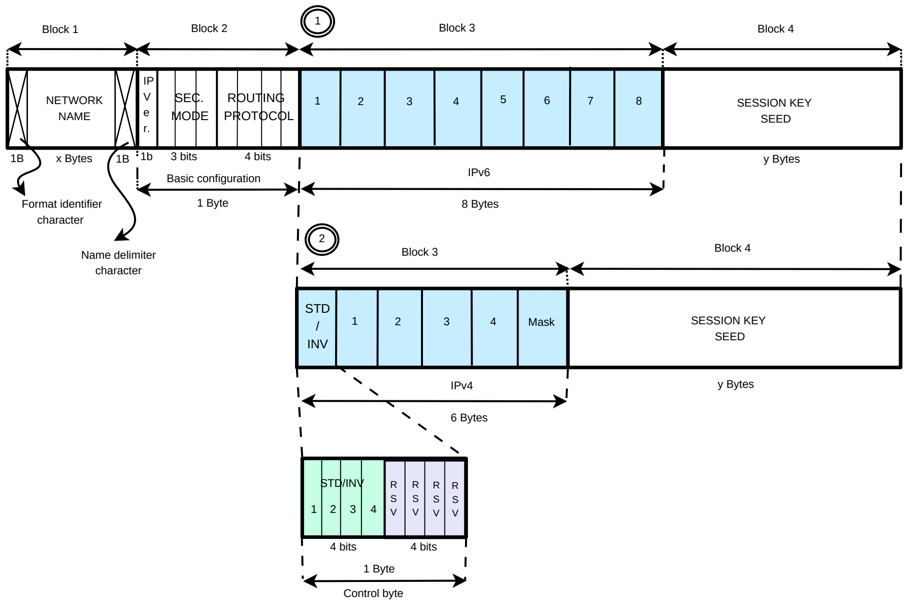

Figure 5: Detailed SSID partitioning strategy.

partition the SSID into four different blocks as shown in figure 4. The name of the network, that is, the legacy SSID, will be in the first block. The second block includes basic network properties: whether it is an IPv4 or IPv6 network, which encryption mode is used, and also which MANET routing protocol should be running. The third block identifies the subnetwork prefix (also including the network mask when IPv4 is used). Finally, the fourth block holds a random value to be used as seed when attempting to derive the session key used for 802.11 MAC encryption.

The proposed solution requires that users attempting to connect to an autoconfigurable MANET parse the SSID to extract all the information required to be connected to the MANET. All data is retrieved automatically through client software, allowing to fully and transparently configure the network connection in terms of both layer 2 and layer 3 parameters.

In terms of advantages offered, our configuration strategy is: (i) scalable, (ii) decentralized, (iii) robust, (iv) efficient, and (v) fully automatic. The arguments that support this statement are the following:

i) Since beacons are periodically generated at a controlled rate by all MANET stations according to a randomization algorithm, the configuration information is made available to the whole MANET independently of its size, which makes the solution scalable.

ii) Any station attempting to join the MANET is able to obtain configuration data just by listening to the beacons from any nearby MANET member, avoiding the need for a central coordinator.

iii) As long as a single MANET member remains active, the proposed configuration strategy remains immune to the loss of participating stations, which makes the process robust to failures.

iv) The proposed strategy does not generate additional network traffic, and allows achieving full node configuration in a short period of time (see section 6), thus offering high efficiency.

v) Since the entire setup process is automatic and transparent, it does not require any technical skills from the user, and even inexpert users are able to take full advantage of MANETs in a seamless manner.

5 Implementation details

In this section we describe how the different parameters required for configuration are embedded into the SSID string and later parsed. We will also offer details of an actual implementation of our approach in a Linux-based testbed. The support for 802.11 wireless cards in the Linux operating system has been available since the late nineties through the Wireless Extensions API [19] developed by Jean Tourrilhes, along with a set of wireless tools [20] accessible through the command line that were developed by the same author.

For our endeavour, we also developed command line applications that allow a user to join an existing MANET with autoconfiguration support, as well as starting such a MANET. The latter option requires the user to define the value of the different parameters required to fully configure a network interface card, which includes both layer-2 and layer-3 setup.

Concerning layer-2 parameters, deploying an IEEE 802.11 based MANET basically requires defining the operation mode (ad-hoc), the SSID, the channel used, and some security details. The latter include the security protocol used (WEP, WPA, or WPA2) and the shared key used for authentication and/or encryption.

At layer-3 we must define which version of the IP protocol is used and, for that IP version, the subnetwork used through a network ID and a network mask. To support multi-hop communication, the routing protocol used (e.g., AODV[12], OLSR [2], DYMO [5]) must also be defined.

5.1 Proposed SSID partitioning strategy

Figure 5 illustrates the proposed SSID partitioning strategy, which has been implemented and validated in a real-life testbed. Notice that, in order to distinguish regular beacons from our formatted beacons, a special (non-printable) character has been inserted just at the beginning of the SSID block, thus allowing to quickly identify SSIDs formatted according to our proposal. The network name, that is, the SSID according to its original definition, appears next, followed by another non-printable character that ends block 1. Block 2 is composed by a single byte where the first bit indicates which IP version is in use (IPv4 or IPv6), followed by 3 bits that indicate which 802.11 security mode is active (0=open access, 1=WEP-64, 2=WEP-128, 3=WPA-PSK, 4=WPA2-PSK, 5-7=reserved for future extensions); concerning the last 4 bits in block 2, they are used to identify the MANET routing protocol used (0=forbidden value, 1=OLSR, 2=AODV, 3=DYMO, 4=DSR, 5-15=reserved for future extensions). Notice that value 0 is forbidden to avoid a situation where this single byte block is set to the NULL value, which would be considered as an *end of string* character by the operating system, thus causing and error.

Depending on which IP version was defined in block 2, block 3 will contain an IPv6 network address field (8 bytes) or an IPv4 network address field. In case IPv6 is used, these 8 bytes represent the first half of the address within the Unique Local Unicast [13] range of addresses (FC00::/7); the latter 8 bytes (Interface ID) are derived from the MAC address of the wireless network interface according to the strategy defined in RFC 4291 [14]. If IPv4 is used instead, we identify the network using 4 bytes plus an extra byte to set the network mask. To avoid those situations where one or more bytes are zero (NULL character), we use the first byte (STD/INV) to invert possible NULL values in any of the four bytes that define the IP address, thus converting any 0x00 value into 0xFF. Stations attempting to configure themselves must reverse the inverted bytes to recover the original values. Notice that this strategy was not required for IPv6 since we rely on the Unique Local Unicast range of addresses, which allows picking any value for the 7 bytes following the first, which means we can easily discard any 0x00 values appearing and pick other values instead.

Concerning the last block, it includes the session key seed, which is used to derive the actual session key that will be used to perform MAC layer encryption.

5.2 Deriving the session key

When a new MANET is generated, the value for the session key seed is picked randomly. This seed allows deriving the session key by supposing that all users are aware of a fixed pre-shared key (PSK). When relying on standard 802.11 this shared key is used directly for MAC layer encryption; however, with our solution, this shared key is replaced by a variable session key. This strategy complicates the discovery of the MAC layer encryption key by a potential attacker by making it different every time a new ad hoc network is created.

The size of the seed itself is variable, and depends on the number of bytes used to identify the network (x). In our solution we will restrict the size of this network identifier to a maximum of 10 characters, which is not considered a restriction since 10 characters are enough to uniquely identify an ad-hoc network in any plausible scenario. Once the network identifier is defined, the size of the session key seed is picked so as to fill up the SSID size, thus reaching the maximum length for the SSID field. Although in theory there are 32 bytes available for the SSID, the fact that the operating systems handle it as a string ending with the NULL character reduces it to 31 bytes. This way the size of the session key seed will be either 20-x or 22-x bytes (10 bytes in the worst case), depending on whether IPv6 or IPv4 is used, respectively.

One of the limitations of having the seed embedded into the SSID has to do with the handling of NULL values, as mentioned above. This means that the number of possible combinations will be slightly reduced by this restriction. Thus, the original space of $256^{(20-x)}$ combinations ($256^{(22-x)}$ for IPv4) is reduced to $255^{(20-x)}$ ($255^{(22-x)}$ for IPv4). In the worst case conditions (if the network name uses all 10 characters) there are still about 10^{24} possible seeds ($\sim 7 \times 10^{28}$ for IPv4); this means that the chances that the same seed repeats for a same group of users becomes negligible.

Figure 6 offers more details about the process of session key generation. Initially, the key shared by all MANET users is combined with the seed made available in the SSID (block 4) by using a hash mechanism such as MD5 [15], SHA-1 [3], or RIPMED-160 [4]. Since the number of bits in the hash may be shorter than the one required by the selected security mode, the hash output is fed back to generate a new hash until the key generator module gathers enough bits. Depending on the security mode selected, the key generator module may have to chop part of the input in order to obtain the correct number of bits for encryption. For example, if MD5 is used for hashing and WEP-64 is used for encrypting, a single hashing round suffices since the 128 bit output is enough to obtain the 40 bits required for a valid session key. On the contrary, if WPA-PSK is used, we need two MD5 hashing rounds to generate a session key of 256 bits, as required.

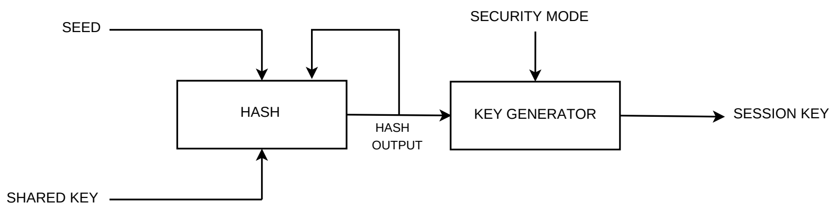

Figure 6: Session key generation process.

Table 1: Average time overhead associated with autoconfiguration tasks.

Autoconguration tasks	***No security***	***WEP (64 & 128 bits)***	***WPA (256 bits)***
Obtain configuration	4 / 20 µs	4 / 20 µs	4 / 20 µs
Generate key	-	23 µs	46 µs
Establish SSID	4,400 µs	4,400 µs	4,400 µs
Apply security key	-	200 µs	17,500 µs
Set IP address	4,000 µs	4,000 µs	4,000 µs
Start routing protocol	5,600 µs	5,600 µs	5,600 µs
Total time	***~14 ms***	***~14.2 ms***	***~31.5 ms***

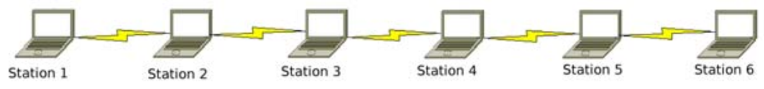

Figure 7: Chain topology used to measure the propagation time for autoconfiguration beacons.

5.3 Methodology of use

In our scheme, we suppose that there is a group of users that regularly creates and joins an ad-hoc network with a specific goal. An example can be a firefighting unit, where on every mission a same group of firemen creates a MANET for communicating among themselves. All users use a same key for accessing the network, referred to as *shared key* in the previous section.

When these users intend to join the same ad-hoc network using our proposed solution, one of them (e.g., the head of the fire squad) creates the ad-hoc network, defining all the parameters required; among them we have the network name, the security mode, the routing protocol used, the IP addressing information and the *seed* used for session key generation. When the terminals used by the other users listen to the beacons generated, they will immediately parse the SSID field to retrieve the configuration details and successfully attach themselves to the MANET. This means that first there will be a layer-2 connection establishment in order to become a member of the Independent Basic Service Set (IBSS) created, followed by IP parameter definitions and the launching of the appropriate MANET routing daemon.

From that point on, any subsequent MANET generated would be quite similar, except that the seed used will always differ, and thus also the session key. This requires a potential attacker to find the session key used, and to launch the attack within the lifetime of a specific ad-hoc network (i.e., for a specific seed). Compared to the default solution, where the session key would be used over and over again, this strategy significantly reduces the effectiveness and interest at performing malicious activities.

6 Validation and performance analysis

In the previous section we described the implementation details of the proposed autoconfiguration system, which includes two components: one that allows creating the ad-hoc network, executed only by the first station, and another that allows joining an existing autoconfigurable MANET. In this section we present some performance results obtained when validating our solution in our ad-hoc network testbed.

Performance measurements were made using five middle-range laptops with similar hardware, running at 1.6 GHz with a single CPU and with 1 GB of RAM. The wireless cards used were Intel embedded devices supporting the IEEE 802.11g standard, and all terminals are within transmission range of the terminal that initially creates the ad-hoc network, unless stated otherwise.

6.1 Assessing the overhead introduced per task

Table 1 shows the average time overhead results obtained for the different security strategies. We do not include WPA2 encryption since it is not yet supported by the Linux OS in ad-hoc mode.

With respect to the first task, the MANET creation component requires parsing the user's input and generating an SSID string with autoconfiguration information (according to the strategy shown in figure 5), while the autoconfiguration component must merely parse the beacon received to extract configuration information. Thus, while the latter task is achieved in just 4 µs, the former (SSID generation) requires 20 µs.

The remaining tasks are similar for both components developed. In particular, the second task is related to key generation, which is achieved according to the strategy shown in figure 6; obviously, this step is skipped if security is disabled. The third and fourth tasks consist in setting the layer-2 parameters, such as the SSID and the encryption key. In case WPA is used, a configuration file must be created before launching the *wpa_supplicant* tool, which is responsible for WPA/WPA2 configuration tasks in Linux. This causes the time associated to that task to account for more than half of the total configuration time.

The last two tasks - IP definition and launching the MANET routing daemon - are related to network layer configuration, being common in all cases.

Overall, the autoconfiguration times can be considered quite low, although we have to take into account that the measurements presented in this section refer to the tasks taking place at the application layer. Since the dissemination of configuration information requires beacons to be received, and autoconfiguration tasks to be completed, prior to start generating new beacons, the total autoconfiguration time is usually higher. In the next section we will focus on these issues.

6.2 Autoconfiguration times in a multi-hop environment

When attempting to autoconfigure different stations in a wireless multi-hop environment, two different issues must be taken into account: (i) there is a delay from the time the first station creates the MANET to the time nearby stations are able to receive the first beacon with autoconfiguration data embedded into the SSID; and (ii) when a new station wants to join the MANET. it must scan the different channels for beacons containing autoconfiguration data. Since in the ad-hoc mode the beacon generation process is distributed and follows a random algorithm, the actual time required to detect the beacons may vary.

To study the multi-hop propagation behaviour of autoconfiguration data, we devised a scenario (see Figure 7) where nodes are arranged according to a chain topology. Distances between nodes are high enough to assure that radio communications are only possible with one-hop neighbours.

In our setting, Station 1 is responsible for starting the MANET. So, at the beginning of our experiment, Station 1 uses the autoconfiguration application to create a new ad-hoc network, while Stations 2 to 6 attempt to connect to the existing network by starting the autoconfiguration application in the *join* mode.

Figure 8 shows how the autoconfiguration information propagates at multiple hops. We can see that Station 2 gets configured in about 2 seconds since it is very close to the station that initiates the ad-hoc network. In particular, most of this time is associated with detecting the beacon, being configuration parameters applied in just a few milliseconds, as shown earlier (see table 1).

As we increase the number of hops, it would be desirable to experience a linear increase of this autoconfiguration time, being such linear increase represented as *best case* in figure 8. However, experimental results show that the average propagation times are associated with a more than proportional increase, which is explained by the random beacon generation process. Remember that the IEEE 802.11 standard establishes that, in the ad-hoc network operation mode, beacons are generated by all stations involved in a distributed fashion by following a randomization algorithm. Thus, in our example, Station 2 would only generate a beacon about half the times, while Station 1 would generate beacons in the other half of the cases. As more stations get involved, the chances that a particular station generates a beacon become smaller, which slows down beacon dissemination. In our scenario, we find that the station at 5 hops from the first one (Station 6) must wait on average 14.8 seconds to detect the first autoconfiguration beacon. As a final remark, we should emphasize that such beacon propagation times are the typical times for multi-hop ad-hoc network environments, being that our solution does not impose a significant additional delay to the process.

In terms of scalability, we consider that our solution is scalable by design since the configuration information data propagates at the beacon propagation rate, which becomes highly effective even in large-sized and highly disperse MANET environments.

Concerning new nodes intending to join the MANET, they can initiate the configuration process as soon as the ad-hoc network is detected (after any periodic beacon is received), usually waiting for only a few seconds on average. In this context, we also measure the delays introduced by routing protocols, that is, the routing topology dissemination time. Notice that, once a station becomes configured and connected, there will be an additional delay introduced by the routing protocol to update the network topology. In our testbed, the routing protocol adopted was OLSR, using the standard parameter values defined in [2]. Thus, we performed a second group of tests where we measured the time elapsed from the instant when the autoconfiguration application completes its tasks, until a valid route to the first node in the chain topology becomes available. Notice that, after the beacon is detected, each station will apply the autoconfiguration parameters, which also includes launching the OLSR protocol daemon.

In the tests that follow, all the previous nodes in the chain are configured and connected from a routing perspective when the new station arrives.

Figure 9 presents our experimental results assuming that Stations 2 to 6 will gradually join the network, creating the topology shown in Figure 7. The values represented in the figure 9 show that routing information dissemination with OLSR imposes a significant time overhead, especially at more than one hop. This is expected since, in the scope of OLSR, communication with one-hop neighbours only requires neighbour detection procedures, while higher hop counts require topology updating procedures to be triggered. Thus, when Station 2 attempts to join the network and contact Station 1, OLSR takes between

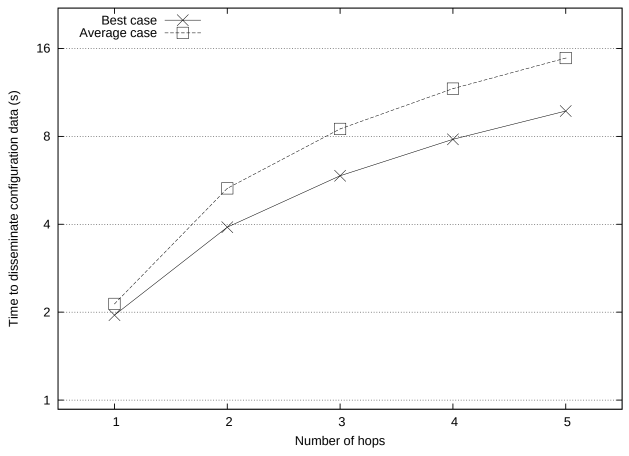

Figure 8: Autoconfiguration times for nodes at multiple hop distances from the initiating node.

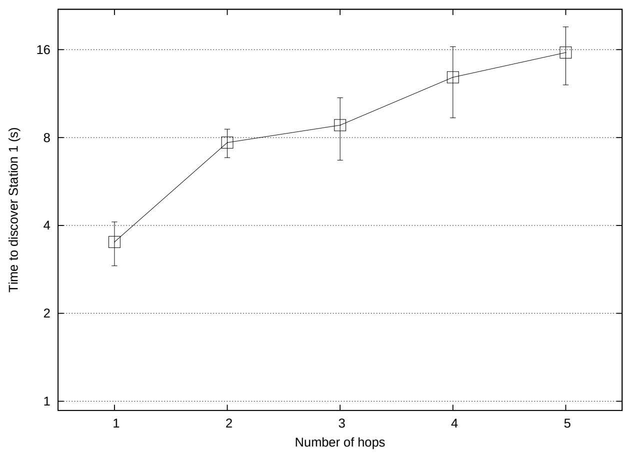

Figure 9: OLSR topology updating time when joining the network at different hop counts from Station 1.

3 and 4 seconds to provide a route to this station, while for Station 6 it will take OLSR between 12 and 19 seconds to provide a valid route.

By combining the results of Figures 8 and 9, we find that the time required for a MANET to be fully connected and operational can be reduced to less than one minute if the proposed autoconfiguration strategy is adopted, even with stations located several hops away from the station starting the MANET, and even when using a routing protocol with a relatively slow responsiveness (e.g. OLSR).

7 Comparison between BlueWi and the proposed solution

In this section we perform a comparison between our solution and the BlueWi solution introduced in section 3.

As referred before, BlueWi requires clients to establish a Bluetooth channel with a BlueWi server to retrieve all the configuration parameters required. The Wi-Fi interface is then configured according to that parameter set. Our proposal significantly differs from BlueWi since it does not assume any sort of server. In fact, any station can start the ad-hoc network, and as long as a single station keeps that network alive, other stations can autoconfigure themselves and join the network.

Figure 10 shows the total autoconfiguration time when attempting to simultaneously configure different numbers of terminals. All terminals are assumed to be at one hop from either the Bluetooth server (BlueWi) or the station that starts the MANET (SSID-based proposal) for comparison. In terms of radio range, the BlueWi solution limits the maximum distance between the Bluetooth server and the stations being configured to 10 meters (default Bluetooth range) or 100 meters, depending on the Bluetooth device class. For our SSID-based solution, stations at one hop from the station starting the MANET are able to detect its beacons for distances up to 250 meters although, as shown in the previous section, multi-hop configuration is possible and does not suppose any impediment.

From figure 10 we can see that the autoconfiguration time for our SSID-based solution is independent of the number of stations involved. This is expected since beacons are broadcasted, being received by all wireless devices within range. Concerning BlueWi, we find that autoconfiguration tasks require several seconds more. This additional time is mostly associated with Bluetooth device discovery procedures (*Inquiry*), which takes about 5.12 seconds to complete, and that are a prerequisite before attempting to contact the Bluetooth server. Also, the number of concurrent stations retrieving configuration parameters will reduce the channel capacity dedicated to each station, thereby increasing the total time involved.

To complete our comparison between BlueWi and our proposal, table 2 summarizes the main differences between both solutions. Overall, we find that the

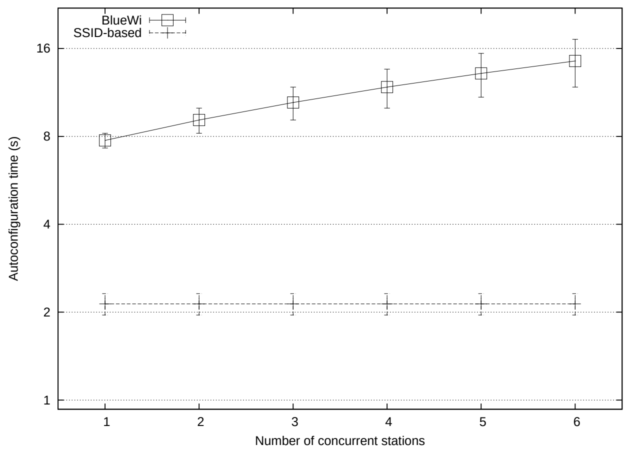

Figure 10: Autoconfiguration time when varying the number of terminals being configured at one hop.

Table 2: Comparison between BlueWi and our SSID-based autoconfiguration technique.

Characteristic	***BlueWi***	***SSID-based***
Configuration strategy	Centralized	Distributed
Autoconfiguration server	Required	Not required
Multi-hop configuration dissemination	Not supported	Supported
Wireless technologies	Wi-Fi, Bluetooth	Wi-Fi
Number of simultaneous users serviced	7	No limit
IPv4 support	Yes (DHCP-like)	Yes (MAC address based)
IPv6 support	No	Yes (MAC address based)
WEP/WPA/WPA2 support	Yes	Yes
Rotating encryption keys	Yes (manually)	Yes (random seed)
Routing protocol	Any	Any
User control and logging	Yes	No
User access control	Bluetooth pin	Pre-shared Key
Best-case configuration time	7.31 s	1.95 s

proposed autoconfiguration system based on SSID parameter embedding offers significant improvements over a pre-existent solution (BlueWi), representing a significant step forward in the state-of-the-art within the field of MANET autoconfiguration.

8 Conclusions and future work

Despite all on-going efforts, the issue of MANET usability is still an important research topic since the complexity when attempting to configure MANET terminals remains high. Besides complexity itself, other issues such as the need to rely on encrypted communications further complicate the configuration problem.

In this paper we propose a novel solution for terminal autoconfiguration that is able to fully configure both layer-2 and layer-3 parameters that are critical to join an 802.11-based MANET. Our solution relies on the SSID field that is present on the periodic beacons generated by IEEE 802.11 compliant stations to announce basic configuration data. By listening to beacons and parsing the SSID field, new stations are able to determine all the information required to successfully join the MANET.

To validate our proposal we developed two software components, one that allows creating a new autoconfigurable MANET, and another one that allows joining an existing autoconfigurable MANET. Experimental results show that both software components are able to perform all the configuration tasks required in a very short period of time. In particular, the total time required is below 15 ms if security is basic (WEP) or disabled, and it is below 32 ms if WPA is used instead.

By deploying a small scenario using a chain topology we showed that multi-hop configuration dissemination can be performed in an efficient manner, introducing on average a delay of about 3.2 seconds per hop. Also, experimental results have showed that, after the configuration process is completed, additional time may be required to allow the chosen routing protocol to update the topology. This is particularly true for proactive routing protocols such as OLSR, which require several seconds to detect new stations and update the network topology.

To complete our analysis, we compared our proposal against BlueWi, a similar solution available in the literature, showing that our strategy offers significant benefits and improvements with respect to the latter.

Overall, we consider that the proposed solution can fill-in the gap between regular users and ad-hoc network technologies, allowing to accelerate the adoption of distributed communication paradigms to a wider range of application scenarios.

As future work we plan to develop a similar set of tools to other operating systems besides GNU/Linux, thus embracing a greater number of potential users.

Acknowledgments

This work was partially supported by the *Ministerio de Educacii¿œn y Ciencia*, Spain, under Grant TIN2011-27543-C03-01.

References

[1] IEEE 802.15.1(tm) IEEE Standard for Information technology– Telecommunications and information exchange between systems– Local and metropolitan area networks–Specific requirements Part 15.1: Wireless Medium Access Control (MAC) and Physical Layer (PHY) Specifications for Wireless Personal Area Networks (WPANs(tm)), 2002.

[2] T. Clausen and P. Jacquet. Optimized link state routing protocol (OLSR). Request for Comments 3626, MANET Working Group, http://www.ietf.org/rfc/rfc3626.txt, October 2003. Work in progress.

[3] D. Eastlake and P. Jones. US Secure Hash Algorithm 1 (SHA1). IETF RFC 3174, September 2001.

[4] H. Dobbertin, A. Bosselaers, and B. Preneel. RIPEMD-160, a strengthened version of RIPEMD. In *Fast Software Encryption, LNCS 1039*, pages 71–82, 1996.

[5] I. Chakeres and C. Perkins. Dynamic MANET On-demand (DYMO) Routing. Internet Draft, MANET Working Group, draft-ietf-manet-dymo-12.txt, February 2008. Work in progress.

[6] IEEE 802.11 WG. International Standard for Information Technology - Telecom. and Information exchange between systems - Local and Metropolitan Area Networks - Specific Requirements - Part 11: Wireless Medium Access Control (MAC) and Physical Layer (PHY) Specifications, ISO/IEC 8802-11:1999(E) IEEE Std. 802.11, 1999.

[7] IETF. MANET Working Group Charter. http://www.ietf.org/html.charters/manet-charter.html.

[8] E. Kulla, M. Hiyama, M. Ikeda, L. Barolli, V. Kolici, and R. Miho. MANET performance for source and destination moving scenarios considering OLSR and AODV protocols. *Mobile Information Systems*, 6:325–339, 2010.

[9] S. Manvi, M. Kakkasageri, and J. Pitt. Multiagent based information dissemination in vehicular ad hoc networks. *Mobile Information Systems*, 5:363–389, 2009.

[10] M. Mohsin and R. Prakash. IP Address Assignment in a Mobile Ad Hoc Network. In *Proceedings of Military Communications Conference (MILCOM 2002)*, volume 2, pages 856–861, Anaheim, California, USA, 2002.

[11] N. Qadri, M. Altaf, M. Fleury, and M. Ghanbari. Robust video communication over an urban VANET. *Mobile Information Systems*, 6:259–280, 2010.

[12] C. E. Perkins, E. M. Belding-Royer, and S. R. Das. Ad hoc on-demand distance vector (AODV) routing. Request for Comments 3561, MANET Working Group, http://www.ietf.org/rfc/rfc3561.txt, July 2003. Work in progress.

[13] R. Hinden and B. Haberman. Unique Local IPv6 Unicast Addresses. IETF RFC 4193, October 2005.

[14] R. Hinden and B. Haberman. Unique Local IPv6 Unicast Addresses. IETF RFC 4193, October 2005.

[15] R. Rivest. The MD5 Message-Digest Algorithm. IETF RFC 1321, April 1992.

[16] J. C. Reyes, E. Burgoa, C. T. Calafate, J.-C. Cano, and P. Manzoni. A MANET autoconfiguration system based on Bluetooth technology. In *3rd International Symposium on Wireless Communication Systems (ISWCS)*, Valencia, Spain, September 2006.

[17] J.-P. Sheu, S.-C. Tu, and L.-H. Chan. A distributed IP address assignment scheme in ad hoc networks. *Int. J. Ad Hoc Ubiquitous Comput.*, 3(1):10–20, 2007.

[18] J.-Z. Sun. Mobile ad hoc networking: an essential technology for pervasive computing. In *International Conferences on Info-tech and Info-net*, volume 3, pages 316–321, 2001.

[19] J. Tourrilhes. Wireless extensions for linux. Hewlett Packard Laboratories, Palo Alto, 1996. Available at the author's home page at http://www.hpl.hp.com/.

[20] J. Tourrilhes. Wireless tools for linux. Hewlett Packard Laboratories, Palo Alto, 2008. Available at: http://www.hpl.hp.com/personal/Jean_Tourrilhes/.

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

[22] K. Weniger and M. Zitterbart. Mobile ad hoc networks - current approaches and future directions. *Network, IEEE*, 18(4):6–11, 2004.

[23] wigle.net. Wireless Geographic Logging Engine. http://www.wigle.net/gps/gps/main/ssidstats, June 2010.

[24] B. Yang and A. R. Hurson. Similarity-based clustering strategy for mobile ad hoc multimedia databases. *Mobile Information Systems*, 1:253–273, December 2005.