INTERNATIONAL JOURNAL OF COMMUNICATION SYSTEMS
Int. J. Commun. Syst. (2009)
Published online in Wiley InterScience (www.interscience.wiley.com). DOI: 10.1002/dac.1092

An efficient mechanism for establishing IP connectivity in next-generation networks

Rui Campos*,† and Manuel Ricardo

INESC Porto, Faculdade de Engenharia, Universidade do Porto, Rua Dr. Roberto Frias, 378 4200-465 Porto, Portugal

SUMMARY

The changes in the communication paradigm envisioned for next-generation networks (NGNs), with peer-to-peer/symmetric attachments gaining momentum and two Internet Protocol (IP) versions coexisting, will pose new challenges to mobile communication networks. Traditional IP auto-configuration mechanisms will not work properly, since they were designed mostly having in mind a client–server/asymmetric attachment model, they assume a single IP version paradigm, and they target the auto-configuration of devices only. The IST Ambient Networks (ANs) project has introduced a new concept—the AN—that enables handling every communication entity, either a single device or an entire network, as an AN. This paper describes a new efficient mechanism, named Basic Connectivity (BC) mechanism, for auto-configuring IP connectivity between attaching ANs. A proof-of-concept prototype, experimental results, and theoretical analysis show that BC suites the future networking paradigm and represents a solution more efficient than the current trial-and-error mechanism for auto-configuring IP connectivity. Copyright © 2009 John Wiley & Sons, Ltd.

Received 31 March 2009; Revised 20 October 2009; Accepted 25 October 2009

KEY WORDS: Ambient Network; IP connectivity; auto-configuration

1. INTRODUCTION

In recent years, we have witnessed the deployment of many communication and networking technologies, from which the wireless field is a prominent example. Heterogeneity in communication and networking implies interworking problems. These problems are now being faced in Internet Protocol networks. Two IP versions are expected to coexist for a long time and multiple IP auto-configuration mechanisms may be in place, bringing up problems either concerning the interconnection of simple devices or entire networks. Additionally, the common thinking that the transition to IPv6 would be based on the dual-stack model and that most devices would become dual-stack,

*Correspondence to: Rui Campos, INESC Porto, Faculdade de Engenharia, Universidade do Porto, Rua Dr. Roberto Frias, 378 4200-465 Porto, Portugal.
†E-mail: rcampos@inescporto.pt

before running out of IPv4 addresses, did not confirm [1]. The fast approaching exhaustion of IPv4 addresses, that could happen in 2010 or 2011 [1], will lead to the deployment of IPv6-only networks. Also, IPv6 has not been the subject of widespread adoption so far; IPv4-only networks are still dominant. Thereby, IPv4-only and IPv6-only networks are envisioned to coexist for a long time. This leads to inefficient network attachments when dual-stack devices are considered. On the other hand, new networking paradigms increasingly assume a symmetric, peer-to-peer relationship between communicating peers, and network components that start having dynamic roles. For instance, a device may act as a simple terminal at a given moment and also as an IP gateway at a subsequent moment, for providing Internet access within a Personal Area Network (PAN) [2, 3]. Legacy attachment procedures are typically asymmetric and obey to a client–server model. Also, the roles of each party and the network services they offer are pre-defined. For example, a terminal (the client) attaches to an infrastructure network (the server) and runs a Dynamic Host Configuration Protocol (DHCP) [4] client to acquire IP configuration parameters from the DHCP server running in the infrastructure. Within the Internet Engineering Task Force (IETF) there have been efforts to standardize protocols able to support both IPv4 and IPv6. Examples are extensions to Mobile IPv6 and Network Mobility (NEMO) [5] and dual-stack DHCP [6]. Yet, none of these solutions considers the efficient establishment of IP connectivity either between devices or networks within the future envisioned communication scenario.

The communication paradigm assumed within the IST Ambient Networks (ANs) project [7] considers both asymmetric and symmetric attachments between ANs, a new concept introduced by the project that enables handling every communicating entity as an AN. In symmetric attachment, peers have similar capabilities and both can request/offer network services; for example, within a PAN every device may be able to run a DHCP server for auto-configuring IP connectivity [2, 3]. In the AN paradigm, attaching devices and/or networks cannot assume any type of network configuration service to be deployed by its peer, since there are multiple possibilities. Even when there is a clear definition of roles, such as in the case of a terminal attaching to an infrastructure network, the coexistence of two IP versions and multiple auto-configuration mechanisms may render IP connectivity auto-configuration difficult and inefficient. Using state-of-the-art solutions, terminals try every possible local auto-configuration mechanism until a mechanism hopefully succeeds. This is not the most efficient solution, namely when the frequency of attachments a network device may perform is high, such as in scenarios where mobility and dynamics are present.

Concerning the attachment between networks, network layer heterogeneity brings up further problems. For instance, when networks with incompatible address spaces attach, some mechanism needs to be provided for internetworking. Moreover, plug and play attachment between the networks is expected to be supported, in a new paradigm where user configuration efforts should be avoided. Nowadays, there is not a generic solution solving these problems. A mechanism was defined by the IETF to enable internetworking between IPv4 and IPv6 networks, named Network Address Translation—Protocol Translation (NAT-PT) [8]. Nonetheless, it suffers from the same problems as the IPv4 NAT [9], namely it limits end-to-end connectivity and, in its simplest form, it only provides one-way connectivity between an IPv6 network and a peer IPv4 network. On the other hand, the usage of private IP address spaces may cause that two attaching ANs run on overlapping address spaces. Currently, there is no standard solution addressing this problem. In the IETF MANET AUTOCONF, the problem has been raised regarding the merging of mobile adhoc networks (MANETs) [10] and draft solutions have been proposed [11]. They are based on defining MANET auto-configuration mechanisms that (1) assign statistically unique addresses to each MANET device in order to avoid duplicate addresses when network merging occurs; (2) provide means for duplicate address detection (DAD) on a per-device basis, so that address conflicts are detected and solved when network merging occurs. Still, these proposals target MANET scenarios in particular. The assumption of assigning statistically unique addresses is not applicable in general.

These problems represent a motivation for defining a new mechanism coping with new communication paradigms. We propose the basic connectivity (BC) mechanism that enables plug and play and efficient IP connectivity establishment between attaching peers, either devices or networks. The BC mechanism does not assume any particular migration path from IPv4 to IPv6. Regarding the attachment of networks it assumes dual-stack border nodes (BNs), since that is the only means to enable internetworking between networks supporting different IP versions. When it comes to the auto-configuration of terminals, the BC mechanism simply takes into account the current reality: almost every operating system (OS) running in IP terminals is dual-stack (the current iPhone OS is the exception). Our proof-of-concept prototype demonstrates that the BC mechanism suites the envisioned networking scenarios. Additionally, experimental results and theoretical analysis show that the BC mechanism represents a solution more efficient than the current IP auto-configuration trial-and-error mechanism.

The remainder of the paper is organized as follows. Section 2 presents the most common auto-configuration mechanisms used in IP networks. Section 3 explains the current trial-and-error auto-configuration mechanism used when two IP versions and multiple IP auto-configuration mechanisms coexist, and Section 4 mentions the related work. Section 5 describes the BC mechanism, Section 6 presents the BC mechanism proof-of-concept prototype, and Section 7 provides the evaluation of the mechanism through theoretical analysis and experimental results. Section 8 draws the conclusions.

2. IP AUTO-CONFIGURATION MECHANISMS

This section describes the most common IP auto-configuration mechanisms defined for both IPv4 and IPv6.

Dynamic configuration of IPv4 link-local addresses: The dynamic auto-configuration of IPv4 link-local addresses was deployed by the IETF Zeroconf Work Group [12] and is now available as an IETF RFC [13]. Multiple OSs, such as Windows XP and Linux, already implement it as an alternative solution to DHCP. This solution defines how a host can automatically configure an IPv4 address within the 169.254/16 prefix being valid for communication with other devices in the same link. First, the host generates a random IP address in the 169.254/16 range. Next, it performs DAD by sending out multiple (usually three) gratuitous address resolution protocol (*ARP_REQUEST*) messages [14], in order to assess if the address is already in use; if a reply is received, the address is being used by other terminal and a new address must be tried. Finally, the host assigns the IP address to the local network interface, and link-local connectivity becomes possible.

IPv6 stateless address auto-configuration: This solution, specified in [15], defines the steps carried out by a host to auto-configure its network interfaces in IPv6, without using a centralized service. The auto-configuration process comprises the generation of a link local and a global address, and a DAD procedure, performed using Neighbour Solicitation (NS) messages of the Neighbour Discovery Protocol (NDP) [16], in order to verify the uniqueness of the tentative address; if a reply is received—a Neighbour Advertisement (NA) message—the tentative address must be considered to be in use by other device and it cannot be configured. Before sending out the NS message, the host shall join the solicited-node multicast address of the tentative address, computed as a function of the tentative address, so that it is able to detect the presence of a peer using the address [15]. This is performed by sending out multiple Multicast Listener Discovery (MLD) Report messages [17]. The auto-configuration of a link-local address is carried out upon network interface activation, and after combining the well-known prefix FE80::0/10 with the interface identifier (ID), based on the MAC address. The configuration of a global address is accomplished by combining the prefix announced by a local router, using the Router Advertisement (RA) messages defined in [16], with the interface ID. Still, this is only accomplished if the M flag included in the RA message is inactive; otherwise, the host is intended to use DHCPv6 (see below). The RA message also includes a so-called O flag, which informs the host whether it should use DHCPv6 to obtain optional information, when the IP address auto-configuration is performed based on the prefix announced in the RA message.

Dynamic host configuration protocol: The DHCP [4] provides a framework for passing configuration information to hosts, using a client/server model. It is based on the exchange of four signalling messages. The client broadcasts a *DHCP_DISCOVER* message in order to discover available servers; it may receive one or more *DHCP_OFFER* messages. Based on the configuration parameters included in these messages, the DHCP client selects one of the DHCP servers. Afterwards, it broadcasts a *DHCP_REQUEST* containing the identification of the selected server. Finally, the server will reply with a *DHCP_ACK* message containing the assigned address and optional information. Before configuring the assigned IPv4 address, hosts have to run the DAD procedure. With the advent of IPv6, a new DHCP version (DHCPv6) [18] came up, considering a different operation model: (1) a well-known multicast address is used by clients to address all the servers in the link, instead of broadcasts; (2) unlike DHCPv4, which is used to perform the whole host configuration, DHCPv6 can be used to just complement the stateless mechanism; (3) the messages defined for DHCPv6 are different in name and format. In real implementations, DHCPv4 and DHCPv6 are used independently for dual-stack hosts. A possible solution to integrate the two mechanisms is suggested in [6]. Still, it only brings up more efficiency when the two IP versions are simultaneously available. As mentioned above, this is not the typical case expected for next-generation networks (NGNs).

3. TRIAL-AND-ERROR MECHANISM

Currently, a dual-stack device runs both IPv4 and IPv6 auto-configuration mechanisms, whether two IP versions are supported by its communicating peer or not. A trial-and-error mechanism is used. In what follows, we describe the IPv4 and IPv6 auto-configuration mechanisms commonly used today, based on the auto-configuration solutions mentioned in Section 2.

The flowchart of Figure 1 shows the current auto-configuration procedure for IPv4, when a device is attaching to a given peer device or network. The device starts the auto-configuration process by sending a *DHCP_DISCOVER* message. If one or more *DHCP_OFFER* messages are received, the device sends a *DHCP_REQUEST* to the selected DHCP server, as described in Section 2, and awaits for the corresponding *DHCP_ACK* message sent by the server; if no *DHCP_ACK* is received before a timeout, a new *DHCP_REQUEST* is sent, if the maximum number of *DHCP_REQUEST* messages has not yet been reached. Otherwise, it keeps sending *DHCP_DISCOVER* messages until *DHCP_OFFER* messages are received or the maximum number of retransmissions for *DHCP_DISCOVER* is reached. In this case, the Dynamic Configuration

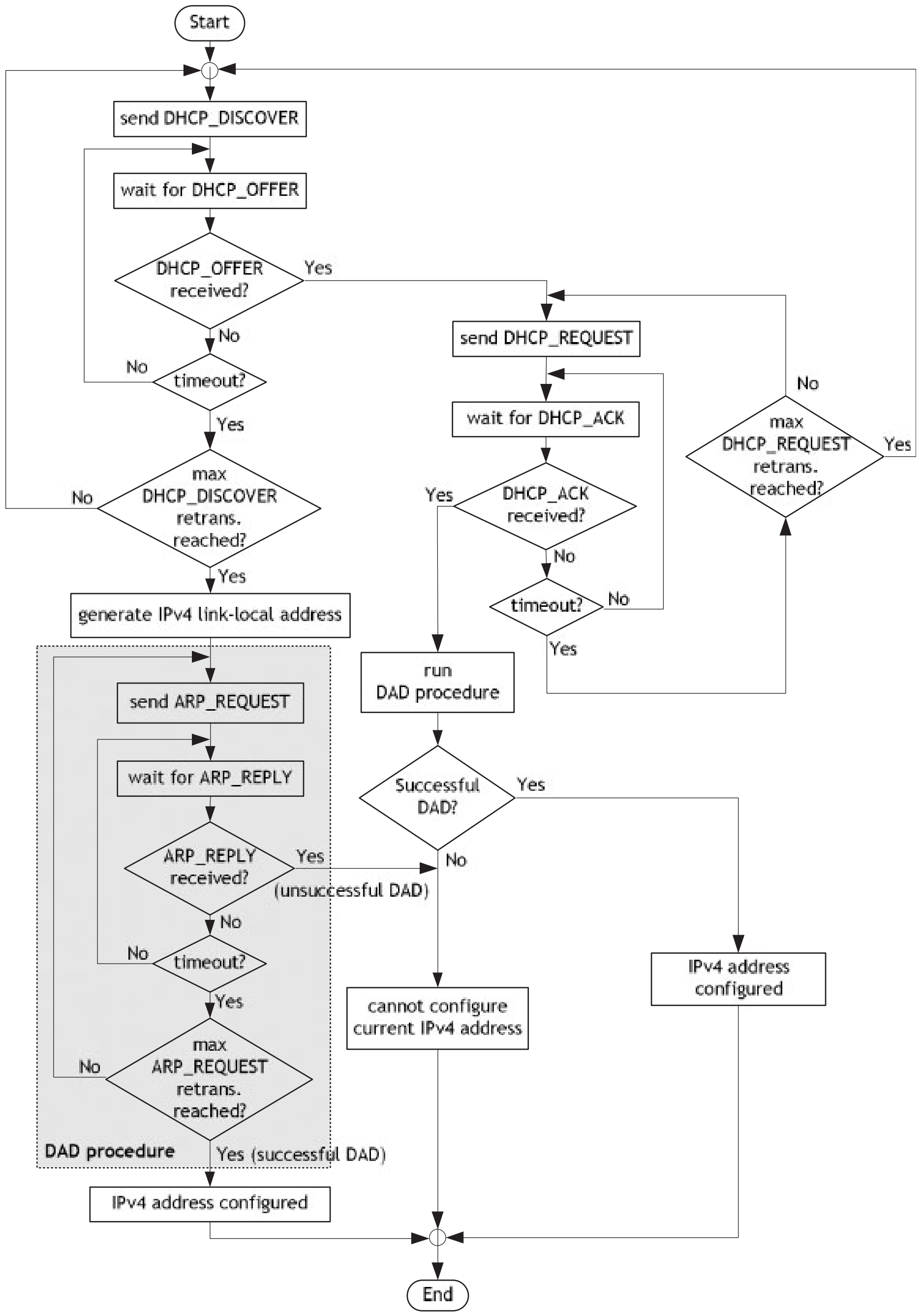

Figure 1. Current IPv4 auto-configuration mechanism.

of IPv4 Link-Local Addresses procedure is invoked. In both cases, DAD is performed in order to check the uniqueness of the IPv4 address to be configured. The DAD procedure consists of the following steps. The node testing the uniqueness of IP addresses either locally generated or assigned by the DHCP server, sends out an *ARP_REQUEST* message addressed to that IP address. If an *ARP_REPLY* is received before a timeout, the DAD procedure ends with the result 'Unsuccessful DAD'. Otherwise, after the timeout a new *ARP_REQUEST* message is sent if the maximum number of *ARP_REQUEST* messages (usually 3) was not yet reached. If the maximum number is reached, the DAD procure ends with the result 'Successful DAD'. When no duplicate address was detected, the IP address is assigned to the corresponding network interface.

The flowcharts of Figure 2 illustrate the IPv6 auto-configuration mechanism. The auto-configuration of a link-local address is always performed first (Figure 2(a)); in Section 2, we have described how the IPv6 link-local address is generated. Before the address is assigned, a DAD procedure is also run, which is similar to the DAD procedure described for IPv4; the difference has to do with the use of the NDP protocol. If the DAD procedure ends with the result 'Successful DAD', the IPv6 link-local address is assigned to the local network interface; else, no IPv6 link-local address is configured. Afterwards, the device starts the auto-configuration of a global IPv6 address (Figure 2(b)). As a first step, it sends an *RS* message and awaits for the corresponding *RA* message. If an *RA* is received, the flag *M* included in this message (see Section 2) is checked, so that the device knows the means available to configure the global IPv6 address: (1) using the RA message sent out by the on-link IPv6 router; (2) using DHCPv6. If flag *M* is equal to 0, the first option is used; otherwise, DHCPv6 shall be used. If an *RA* message is not received after a given timeout, a new *RS* is issued; this process is repeated until the maximum number of *RS* messages is reached. In that case, DHCPv6 is tried. The procedure using DHCPv6 is similar to that using DHCPv4, as it can be verified in the flowchart. However, in this case, if the DAD procedure ends with the result 'Unsuccessful DAD' while using DHCPv6, the DHCPv6 client will issue a *DHCP_DECLINE* message, so that the server is informed that the client could not configure the assigned address.

The auto-configuration mechanisms illustrated in Figures 1 and 2 are run in parallel by a dual-stack device. Therefore, if, for instance, the device is attaching to an IPv4-only access network, supporting DHCP, a set of signalling messages will be wasted in the auto-configuration process. On the other hand, if the device is attaching to a peer that only supports the Dynamic Configuration of IPv4 Link-Local Addresses auto-configuration mechanism, both time and signalling messages will be wasted; similar reasoning could be elaborated if an IPv6-only peer was considered. The current IP auto-configuration approach considers a dual-stack model and the need to have IPv4 and IPv6 connectivity established simultaneously. Within the current and the future IP communication paradigm this may not be required, due to the deployment of IPv4-only and/or IPv6-only networks. Additionally, in the current approach, it is assumed that in general a device will attach to a network infrastructure running a DHCPv4/v6 server or IPv6 stateless auto-configuration. This is not aligned with the new envisioned peer-to-peer, symmetric communication paradigm, where both parts can run a DHCP server, for instance.

4. RELATED WORK

The trial-and-error mechanism described in Section 3 is not the most efficient solution for doing IP auto-configuration in a context where IPv4 and IPv6 may coexist. On the other hand,

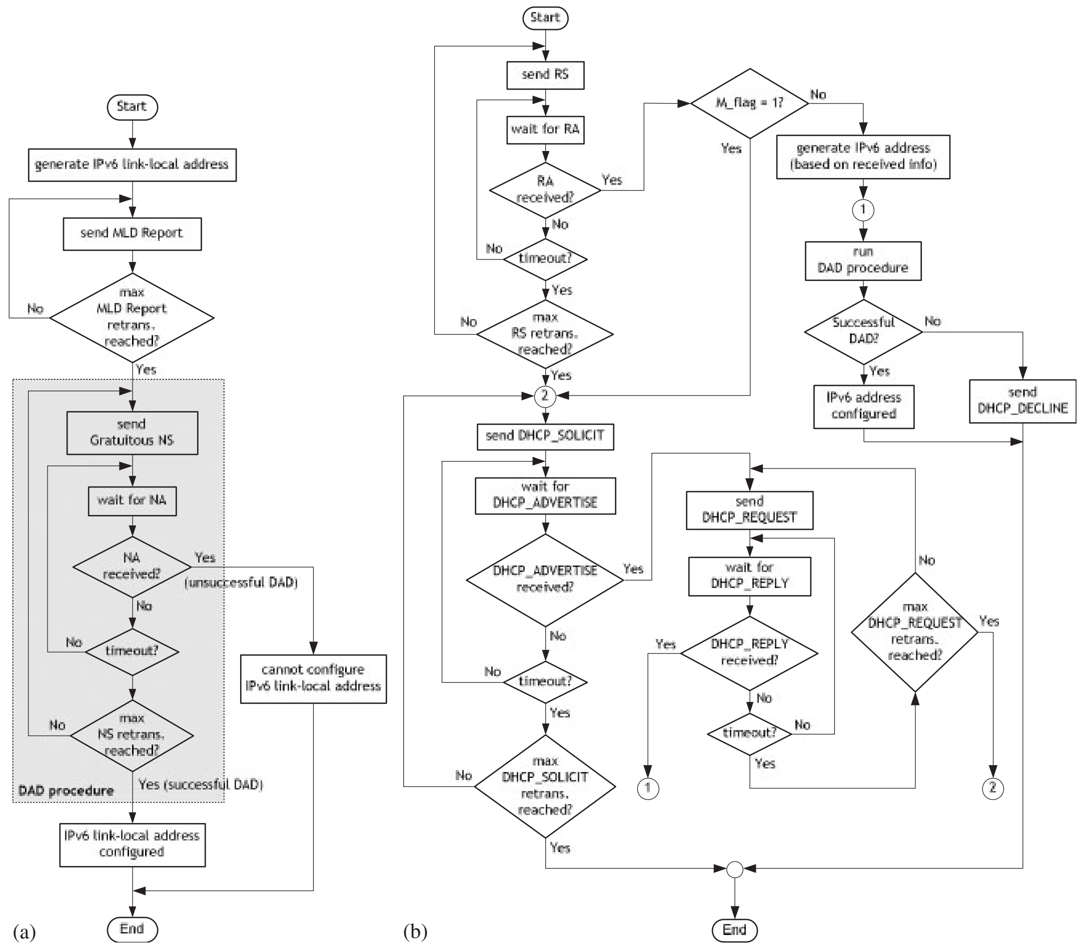

Figure 2. Current IPv6 auto-configuration mechanism: (a) link-local address auto-configuration; (b) global address auto-configuration.

auto-configuration for attaching networks is currently incipient. Auto-configuration is essentially possible when it comes to the routing issue, using the well-known Border Gateway Protocol [19], but does not address the issues we have mentioned in Section 1, for instance, the attachment between adjacent IP networks running on distinct addressing schemes. In this section, we present the solutions that have been proposed so far to deal with these aspects.

The 6 to 4 mechanism [20] was proposed to enable internetworking between IPv6 realms that can transparently communicate over existing IPv4 networks. Still, this solution does not solve the specific problem of adjacent attaching networks, it does not deal with the auto-configuration of IP connectivity between such networks, and it addresses a specific scenario. In addition, it is unable to automatically adapt to different networking contexts, as it is required in NGNs. For instance, if two IPv6 networks previously connected over IPv4 become adjacent (e.g. because one of them moved), the 6to4 mechanism is unable to identify that an IPv6-only setup is possible to enable IP connectivity between the two adjacent networks.

The Ad-Hoc Configuration Protocol (AHCP) is a recent protocol proposed to address IP auto-configuration in multi-hop mesh networks. It aims at replacing the classical auto-configuration protocols, such as DHCPv4 and DHCPv6, in networks where they are difficult or impossible to deploy. Yet, this is just another IP auto-configuration, that has the same advantages as the dual-stack DHCP solution referred in Section 2. Also, like the latter, it only leads to greater efficiency when the two IP versions are simultaneously available, which is not the common scenario envisioned for NGNs (see Section 1). In addition, AHCP does not cope with the auto-configuration between attaching networks.

MANET auto-configuration is a hot research topic in the IETF and several solutions have been proposed in the last few years [11]. However, these are yet other auto-configuration solutions targeting the MANET-specific scenario. On the other hand, the IPv4 and IPv6 coexistence is not properly addressed; For example, no solution is defined to address the scenario of two attaching MANETs running on different IP versions. On the other hand, the auto-configuration solutions defined so far are also inefficient when two IP versions coexist.

In the past, the authors proposed a solution for addressing the efficient auto-configuration of dual-stack terminals, called Meta Auto-configuration Framework (MAF) [21]. Still, MAF does not address the auto-configuration of networks, namely the auto-configuration issues arising when two adjacent networks attach. Moreover, it is just a preliminary auto-configuration solution for NGNs without any practical evaluation. The auto-configuration solution for NGNs proposed herein—the BC mechanism—represents an evolution from MAF, where both terminals and networks come into play. The BC mechanism is described in the next section.

5. BC MECHANISM

The BC mechanism proposed here deals with the problems mentioned in Section 3 and addresses peer-to-peer, symmetric attachments between devices or networks. The BC Manager (BCM) is the central entity of the BC mechanism. It manages the establishment of IP connectivity between attaching peers. BCM communicates with peer BCMs for negotiating the proper IP version, auto-configuration mechanism, and addressing scheme (when establishing connectivity between networks). Furthermore, it interacts with peer BCMs to coordinate the configuration of the local services and resources, network layer, and local auto-configuration mechanisms accordingly. The BC mechanism does not implement any auto-configuration mechanism by itself. Rather, it selects the proper local auto-configuration mechanism, such as DHCP [4] or IPv6 Stateless Address Auto-configuration [15], and relies on it for establishing IP connectivity. The flexibility of the BC mechanism allows easy integration of new auto-configuration mechanisms, e.g. MANET auto-configuration mechanism, and permits the selection of the proper auto-configuration according to the different contexts of a network node. For instance, if a specific node is connecting to a MANET, it shall use a MANET auto-configuration mechanism; conversely, if it is connecting to a node deploying the Dynamic Configuration of IPv4 Link-local Addresses mechanism [13], it has to use this mechanism instead.

In the attachment between nodes, peer BCMs negotiate only the IP version and auto-configuration mechanism to be used for establishing IP connectivity, and select the agreed IP version and auto-configuration mechanism accordingly. Concerning the attachment of networks, the AN project assumes the existence of a BN [22] located at the border of the network, representing it to the outside world. Also, the project defines a new internetworking layer, named Node ID (NID) layer [22], sitting between the network layer (IP) and the transport layer, which is the lowest common communication layer within the AN framework. When dealing with the attachment of networks, besides establishing connectivity between the BNs of the networks, BCMs negotiate how to interconnect the address spaces of the networks. The announcement of the internal address space depends on internal policies and leads to two scenarios. When internal policies disable revealing the internal addressing, BCM does not announce its internal address space to the peer network, and configures the local BN as a NID router, hiding the address space behind it. On the other hand, if internal policies enable revealing the address space, but the networks support incompatible address spaces, BCMs agree on the configuration of, at least, one NID router in one of the BNs. If networks have compatible address spaces, the BCMs configure one or two IP (IPv4 or IPv6) routers between the networks. By default, the BC mechanism assumes the presence of the NID layer as the common internetworking layer. Nevertheless, conceptually it works with other internetworking solutions, such as NAT-PT and the set of solutions recently proposed to replace NAT-PT [23].

Figure 3 shows the architecture of the BC mechanism when running within a given node. The BC Protocol (BCP) is used to exchange control information between BCMs. Within the AN framework, the BCP information is transported by the AN Attachment Protocol (ANAP) [24]. ANAP copes with the establishment of a security association between attaching peers and consists of a four-way message exchange. When running between adjacent ANs, i.e. ANs sharing the same wired or wireless technology (e.g. WLAN, Ethernet), ANAP runs over Layer 2 protocols [25]. The information elements included in the BCP messages are piggybacked over these four ANAP messages. BCP considers three messages: NEGOTIATE, AGREEMENT, and DONE. Each message starts with a common header (1 byte), which includes the type of the message; the other information elements are specific to each message. The BCP messages are shown below using the ABNF syntax [26].

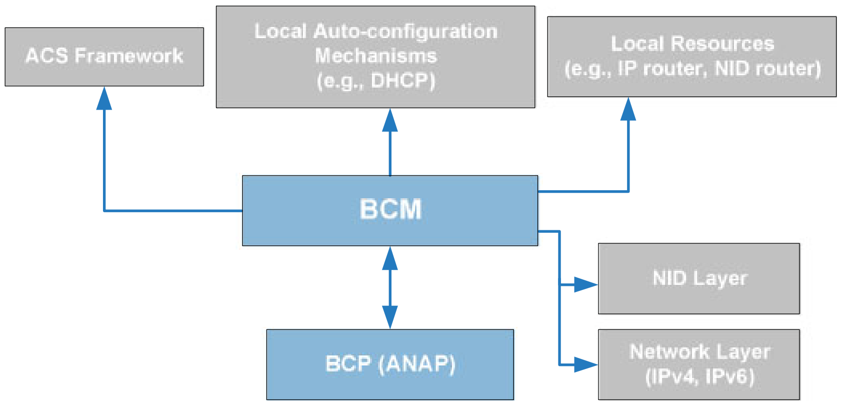
ACS Framework
Local Auto-configuration Mechanisms (e.g., DHCP)
Local Resources (e.g., IP router, NID router)
BCM
NID Layer
BCP (ANAP)
Network Layer (IPv4, IPv6)

Figure 3. BC mechanism architecture within a given network node running it.

The NEGOTIATE message is defined as follows:

```
NEGOTIATE  = message-type
             node-type
             network-layer-versions-supported
             autoconf-mechanisms-supported
             [address-space-type]
             [netmask-length]
             [subnet-address]
```

where node-type (1 byte) defines the type of device sending the message, a stand-alone node or a BN of an AN, and network-layer-versions-supported (1 byte) and autoconf-mechanisms-supported (1 byte) specifies the network layer versions (e.g. IPv4), and auto-configuration mechanisms supported by the current node, respectively. The address-space-type (1 byte), e.g. IPv4 address space, the netmask-length (1 byte), and subnet-address (variable length), e.g. 192.168.10.0, are optional information elements that refer to attachments performed between networks (ANs). Thus, the size of the NEGOTIATE message depends on the type of node running the BC mechanism. If the mechanism is run by a stand-alone node, the size of the message is equal to 4 bytes. Conversely, if the mechanism is run by a BN, the optional information elements are in place and the size is greater. The actual size of the message depends on the type of address space. For instance, for an IPv4 address space the size of the message is 10 bytes, whereas for an IPv6 address space it is 22 bytes.

The AGREEMENT message includes the following information elements:

```
AGREEMENT  = message-type
             network-layer-version
             autoconf-mechanism
             local-remote-autoconf-server
```

where network-layer-version (1 byte) is the network layer version agreed to be used for establishing connectivity between the communicating peers, autoconf-mechanism (1 byte) specifies the auto-configuration mechanism to be used for address configuration, and local-remote-autoconf-server (1 byte) defines which peer deploys the server part of the auto-configuration mechanism, if applicable (e.g. DHCP). The size of the message is 4 bytes.

Finally, the DONE message includes the message-type field only, since no further information needs to be transferred between the peers. This message is used for terminating the process.

The execution of the BC mechanism is illustrated in Figure 4 considering a symmetric scenario of two attaching terminals. One of the terminals is dual-stack and the other is IPv4-only. Only the right IP auto-configuration mechanism (the Dynamic Configuration of IPv4 Link-Local Addresses [13]) is run over the local link. The size (in bytes) of the ANAP messages and the BCP information elements are shown between parentheses.

In its current version, the BC mechanism targets single-homed terminals and networks. However, it can also deal with multi-homed terminals/multi-homed networks, if a BCM is run per network interface. In that case, the BCM for each interface is in charge of negotiating with the peer BCMs the proper auto-configuration mechanism to be used. Still, this is not the most efficient approach. An approach where a single BCM deals with the whole set of network interfaces may make the negotiation process more efficient. For example, if two attaching multi-homed devices have more than one communication technology in common, a single communication link may be used to

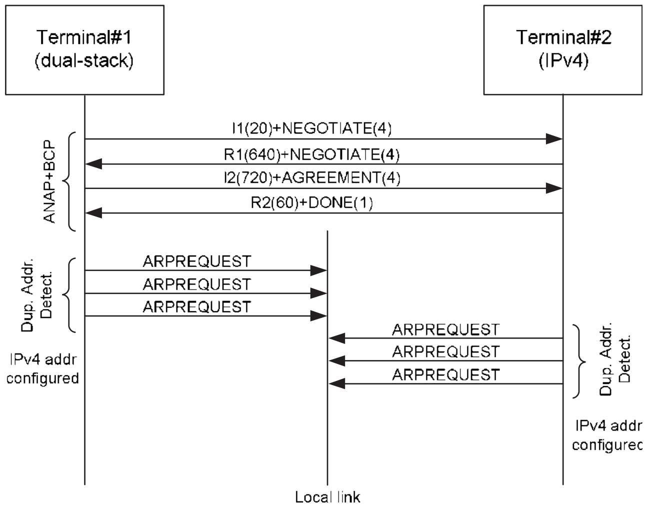

Figure 4. Message sequence chart illustrating the execution of the BC mechanism when two terminals attach.

negotiate the IP auto-configuration mechanism(s) to be used over each communication link. This approach will be considered in the next version of the mechanism.

6. BC MECHANISM PROTOTYPE

We have implemented the BC mechanism under FreeBSD OS, using C++. Figure 5 shows the class diagram of the BC mechanism prototype. Below, we provide a brief description of each class:

- **CBcm** is the core class. It implements the procedures assigned to the BCM in the overall BC mechanism and interacts with the other classes of Figure 5.
- **CBcp** implements the BCP protocol. When CBcm needs to send signalling information to a peer BCM, it passes the corresponding information elements to CBcp, which is responsible for creating the proper BCP message and to send it towards the peer. On the other hand, CBcp is in charge of (1) handling incoming BCP messages to retrieve information elements transported on it, and (2) passing these elements to CBcm.
- **CLocalconf** deals with the interaction with local resources and auto-configuration mechanisms. After the negotiation phase, CBcm interacts with this class in order to perform the required configurations, such as 'start DHCP server' and 'configure IPv4 router'.
- **CBcmdb** reads a local configuration file specifying the characteristics of the current node and the characteristics of the network to which it belongs, when the node is a BN. Information present in this configuration file includes type of node (single node or BN), local IP versions and auto-configuration mechanisms supported and, if applicable, the address space being used within the network to which the BN belongs to.

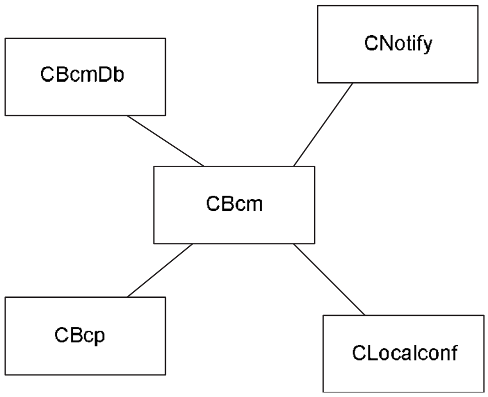

Figure 5. BC mechanism architecture.

- **CNotify** copes with notifications towards the Ambient Control Space [24]. Different notification mechanisms are used depending on the entity being notified; CNotify deals with this in order to render the process fully transparent from CBcm's standpoint.

The BC prototype makes part of the overall AN prototype [27], and it was demonstrated at the final audit of the project. The deployment of the BC mechanism allowed us to prove that (1) it does work in practice, both in stand-alone mode and in conjunction with the other software modules developed in the context of the project, and (2) that its implementation and integration with legacy IP auto-configuration solutions is a straightforward process. Although the main goal of the prototype was to be used as a proof-of-concept, it was also useful to get some experimental results, as we will see in Section 7.

7. EVALUATION

This section is devoted to the evaluation of the BC mechanism. Our purpose is to evaluate the benefits of considering the BC mechanism as part of the attachment procedure defined by the ANAP protocol within the AN framework [24]. We name the trial-and-error mechanism used to establish IP connectivity, when multiple IP versions are present, as legacy mechanism. In order to compare the two mechanisms, the ANAP protocol is used in both mechanisms, for establishing a security association between attaching peers (ANs), as defined in [24]. The comparison is performed based on the overhead, auto-configuration delay, and energy consumption. The evaluation of BC from the functional point of view was carried out using the BC prototype.

7.1. Scenarios

The scenarios shown in Figure 6 were considered for evaluating the BC mechanism from a functional point of view. Scenario 1 consists of the attachment of two nodes connected to the same link. As shown in Figure 6(a), in this scenario, both nodes support a DHCP client and a DHCP server. Using the legacy mechanism, either IP connectivity was not established or it was established but two IP networks were created over the link, which is clearly unnecessary. In the

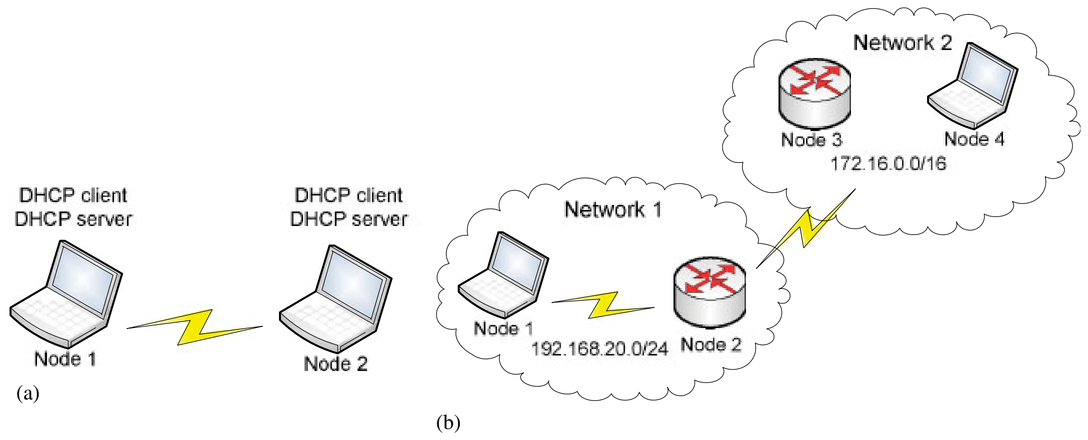

Figure 6. Scenarios used for evaluating the BC mechanism from the functional point of view: (a) Scenario 1 and (b) Scenario 2.

first case, the DHCP clients running in each node accepted the *DHCP_OFFER* sent by the DHCP server running in the same node and no IP connectivity was configured at all. In the second case, each DHCP client accepted the *DHCP_OFFER* of the server running in the peer node, and two IP networks were established over the same link; this problem would also be observed for DHCPv6 or if the attachment between IPv6 routers was considered. The BC mechanism solved the problem by defining, during the negotiation phase, the node running the DHCP client and the node running the DHCP server; it guaranteed that a single DHCP server was running after the attachment procedure completed, and that a single IP network was created.

In order to demonstrate that the BC mechanism also works for the attachment between networks, Scenario 2 (Figure 6(b)) was also tested using the BC prototype. After running the BC mechanism between the BNs of each network (in this case, Node2@Network1 and Node3@Network2) we have verified that (1) the BNs could establish IP connectivity between them, as it happens for the attachment between stand-alone nodes; (2) the BNs were able to negotiate the final addressing scheme and the configuration of each BN after attachment—in this case, the address spaces in each network were maintained and Node 2 was configured as an IPv4 router between the subnets 192.168.20.0/24 and 172.16.0.0/16; (3) Node 1 and Node 4 were directly connected at IP level after the BC mechanism completed its job. If the networks had incompatible or overlapping address spaces, the only difference would be the local configurations at each BN which, in turn, would influence the type of connectivity established between Node 1 and Node 4. In those cases, either one BN or both had to be configured as NID routers instead of IP routers and Node 1 and Node 4 would be directly connected only at NID level, after the BC mechanism completed.

The three scenarios illustrated in Figure 7 (Scenario 3, 4, and 5) were used to compare the two mechanisms. Scenario 3 is purely symmetric and refers to the creation of a PAN, using one or more wireless technologies. Scenario 4 is purely asymmetric and considers the attachment of multiple terminals to an 802.11 access network; the message sequence charts illustrating the execution of the BC mechanism in Scenarios 3 and 4 can be found in [22]. Finally, Scenario 5 considers a prominent scenario for NGNs: the extension of a wired access network by means of an 802.11s [28] Wireless Mesh Network (WMN), where terminals attach to the Mesh Access Points (MAPs)

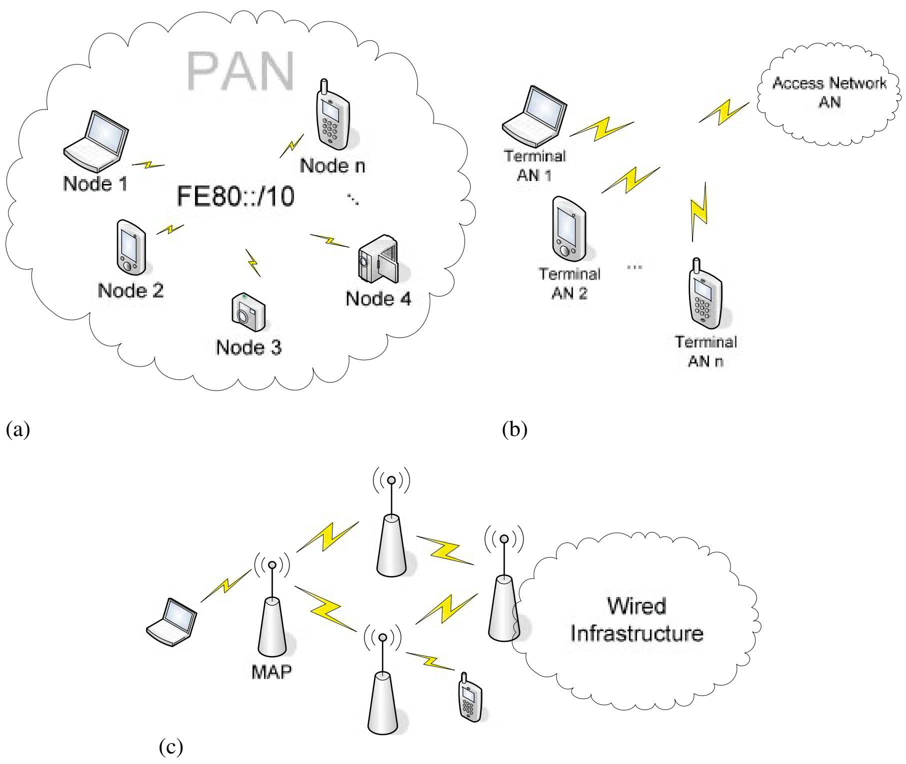

Figure 7. Scenarios used for comparing the BC and legacy mechanisms: (a) Scenario 3; (b) Scenario 4; and (c) Scenario 5.

[28] and MAPs form a square lattice topology. In the three scenarios, we assume the devices are dual-stack. IPv4 and IPv6 and their auto-configuration mechanisms were considered in the analysis. For IPv4, the DHCP protocol [4] and the Dynamic Configuration of IPv4 Link-Local Addresses [13] along with the Address Resolution Protocol (ARP) [14], used for DAD, were considered. For IPv6, the IPv6 Stateless Address Auto-configuration [15] and DHCPv6 [18], together with the NDP [16] and the MLD protocol [17] were considered.

7.2. *Mathematical expressions*

This section formalizes some of the concepts used in the theoretical analysis of the legacy and BC auto-configuration mechanisms. We first refer to the expression used to compute the overhead reduction (OR) provided by the BC mechanism. Then, we provide the expressions defined in [29] for calculating the energy consumed by an 802.11b Network Interface Card (NIC) to send/receive unicast/broadcast frames. Finally, we derive the expressions used for calculating the energy wasted when the legacy mechanism is used in Scenarios 4 and 5.

The *Overhead Reduction OR* provided by the BC mechanism is defined as follows:

$$OR(\%) = \frac{L_{\text{o}} - BC_{\text{o}}}{L_{\text{o}}} \times 100 \tag{1}$$

Table I. Energy consumed by an 802.11b NIC as a function of the size of the unicast/broadcast frame being sent/received.

Action	Managed mode (μJ)	Ad-hoc mode (μJ)
Unicast send	$0.74\times S+431$	$3.85\times S+431$
Broadcast send	$2.36\times S+272$	$5.47\times S+272$
Unicast recv	$0.38\times S+316$	$3.49\times S+316$
Broadcast recv	$0.52\times S+50$	$3.63\times S+50$

where L_o and BC_o represent the overhead—total number of bytes or total number of signalling frames exchanged—for the legacy and BC mechanisms, respectively.

The energy consumed by an 802.11b NIC to send/receive unicast/broadcast frames has been analysed in [29]. Different linear equations are provided for each case: (1) unicast frame transmission; (2) unicast frame reception; (3) broadcast frame transmission; and (4) broadcast frame reception. Table I summarizes those equations.

In order to evaluate the energy wasted when using the legacy mechanism in practice, concrete scenarios need to be considered. In what follows, we derive the mathematical expressions for computing the energy wasted per frame in Scenarios 4 and 5.

The *total energy wasted in Scenario 4 per broadcast frame* (E_{w_4}) can be computed using the following equation:

$$E_{w_4}(J/\text{frame}) = E_{s_b,m} + \sum_{i=1}^{N} E_{r_b,m,i} \tag{2}$$

where N defines the number of terminals already attached to the access network, $E_{s_b,\mathrm{m}}$ represents the energy consumed by an 802.11 NIC to send a given broadcast frame in managed mode, and $E_{r_b,m,i}$ represents the energy consumed to receive the same broadcast frame in managed mode for terminal i. These two values can be easily computed using the equations in Table I.

The *total energy wasted per broadcast frame in Scenario 5* (E_{w_5}) within the WMN can be defined as follows:

$$\begin{aligned} E_{w_5}(J/\text{frame}) = {} & s_n^2.E_{s_b,a} + 4(2.E_{r_b,a}) + 4(s_n-2)(3E_{r_b,a}) \\ & + (s_n-2)^2.(4E_{r_b,a}) + E_{r_b,a}, \forall s_n \in \mathbb{N} \wedge s_n \geqslant 2 \end{aligned} \tag{3}$$

where s_n denotes the number of WMN nodes in a side of the square lattice (e.g. a 4-node WMN has $s_n=2$ and a 9-node WMN has $s_n=3$), $E_{s_b,a}$ is the energy consumed to send a broadcast frame in ad-hoc mode, and $E_{r_b,a}$ is the energy consumed to receive a broadcast frame in ad-hoc mode. Equation (3) considers the energy consumed to send a broadcast frame within the WMN using a flooding mechanism, as defined in [28], and the energy consumed to receive the same frame. The number of broadcast frames received by each WMN node depends on its number of neighbours. For the square lattice topology considered here, a WMN node can have 2, 3, or 4 neighbours. The number of times the WMN node receives the broadcast frame is equal to its number of neighbours, due to the flooding mechanism. The proof that the expressions included in Equation (3) for the number of nodes with 3 and 4 neighbours hold for $s_n \geqslant 2$ can be found in the appendix; the number of nodes with 2 neighbours is always 4 (the WMN nodes in the corners of the square lattice). The

Table II. Message sizes and corresponding number of retransmissions considered in the theoretical analysis.

Message	Size (bytes)	No. of retrans.
ARP_REQUEST	28	2
DHCP4_DISCOVER	321	2
DHCP4_OFFER	368	—
DHCP4_REQUEST	365	—
DHCP4_ACK	302	—
DHCP6_SOLICIT	176	2
DHCP6_ADVERTISE	221	—
DHCP6_REQUEST	198	—
DHCP6_REPLY	241	—
I1	20	—
I2	720	—
MLD_REPORT	126	1
NDP_NEIGHBOR_SOLICITATION	64	2
NDP_ROUTER_ADVERTISEMENT	104	—
NDP_ROUTER_SOLICITATION	56	2
R1	640	—
R2	60	—

last term in Equation (3) is related to the energy required to receive the frame from a node outside the WMN, as it happens when a WMN is used to extend a wired infrastructure; if the WMN is a stand-alone network, this term disappears.

7.3. Results

This section presents the results obtained for Scenarios 3, 4, and 5. We characterize the auto-configuration delay, the OR provided by the BC mechanism, and the wasted energy when the legacy mechanism is used. Overheads do not consider Layer 2 headers, so the results are made independent of the medium used. The analysis was performed using messages sizes according to the specifications [4, 13–18, 30, 31], or according to the typical sizes found in practice when standards leave values as open. Table II summarizes the size of the messages and the corresponding number of retransmissions considered in our analysis. The results provided herein are meant to give insights on the differences between the use of the legacy and BC mechanisms.

The plots of Figure 8 show the total overhead incurred by the attachment process when the legacy and the BC mechanisms are used in Scenarios 3, 4, and 5 and IPv6 is used to establish IP connectivity; similar plots for IPv4 can be found in [22]. For Scenario 3, the total overhead represented by the curves considers one attachment per device performed while forming the PAN. There is a significant saving provided by the BC mechanism. For instance, for a PAN of 30 devices, the BC mechanism saves about 60 kB of signalling, whereas for 50 devices it can save about 100 kB. The OR is shown in Table III. The reduction is slightly higher when IPv4 is used to establish IP connectivity within the PAN. For Scenario 4, it is worth noting that the BC mechanism saves about 0.5 and 1 MB of signalling when 500 and 1000 terminals, respectively, perform a single attachment to the access network. The saving is even more relevant if we consider multiple attachments per terminal. For instance, if 10 attachments per terminal are considered, the BC mechanism saves about 5 MB for 500 attaching terminals and about 10 MB

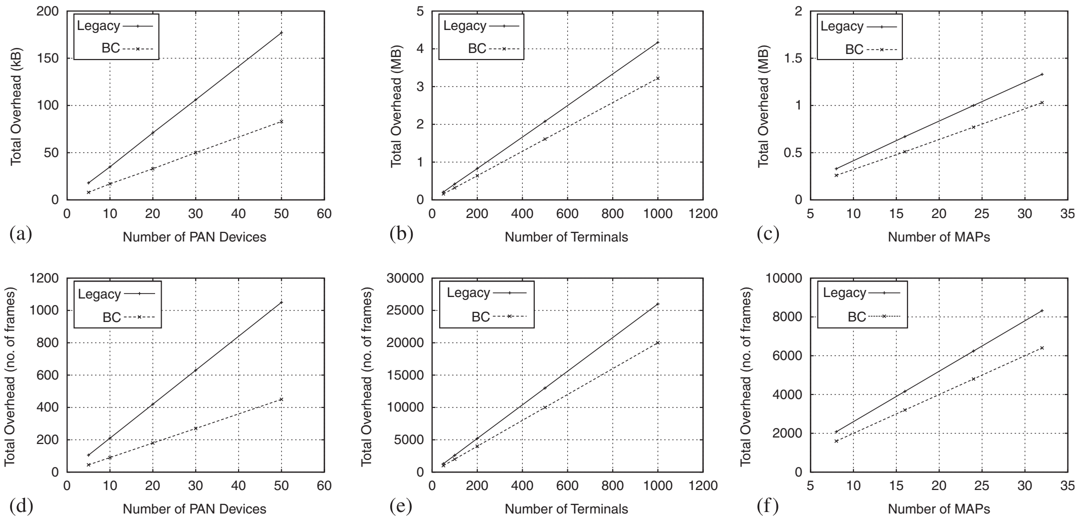

Figure 8. Total overhead incurred by each mechanism for Scenarios 3, 4, and 5, when IPv6 is used for establishing IP connectivity: (a) Scenario 3 (no. of bytes); (b) Scenario 4 (no. of bytes); (c) Scenario 5 (no. of bytes); (d) Scenario 3 (no. of frames); (e) Scenario 4 (no. of frames); and (f) Scenario 5 (no. of frames).

Table III. Overhead reduction when using the BC mechanism for Scenarios 3, 4, and 5.

		Overhead reduction (%)	
Scenario	Active Version	Bytes	Frames
3	IPv4	58	67
	IPv6	53	57
4	IPv4	28	50
	IPv6	23	30
5	IPv4	28	50
	IPv6	23	30

for 1000 attaching terminals. The OR provided by the BC mechanism with respect to the legacy mechanism is also shown in Table III. As in Scenario 3, the reduction is slightly higher when IPv4 is the active version. Finally, for Scenario 5, the same OR of Scenario 4 is achieved (see Table III). The curves shown in Figure 8 were obtained considering that, on average, ten 802.11 terminals per MAP perform one attachment to the WMN. They represent the signalling overhead generated by each mechanism within a WMN. It can be observed that, for instance, for a 16-node WMN, BC saves 150 kB of signalling for a single attachment per terminal, while for a 32-node WMN (the maximum size targeted for an 802.11s WMN [28]), it can save 300 kB of signalling.

For some communication media, it is more suitable to consider the overhead expressed in the number of frames, instead of the number of bytes, since the medium access overhead contributes the most to both the time and energy required to transmit a frame; the size of the frame is a second-order factor. This is the case of 802.11. Table III provides the OR regarding the number of signalling frames. The plots of Figure 8(d)–(f) show the number of signalling frames sent by each mechanism in each scenario. For Scenario 4, 3000 and 6000 frames are saved by the BC mechanism when 500 and 1000 terminals are considered, respectively. It is interesting to note that this amount of frames could be used for transporting about 4.3 and 8.6 MB of user data, respectively, if frames of 1500 bytes were considered. These are much higher values than the values mentioned above, expressed in number of bytes. The same type of reasoning could be performed for Scenarios 3 and 5.

In order to assess the energy wasted by the legacy mechanism, due to the use of a trial-and-error approach, we have considered Scenarios 4 and 5; IPv4 connectivity is established in both cases. This energy is saved by the BC mechanism and can be used to send user data instead. The additional energy required to transmit the BCP information elements piggybacked over the attachment protocol is negligible, since the fixed energy cost of transmitting/receiving frames is more significant [29]. The plots of Figure 9 show the amount of energy wasted when using the legacy mechanism in these scenarios, and the amount of data that could be transmitted/received by an 802.11b NIC using the wasted energy. The plots were obtained using the equations provided in Section 7.2. In our calculations for Scenarios 4, we considered that terminals attach sequentially; as such, the number of nodes receiving the broadcast messages sent out by an attaching terminal increases linearly as new devices attach. For Scenario 5, the average number of terminals per MAP attaching to the WMN was set to 10. In both scenarios, we evaluated the energy consumed along 30 days, assuming a single attachment per terminal per day. The plots in Figure 9(a),(c) were obtained by considering all the unnecessary messages broadcasted by the legacy mechanism in each scenario. These messages are both unnecessarily transmitted by a node and received by the nodes attached to the same link; both transmission and reception of messages contribute to the total wasted energy [29]. For Scenario 4, the total wasted energy increases exponentially with the number of attaching terminals. For 1000 terminals, about 17.3 kJ of energy is wasted when using the legacy mechanism, which corresponds to the energy required by an 802.11b NIC to transmit more than 15 GB of data and receive more than 27 GB (cf. Figure 9(b)). For Scenario 5, we analysed the energy wasted within the WMN; the energy wasted by the terminals attaching to the WMN is not taken into account. The total wasted energy increases exponentially with the number of MAPs forming the WMN. For a 25-node WMN, 4.6 kJ of energy is wasted by the legacy mechanism. With this amount of energy, an 802.11b NIC could transmit more than 2 GB of data and receive about 2.5 GB, as shown in Figure 9(d). These results are mostly relevant for energy constrained WMNs, such as emerging solar-powered WMNs [32]. Different results for both scenarios would be obtained if other NIC was considered; here we aim at reasoning about the order of magnitude of the wasted energy.

We now present the auto-configuration delay introduced by each mechanism in Scenario 3. The delay per PAN device during the auto-configuration of an IPv4 address was measured experimentally using Ethereal [33]; the logs are available at [34]. Ten samples were taken into account. Table II presents the measured mean delay and the corresponding 95% confidence interval, as well as the delay reduction—the difference between delays normalized to the legacy mechanism delay. It is important to realize that these values are implementation dependent; for instance, the number of retransmissions of the *DHCPDISCOVER* message before the

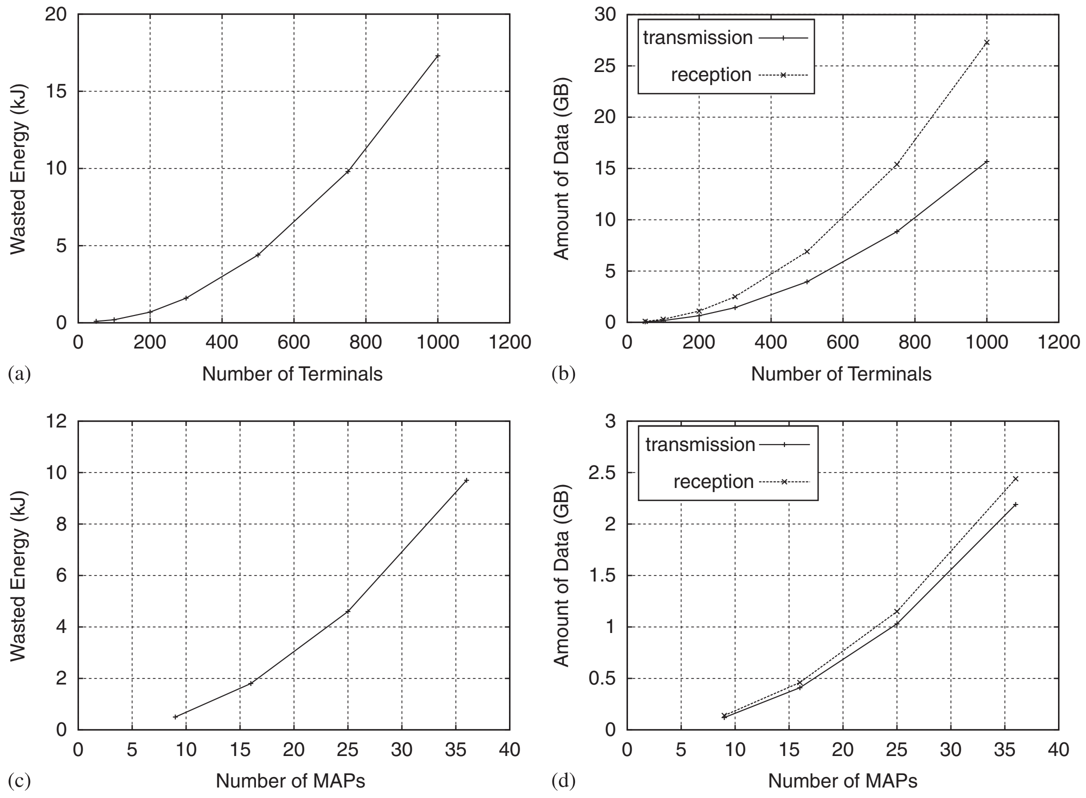

Figure 9. Total energy wasted in Scenarios 4 and 5 by the legacy mechanism and the amount of data that could be transmitted/received by an 802.11b NIC using the wasted energy: (a) Scenario 4 (wasted energy); (b) Scenario 4 (amount of data); (c) Scenario 5 (wasted energy); and (d) Scenario 5 (amount of data).

Table IV. Auto-configuration delays introduced by the BC and legacy mechanisms in Scenario 2.

Parameter	Legacy (s)	BC (s)	Delay reduction (%)
Mean	61.8	3.0	95.2
Confidence interval (95%)	[61.6, 62.0]	[2.2, 3.7]	[94.0, 96.4]

device gives up (in our case 2 retransmissions), will have impact on the auto-configuration delay (Table IV).

The delay introduced by the legacy mechanism is greater than 1 min, while the BC mechanism takes 3.0 s, on average, to complete. This has to do with the trial-and-error approach assumed by the legacy mechanism. When using the legacy mechanism, a PAN device first tries to auto-configure an IPv4 address using DHCP by sending out multiple DHCPDISCOVER messages until it realizes there is not any DHCP server running. Subsequently, it tries the alternative auto-configuration mechanism and configures an IPv4 link-local address. The BC mechanism saves 58.8 s, on average, by negotiating in advance the proper auto-configuration mechanism to be used. During the negotiation phase, BCMs can conclude that IP address auto-configuration can only be performed using the Dynamic Configuration of IPv4 Link-Local Addresses mechanism. Consequently, the delay introduced by the attempt to auto-configure IP connectivity through DHCP is eliminated from the BC mechanism.

7.4. Discussion

The heterogeneity currently found in IP networks and the advent of a new communication paradigm brings up new problems. State-of-the-art solutions either do not deal with these problems at all or do not deal with them in the most efficient way. The BC mechanism addresses the new problems and represents a solution more efficient than the legacy mechanism used to auto-configure IP connectivity.

By testing Scenarios 1 and 2, we have confirmed that the BC mechanism can address the new envisioned communication paradigm, where symmetric attachments can take place, devices may have dynamic roles, and two IP versions and corresponding auto-configuration mechanisms are present. We have demonstrated that the BC mechanism can establish IP connectivity in scenarios where the legacy mechanism cannot. On the other hand, through experimental and theoretical analysis, we verified that the BC mechanism is significantly more efficient than the legacy mechanism. The BC mechanism avoids both the transmission and reception/processing of unnecessary messages by each network device connected to the link over which IP connectivity is being configured. This allows energy and CPU time saving, an important issue namely for devices with limited resources and running on battery power, such as handheld, sensor devices, and solar-powered access points [32]. In addition, the BC mechanism avoids the auto-configuration delay that may be introduced by the legacy mechanism, which renders IP auto-configuration a slow process; for example, the bootstrapping of a PAN may take more than 1 min. Still, we shall observe that the actual overhead and delay reduction provided by the BC mechanism depends on multiple factors, such as the scenario being considered, the network layer, and the corresponding auto-configuration mechanism used for IP connectivity configuration. Also, the actual energy saving depends on the hardware and the specific technology(ies) considered.

In the evaluation presented, we considered IPv4 and IPv6 for establishing IP connectivity and their corresponding auto-configuration mechanisms. Moreover, we evaluated the BC mechanism for three representative scenarios. For these scenarios, the BC mechanism provides significant reductions in overhead and auto-configuration delay, and it may save a considerable amount of energy. The benefits of BC would be even more significant if: (1) further possible network layers and/or auto-configuration mechanisms were considered; (2) the number of retransmissions of the DHCPDISCOVER message, while attempting to auto-configuring IP connectivity using DHCP, was higher. Our analysis considered two retransmissions, according to [4]. In practice, higher number of retransmissions may take place. For instance, the DHCP implementation under Linux OS considers five retransmissions. In that case, the overhead and delay reductions and the wasted energy would be higher.

Although the BC mechanism was designed having ANs in mind, and with the purpose of being integrated into the AN framework [24], it can be used in other setups. BC can be integrated in any existing attachment procedure. For instance, it could be integrated within the 802.11 attachment procedure usually carried out using the Extensible Authentication Protocol (EAP) [35], with the

BCP information elements piggybacked over the EAP messages. On the otherhand, it is not limited to work together with the NID internetworking solution. Solutions such as NAT64 and others referred in [22] can be employed as well.

8. CONCLUSION

The paradigm in mobile communication networks is changing, with symmetric attachment between peers gaining momentum and two IP versions coexisting. In this paper, we presented the BC mechanism, a new efficient mechanism used to establish IP connectivity between communicating peers (ANs). By using a proof-of-concept prototype, experimental results, and theoretical analysis, we showed the usefulness of the mechanism, both from the efficiency and functional points of view. Its deployment allowed us to prove that it does work in practice and that its implementation and integration with legacy IP auto-configuration solutions is a straightforward process. The analysis herein performed focused on overhead, auto-configuration delay, and energy consumption. The obtained results give the first insights on the real benefits provided by the BC mechanism. The BC mechanism is significantly more efficient than the legacy mechanism. It avoids both the transmission and reception/processing of unnecessary messages and avoids the auto-configuration delay that may be introduced by the legacy mechanism. Although this mechanism was designed having ANs in mind, it can be used in other setups. BC can be integrated in any existing attachment procedure. Also, it can work together with internetworking solutions other than the NID internetworking layer defined in the AN overall architecture.

As future work, we shall consider the extension of the BC mechanism to address multi-homed devices and networks more efficiently and the renegotiation of the auto-configuration methods used in each concrete scenario to establish IP connectivity. Although the current results are already relevant and prove the benefits of the BC mechanism, the extension of the analysis to more dynamic scenarios where, for instance, the underlying IP auto-configuration legacy mechanism are renegotiated, may further emphasize its advantages; this is left for future work too.

APPENDIX A

The expression for the number of nodes with 3 neighbours (N_3) when a square lattice topology is considered is demonstrated using the mathematical induction method. Our statement is that N_3 is defined by:

$$N_3 = 4.(s_n - 2) \quad \forall s_n \in \mathbb{N} \wedge s_n \geqslant 2 \tag{A1}$$

Proof
First, we prove that Equation (A1) is true for $s_n = 2$. For $s_n = 2$, we have a 4-node square lattice topology. In this case, there are only nodes with two neighbours. As such, $N_3 = 0$. Using Equation (A1) we get:

$$N_3 = 4 \cdot (2 - 2) = 0 \tag{A2}$$

The expression then holds for $s_n = 2$.

We now need to prove that the equation is valid for $s_n = k+1$, assuming that it is valid for $s_n = k$. For $s_n = k$ we get:

$$N_3(k) = 4\cdot(k-2) \tag{A3}$$

For $s_n = k+1$ we have:

$$N_3(k+1) = 4\cdot(k+1-2) = 4\cdot(k-1) \tag{A4}$$

But, $N_3(k+1)$ can also be defined as a function of $N_3(k)$:

$$N_3(k+1) = N_3(k) + 4 \tag{A5}$$

since when the side of the square lattice is increased by one, one node per side is added to the set of nodes having 3 neighbours. If Equation (A5) is simplified we can prove that it leads to the same result obtained using Equation (A1):

$$N_3(k+1) = 4\cdot(l-2) + 4 = 4\cdot(l-2+1) = 4\cdot(l-1) \tag{A6}$$

Thus, the expression holds for all s_n.

The expression for the number of nodes with 4 neighbours (N_4) when a square lattice topology is considered is demonstrated using the same method. Our statement is that N_4 is defined by:

$$N_4 = (l-2)^2 \quad \forall s_n \in \mathbb{N} \wedge s_n \geqslant 2 \tag{A7}$$

Proof
First, we prove that Equation (A7) is true for $s_n = 2$. For $s_n = 2$, we have a 4-node square lattice topology. In this case, $N_4 = 0$, as there are only nodes with two neighbours. Using Equation (A7) we get:

$$N_4 = (2-2)^2 = 0 \tag{A8}$$

The expression then holds for $s_n = 2$.

We need to prove that the equation is valid for $s_n = k+1$ too, assuming that it is valid for $s_n = k$. For $s_n = k$ we get:

$$N_4(k) = (k-2)^2 \tag{A9}$$

For $s_n = k+1$ we have:

$$N_4(k+1) = (k+1-2)^2 = (k-1)^2 \tag{A10}$$

But, $N_4(k+1)$ can also be defined as a function of $N_4(k)$:

$$N_4(k+1) = N_4(k) + (k+1-2)^2 - (k-2)^2 \tag{A11}$$

where $(k+1-2)^2 - (k-2)^2$ represents the number of nodes added to the set of nodes having 4 neighbours, when the side of the square lattice is increased by one. If Equation (A11) is simplified we can prove that it leads to the same result obtained using Equation (A7):

$$N_4(k+1) = (k-2)^2 + (k+1-2)^2 - (k-2)^2 = (k+1-2)^2 = (k-1)^2$$

Thus, the expression holds for all s_n.

ACKNOWLEDGEMENTS

This paper describes work undertaken in the context of the Ambient Networks project, which is part of the EU's IST program. Over 30 organizations from Europe, Canada, and Australia are involved in this Integrated Project, which runs in 2006–2007 in its second phase. The views and conclusions contained herein are those of the authors and should not be interpreted as necessarily representing the Ambient Networks Project.

The authors thank the anonymous reviewers for their fruitful comments which contributed to improve the quality of this paper.

REFERENCES

1. Durand A, Droms R, Haberman B, Woodyatt J. Dual-stack Lite Broadband Deployments Post IPv4 Exhaustion. IETF Internet Draft, draft-ietf-softwire-dual-stack-lite-00, March 2009.
2. Campos R *et al.* Dynamic and automatic interworking between personal area networks using composition. *Proceedings of 16th Annual IEEE International Symposium on Personal Indoor and Mobile Radio Communication*, Berlin, Germany, September 2005.
3. Campos R, Ricardo M. Dynamicand automatic connection of personal area networks to the global internet. *Proceedings of ACM IWCMC'06*, Vancouver, Canada, July 2006.
4. Droms R. *Dynamic Host Configuration Protocol*. IETF RFC 2131, March 1997.
5. Soliman H. *Mobile IPv6 Support for Dual Stack Hosts and Routers*. IETF Internet Draft, draft-ietf-mext-nemo-v4traversal-09, February 2009.
6. Chown T, Venaas S, Strauf C. *Dynamic Host Configuration Protocol* (*DHCP*): *IPv4 and IPv6 Dual-Stack Issues*. IETF RFC 4477, May 2006.
7. IST FP6 Ambient Networks. Available from: http://www.ambient-networks.org [31 March 2009].
8. Tsirtsis G, Srisuresh P. *Network Address Translation—Protocol Translation* (*NAT-PT*). IETF RFC 2766, February 2000.
9. Srisuresh P, Holdrege M. *IP Network Address Translator* (*NAT*) *Terminology and Considerations*. IETF RFC 2663, August 1999.
10. Baccelli E. *Address Autoconfiguration for MANET*: *Terminology and Problem Statement*. IETF Internet Draft, draft-ietf-autoconf-statement-04, February 2008.
11. Bernardos C, Calderon M, Moustafa H. *Ad Hoc IP Address Autoconfiguration*. IETF Internet Draft, draft-bernardos-manet-autoconf-survey-04, November 2008.
12. Zeroconf Work Group. Available from: http://www.zeroconf.org [31 March 2009].
13. Cheshire S, Aboba B, Guttman E. *Dynamic Configuration of IPv4 Link-Local Addresses*. IETF RFC 3927, May 2005.
14. Plummer D. *An Ethernet Address Resolution Protocol*. IETF RFC 826, November 1982.
15. Thomson S, Narten T. *IPv6 Stateless Address Autoconfiguration*. IETF RFC 4862, September 2007.
16. Narten T, Nordmark E, Simpson W. *Neighbor Discovery for IP Version 6* (*IPv6*). IETFRF C4861, September 2007.
17. Vida R, Costa L. *Multicast Listener Discovery Version 2* (*MLDv2*) *for IPv6*. IETF RFC3810, June 2004.
18. Droms R *et al. Dynamic Host Configuration Protocol for IPv6* (*DHCPv6*). IETF RFC 3315, July 2003.
19. Rekhter Y, Li T, Hares S. *A Border Gateway Protocol 4* (*BGP-4*). IETF RFC4271, January 2006.
20. Carpenter B, Moore K. *Connection of IPv6 Domains via IPv4 Clouds*. IETF RFC 3056, February 2001.
21. Campos R, Ricardo M. Dynamic autoconfiguration in 4G networks: problem statement and preliminary solution. *Proceedings of the 1st International ACM Workshop on Dynamic Interconnection of Networks* (*DIN'05*), Cologne, Germany, September 2005.
22. Ahlgren B *et al*. *Connectivity and Dynamic Internetworking Prototype and Evaluation. FP6-CALL4-027662-AN P2/D23-E.2*, V1.0, Ambient Networks project public deliverable, December 2007.
23. Wing D, Ward D, Durand A. *A Comparison of Proposals to Replace NAT-PT*. IETF Internet Draft, draft-wing-nat-pt-replacement-comparison-02, September 2008.
24. Johnsson M *et al*. *Ambient Network System Description. FP6-CALL4-027662-AN P2/ D07-A2*, V7.0, December 2007.
25. Rinta-aho T, Campos R, Mehes A, Meyer U, Sachs J, Selander G. Ambient network attachment. *Proceedings of IST Summit*, Budapest, July 2007.

26. Crocker D, Overell P. *Augmented BNF for Syntax Specifications*: *ABNF* RFC 4234, October 2005.
27. Rembarz R. *D17-H.4 Ambient Control Space Prototype Design*. FP6-CALL4-027662-AN P2/D17-H.4, V1.0, June 2007.
28. Bahr M. Update on the hybrid wireless mesh protocol of IEEE 802.11s. *Proceedings of MASS 2007*, Pisa, 2007.
29. Feeney L, Nilsson M. Investigating the energy consumption of a wireless network interface in an ad hoc networking environment. *Proceedings of INFOCOM 2001*, Anchorage, U.S.A., 2001.
30. Alexander S, Droms R. *DHCP Options and BOOTP Vendor Extensions.* IETF RFC 2132, March 1997.
31. Droms R. *DNS Configuration Options for Dynamic Host Configuration Protocol for IPv6* (*DHCPv6*). IETF RFC3646, December 2003.
32. Todd T, Sayegh A, Smadi M, Zhao D. The need for Access point power saving in solar powered WLAN mesh networks. *IEEE Network* 2008; **22**(3):4–10.
33. Network Protocol Analyzer. Ethereal. Available from: http://www.ethereal.com [31 March 2009].
34. Log files. Available from: http://telecom.inescporto.pt/∼rcampos/logsIPautoconf.zip.
35. Aboba B *et al.* Extensible Authentication Protocol (EAP). IETF RFC 3748, June 2004.

AUTHORS' BIOGRAPHIES

Rui Campos received a Diploma degree in Electrical and Computer Engineering in 2003 from the University of Porto, Portugal. He works as a researcher at INESC Porto and is pursuing his PhD in Electrical and Computer Engineering. His research interests include mobile communications, network autoconfiguration, and spanning tree algorithms.

Manuel Ricardo received a Diploma degree in 1988, an MS in 1992, and a PhD in 2000, all in Electrical and Computer Engineering from the University of Porto, Portugal. He is an associate professor at the University of Porto, where he gives courses in mobile communications and computer networks. He also leads the Communication Networks and Services Area at INESC Porto.