“DNCP: Dynamic Node Configuration Protocol”

By

Rafael Roque Aschoff

Master Thesis

Federal University of Pernambuco
posgraduacao@cin.ufpe.br
www.cin.ufpe.br/~posgraduacao

RECIFE, MAY/2010
Rafael Roque Aschoff

"DNCP: Dynamic Node Configuration Protocol"

THIS THESIS HAS BEEN SUBMITTED TO THE INFORMATICS CENTER, OF THE FEDERAL UNIVERSITY OF PERNAMBUCO, IN PARTIAL FULFILLMENT OF THE REQUIREMENTS FOR THE DEGREE OF MASTER OF SCIENCE IN COMPUTER SCIENCE.

SUPERVISOR: Prof. Dr. Djamel F. H. Sadok

RECIFE, MAY/2010
Aschoff, Rafael Roque
vii, 60 folhas : il., fig., tab.


Inclui bibliografia.


004.6 CDD (22. ed.) MEI2010 – 097
To my parents Ronaldo and Roseane, and my sister Amanda.
Acknowledgments

Firstly, I would like to thank my parents and my sister who are an essential part of my existence, supporting and encouraging me to overcome present and future challenges. I also would like to thank the professors and friends from GPRT/CIn/UFPE and, especially, Prof. Djamel, Prof. Judith and Prof. Eduardo for their dedication and patience, providing a great opportunity for my personal and professional growth. I thank Ericsson, and especially Lars Westberg and Victor Souza for their efforts on the project issues. In addition, I am grateful to all friends that somehow supported me and helped on this personal realization.
Table of Contents

ACKNOWLEDGMENTS .......................................................................................................................... I
TABLE OF CONTENTS ........................................................................................................................... II
LIST OF FIGURES ................................................................................................................................. IV
ABBREVIATIONS AND ACRONYMS ...................................................................................................... V
ABSTRACT ............................................................................................................................................... VI
RESUMO................................................................................................................................................... VII

1 INTRODUCTION ............................................................................................................................... 1
  1.1 Motivation ...................................................................................................................................... 2
  1.2 Objectives ..................................................................................................................................... 3
  1.3 Work Structure ............................................................................................................................. 3

2 BACKGROUND AND RELATED WORK ............................................................................................. 4
  2.1 Background .................................................................................................................................. 4
  2.1.1 Ad Hoc Network ....................................................................................................................... 4
  2.1.2 Ad Hoc Routing ....................................................................................................................... 5
  2.1.3 Address Auto Configuration ...................................................................................................... 7
  2.2 Stateless Approaches .................................................................................................................... 10
  2.2.1 Strong Duplication Address Detection (SDAD) ........................................................................ 10
  2.2.2 Weak Duplicate Address Detection (WDAD) .......................................................................... 12
  2.2.3 Passive Duplicate Address Detection (PDAD) ........................................................................ 12
  2.2.4 Ad Hoc IP Address Autoconfiguration ................................................................................... 13
  2.3 Stateful Approaches ..................................................................................................................... 13
  2.3.1 Agent Based Addressing .......................................................................................................... 15
  2.3.2 MANETconf ............................................................................................................................ 16
  2.3.3 BUDDY ...................................................................................................................................... 17
  2.3.4 PROPHET .................................................................................................................................. 17
  2.3.5 PRIME DHCP .......................................................................................................................... 19
  2.4 Hybrid Approaches ....................................................................................................................... 20
  2.4.1 Hybrid Centralized Query-Based Autoconfiguration (HCQA) ................................................... 21
  2.4.2 PACMAN ................................................................................................................................... 21
  2.5 Comments ..................................................................................................................................... 22

3 DNCP ................................................................................................................................................. 24
  3.1 Overview ...................................................................................................................................... 24
  3.2 Address Allocation ....................................................................................................................... 25
  3.2.1 Binary Number Generation Function ...................................................................................... 26
  3.2.2 Local Addresses Request ........................................................................................................ 31
  3.2.3 Remote Address Request ........................................................................................................ 33
  3.3 Address Maintenance .................................................................................................................... 33
  3.3.1 Address Reclamation .................................................................................................................. 34
  3.3.2 Partition and Merge .................................................................................................................... 37

4 EVALUATION METHODOLOGY ........................................................................................................ 39
  4.1 Simulation ..................................................................................................................................... 39
  4.2 Scenarios ....................................................................................................................................... 39
  4.2.1 Scenario A ................................................................................................................................. 40
  4.2.2 Scenario B .................................................................................................................................. 42
  4.3 Metrics ......................................................................................................................................... 43
  4.4 Static Network Results (Scenario A) ............................................................................................. 44
  4.4.1 Latency ....................................................................................................................................... 44
List of Figures

Figure 2.1. (a) Wireless networking with base station. (b) Ad hoc networking. ............................... 5
Figure 2.2. Sending a message from A to G by flooding................................................................. 6
Figure 2.3. Node joins (D) and node leaves (F) the Network......................................................... 8
Figure 2.4. A network gets partitioned and then merges back.......................................................... 9
Figure 2.5. Flow chart of Strong DAD. ......................................................................................... 11
Figure 2.6. Prophet Allocation algorithm scheme.............................................................................. 18
Figure 3.1: Basic flow for address assigns......................................................................................... 26
Figure 3.2: BNG behavior................................................................................................................ 28
Figure 3.3: Tree of identifiers............................................................................................................ 29
Figure 3.4: Sequence of generated seed and the correspondent status............................................ 30
Figure 3.5: Flow chart of local address allocation ........................................................................... 31
Figure 3.6: Example of local address allocation with three nodes.................................................. 32
Figure 3.7: Remote address allocation message flow....................................................................... 33
Figure 3.8: Address reclamation state machine............................................................................... 35
Figure 4.1: Spiral Grid. ..................................................................................................................... 41
Figure 4.2. Spiral grid filled with 49 nodes. ..................................................................................... 42
Figure 4.3. Latency for address configuration (1 up to 100 nodes).................................................. 45
Figure 4.4. Latency for address configuration varying the network size (4 up to 100 nodes) ......... 45
Figure 4.5. Address latency per node for a network sized 25............................................................ 46
Figure 4.6. DNCP network overhead measured in Scenario A......................................................... 48
Figure 4.7. Network overhead without periodic messages for Scenario A....................................... 49
Figure 4.8. Network throughput in bytes/second varying the number of nodes............................ 50
Figure 4.9. Address configuration errors varying the number of nodes......................................... 51
Figure 4.10. Address latency for Scenario B varying the number of nodes...................................... 52
Figure 4.11. DNCP network overhead measured in Scenario B....................................................... 53
Figure 4.12. Network overhead without periodic messages for Scenario A.................................... 54
Figure 4.13. Number of Network IDs generated and remained varying the number of nodes...... 54
Figure 4.14. Number of changes in the network varying the number of nodes............................. 55
### Abbreviations and Acronyms

<table>
<thead>
<tr>
<th>Abbreviation</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>AODV</td>
<td>Ad hoc On-Demand Distance Vector Routing</td>
</tr>
<tr>
<td>BNG</td>
<td>Binary Number Generator</td>
</tr>
<tr>
<td>DAD</td>
<td>Duplicate Address Detection</td>
</tr>
<tr>
<td>DNCP</td>
<td>Dynamic Node Configuration Protocol</td>
</tr>
<tr>
<td>IP</td>
<td>Internet Protocol</td>
</tr>
<tr>
<td>MANET</td>
<td>Mobile Ad Hoc Network</td>
</tr>
<tr>
<td>NID</td>
<td>Network Identifier</td>
</tr>
<tr>
<td>NS-3</td>
<td>Network Simulator version 3</td>
</tr>
<tr>
<td>NS-2</td>
<td>Network Simulator version 2</td>
</tr>
<tr>
<td>OLSR</td>
<td>Optimized Link State Routing</td>
</tr>
<tr>
<td>PDAD</td>
<td>Passive Duplicate Address Detection</td>
</tr>
<tr>
<td>SDAD</td>
<td>Strong Duplicate Address Detection</td>
</tr>
<tr>
<td>UDP</td>
<td>User Datagram Protocol</td>
</tr>
<tr>
<td>UUID</td>
<td>Universally Unique Identifier</td>
</tr>
<tr>
<td>WDAD</td>
<td>Weak Duplicate Address Detection</td>
</tr>
<tr>
<td>WIFI</td>
<td>Wireless Fidelity</td>
</tr>
</tbody>
</table>
Abstract

A Mobile Ad hoc Network (MANET) is a wireless network in which nodes can freely and dynamically self-organize into arbitrary and temporary network topologies without a central coordination or a pre-established communication infrastructure. In such networks, nodes act cooperatively to forward packets for each other, in order to allow communication between mobile nodes not within direct wireless transmission range. In the past few years, the MANET research field has been very prolific within academia, government, and industry. However, there are some open questions regarding deploying of MANETs that hinder the development of real applications. The advancement of research in the MANET area could improve current performance in using these systems, and promote the emergence of new fields of application. Moreover, there is still a great economic potential to be explored in relation to the use of mobile ad hoc networks.

One of the major problems concerning deploying of MANETs is the appropriate configuration of IP addresses. These addresses must be unique within the same routing domain, or in other words, a MANET must not contain two devices configured with the same network address. The static configuration of ad hoc nodes is not feasible due to the dynamically nature of these network, and schemes developed for traditional infrastructure networks are not suitable. These issues have motivated some research designed to allow nodes to configure without human intervention, resulting in a number of methods for automatically configuration of addresses. These researches, however, have several limitations, mainly related to the limited applicability or introduction of high network overload.

This thesis presents a scalable and efficient protocol for automatic allocation and organization of the address space in mobile ad hoc networks. A review of existing work is presented, considering the advantages and limitations of each one. Finally, the work presents the evaluation of the proposed protocol through different scenarios and metrics.

Keywords: MANET, ad hoc, addressing, auto-configuration, addresses allocation.
Resumo

Rede Móvel Ad Hoc (MANET) é uma rede sem fio onde os nós podem se mover livremente e formar, dinamicamente, topologias de rede temporárias e arbitrárias sem qualquer controle centralizado ou infra-estrutura de comunicação estabelecida previamente. Nestas redes, os nós agem de maneira cooperativa, encaminhando pacotes uns aos outros, de forma a possibilitar a comunicação entre pares de nós que se encontram fora do alcance direto de transmissão sem fio. Nos últimos anos, as pesquisas direcionadas à MANETs têm recebido grande atenção do governo, academia e indústria. Contudo, ainda existem algumas questões em aberto relativas à implantação de MANETs que impedem o desenvolvimento de aplicações reais e difusão das mesmas. O avanço das pesquisas nesta área poderia melhorar o desempenho atual no uso destas redes, bem como propiciar o surgimento de novos campos de aplicação. Além disso, ainda existe um grande potencial econômico a ser explorado no que se refere ao uso de redes móveis ad hoc.

Um dos principais desafios relativos à implantação de MANETs é a configuração adequada dos endereços de rede. Tais endereços precisam ser únicos dentro do mesmo domínio de roteamento, ou em outras palavras, um MANET não deve conter dois dispositivos configurados com o mesmo endereço de rede. A configuração estática dos nós ad hoc não é viável devido à natureza dinâmica destas redes, e esquemas tradicionais desenvolvidos para as redes infra-estruturadas não são adequados. Estas questões motivaram algumas pesquisas destinadas à permitir que os nós se configurarem sem intervenção humana, resultando em uma série de métodos para configuração automática de endereços. Estas soluções, no entanto, apresentam uma série de limitações, relacionadas principalmente à aplicabilidade em cenários restritos ou introdução elevada de sobrecarga na rede.

Este trabalho apresenta um método escalável e eficiente para alocação e organização do espaço de endereçamento em redes móveis ad hoc. Uma revisão bibliográfica dos trabalhos existentes é apresentada, considerando as vantagens e limitações de cada um. Por fim, o trabalho apresenta uma avaliação preliminar do método proposto, considerando diferentes cenários e métricas.

Palavras-chave: MANET, ad hoc, endereçamento, autoconfiguração, alocação de endereços.
1 Introduction

"The rapadura is sweet but not soft."

Brazilian Proverb

Wireless communication is used as a term for transmission of information from one place to another without using cables. It is not a new idea, as early as 1896, Guglielmo Marconi demonstrated the transmission and reception of Morse Code based radio signals over a distance of 2 or more kilometers and one year later was awarded a world's first patent in radio telecommunications [MAR97]. From this beginning, wireless communications has developed into a key element of modern society. From satellite transmission, radio and television broadcasting to the now ubiquitous mobile telephone, wireless communications has revolutionized the way societies function.

Advances in wireless communication have contributed in a straightforward way on the capabilities of portable devices such as notebooks, cellular phones, and personal digital assistants (PDAs). Today, these devices provide a rich computing environment and multiples communication methods based on different radio technologies such as infrared, Bluetooth [BLUE05] (IEEE 802.15.1), Wireless-Fidelity [WIFI07] (WiFi, IEEE 802.11), Ultra Wide Band [UWB03] (UWB, IEEE 802.15.3), to mention a few. These wireless technologies allow people and devices to access and share information in areas without pre-existing communications infrastructure.

The basic solution to meet such requirements is to allow users of mobile nodes with compatible wireless technologies to set up a possibly short lived network just for the communication needs at a moment. This propriety leads to the formation of mobile ad hoc networks (MANETs), where the wireless mobile nodes freely and dynamically self-organize themselves into arbitrary and temporary network topologies, and cooperatively forward packets for each other, in order to allow unicast multi hop communication. Typical application scenarios of MANETs include data sharing between people, meetings and conferences, emergency operations such as disaster rescue, and battlefield communications.

Nonetheless, ad-hoc networks also are often characterized with a high degree of instability, moderate-to-high user mobility, ungoverned growth and low reliability. The network topology changes frequently and is unpredictable, hence requiring a permanent adaptation and
reconfiguration. In such networks, the address assignment consists in one of the biggest challenges to MANET deployment. In practice, in order to communicate among themselves, ad hoc nodes need to configure their network interface(s) with local addresses that are valid within an ad hoc network. Furthermore, the majority of routing protocols assume that mobile nodes in ad hoc networks are configured a priori with unique address identifiers before they enter in the network. Due to dynamic and unpredicted behavior of ad hoc networks, the pre-configuration of static address is not always possible, and conventional addressing mechanisms do not fit well.

1.1 Motivation

Address allocation in MANET must configure each node with a unique IP address before routing can begin. When a new node desires to participate in the network, a MANET must be able to select (or to generate), allocate and assign a conflict free network address for this node. The address configuration process must be efficient and fast since a node cannot participate in communication until it has a configured a unique address.

The peculiar characteristics of MANETs present many challenges with regards to addresses organization that are not found in wired networks. First, due to unpredictable nature of network forming, the manual or static configuration of node addresses is not always possible and not recommended. Second, the usual centralized authority presented in traditional networks that dynamic allocate addresses is not available, in consequence of the distributed and flat configuration of mobile nodes. Finally, as nodes are free to move arbitrarily during its session in the MANET, temporally or even permanent disconnection may occur. In other words, while a node moves, it can be out of transmission range of the network, which also leads to events such as network partitions and merges, in case of group of nodes moving together. Moreover, in general, the nodes are not aware of such events.

The abrupt departure of nodes, together with partitions and merges, is one of the biggest challenges to address allocation in ad hoc world. If a node gets permanently disconnected from the network, due to low battery charge or breaks, for example, its address becomes unusable. If this happens often, and the lost addresses are not recovered, the network will be quickly depleted of free addresses. On the other hand, if a node is only temporally disconnected, the recovery of its address will force it to request a new address when it joins the network again. The partitions and merges exacerbate this problem due to the number of disconnections. More recoveries may be necessary to allocate addresses to new income nodes in case of partitions, and more addresses releases may take place in case of merges. Moreover, the merge of completely distinct networks that have never seen
each other will almost certainly introduce address conflict, if not properly handled. Keeping the communication overhead, required to deal with such events, at a minimum is also another challenge.

The address allocation in mobile ad hoc networks has recently been seen as a very interesting research topic, and many addressing mechanisms were then designed trying to tackle the major requirements of such networks. Nevertheless, the majority of existing approaches focus on specific scenarios, relies on flood messages for solicitation and duplicate address detection, or do not cover well all presented challenges.

1.2 Objectives

This work has as overall objective the proposal and evaluation of a new address auto-configuration protocol for mobile ad hoc network, which is named DNCP – Dynamic Node Configuration Protocol. This protocol aims to provide solutions for realistic scenarios in a MANET, which seeking to solve all the previously presented challenges in a way to insert a minimum of network overhead, while still be simple and efficient, making a good use of address space.

The generated address must be unique to reduce the control overhead introduced by detection of duplication, and allocated in a distributed way, while maintaining the address space consistence over the time. This means that nodes departure as well as network partitioning and merging must be taking into consideration. The solution is meant to IPv4 networks, but it can also be easily modified for IPv6 networks. Moreover, the solution is not dependent on the underlying routing protocol and can be used with either reactive, proactive, and hybrid routing protocols.

1.3 Work Structure

This work has been organized as follows. Chapter 2 gives some brief background information and review of current related work. Chapter 3 describes in details the Dynamic Node Configuration Protocol. Chapter 4 presents the methodology to evaluate de proposed protocol as well as the results of the experiments done. Finally, the conclusions, including the contributions and future works proposals are presented in Chapter 5.
2 Background and Related Work

“The important thing is never to stop questioning.”

*Albert Einstein*

In this chapter the main concepts about addressing in ad hoc network and their enabling technologies are presented. Section 2.1 starts with a description on the relevant concepts. Section 2.2 presents related work found in literature. At the end of this chapter a discussion of the related work is made.

2.1 Background

This section comments some topics that are necessary to better understand the proposed solution and also contextualize the work. Firstly, we present the main point about ad hoc networks and what differentiates it from standard wireless communications. Next, we describe what is so important about the address allocation and maintenance in such networks.

2.1.1 Ad Hoc Network

As we know, ad hoc networks are a specific kind of wireless communication. Wireless networks are mainly characterized by the transfer of information over a distance without the use of wires. Generally speaking, wireless communications could work in two different modes based on the presence of a base station, a central coordination point in which all communication passes through. Figure 2.1 illustrates both methods, presenting devices that communicate with each other through a base station and also in the absence of it.

In the first mode, as we can see from Figure 2.1(a), whenever a device wants to communicate to another one, it needs to send the information to the base station which will properly forward the traffic to the correct destination device. This case is also known as infrastructure network, since in this mode we need a previously configured component, at least the base station. This is the most common mode of wireless communications, sometime called standard or traditional mode.
Figure 2.1. (a) Wireless networking with base station. (b) Ad hoc networking.

Our research is concentrated on the second mode, illustrated in Figure 2.1 (b). In this mode, there are no base stations, and communications goes directly from one device to another, in a peer-to-peer way, what is also called ad hoc networking. Differing from the former case, ad hoc network can be deployed without a central coordination or pre-established communication infrastructure. Because of this, ad hoc networks prove to be very useful in scenarios where the infrastructure was lost, is impossible to deploy or even is not cost effective.

Although the absence of base station can be good to cut off the necessity of communication infrastructure, it also introduces new problems, since the responsibilities of the base station must now be accomplished by the nodes themselves in a distributed and cooperative way. One of these challenges is the new way in which the information goes from origin to destination, which fits the study field of routing protocols. Another issue relates to address allocation and management among nodes, the central topic of this work. An overview of both challenges is presented in the next subsections.

2.1.2 Ad Hoc Routing

In ad hoc networks, the absence of a central coordinator point forces the nodes to cooperate in order to allow communications beyond the local antenna reachability. This procedure is known as multi-hopping technique in which nodes that are too far apart for direct communication use intermediate nodes to send the messages, in order to establish contact.

A simple way to implement multi-hoping is though flood messages. In such way, the message is marked with a sequence number and is sent to all neighbors though local broadcast. If the
receiving node is not the destination, then it forwards the message and keeps track of the sequence number, in order to avoid loops that happen when we re-send the same message. Figure 2.2 shows an example of multi-hoping communication using flooding messages where the originator is node A and destination is node G. The number on the arrows marks how many times the message was forwarded, and the direction indicates the sender and the receiver. The messages are propagated through most of the network, just to reach a two hop away destination. The necessity of a sequence number can be checked by looking to what happens when node D forwards the message to node E. Since node E has already registered the sequence number, it will not forward the message again. New messages are marked with different sequence numbers and the identification of the originator in order to keep track of what has been forwarded.

![Figure 2.2. Sending a message from A to G by flooding](image)

In the example above, six local broadcast messages were sent when just two unicast messages could do the work. Nodes B, C, D and E do not really need to forward the message. If the main goal is simply to reach a particular node in the network, the flooding process can be considered very costly, in other words, it entails too much network overhead.

To overcome this problem, ad hoc networks employ specific routing protocols that specify the path in which the message will be forwarded in order to reach its destination. There are many different types of routing protocols, but overall, they rely on information gathered from the network to determine the optimal path to the destination. Only the nodes that lie on this path must forward the message.

The specific characteristics of ad hoc networks, like dynamic topology and regular packet loss, make the creation and maintenance of optimal paths a great challenge. That makes routing an important research area in ad hoc networks. Two well-known examples of ad hoc routing protocols
are the Ad hoc On-demand Distance Vector (AODV) [PER03] and the Optimized Link State Routing (OLSR) [CLAU03]. AODV is a reactive protocol capable of discovering routes on-demand, that is, routes are discovered only when needed. OLSR, on the other hand, is a proactive routing protocol based on the classical link state algorithm and customized for the requirements of MANET.

The information gathered by these ad hoc routing protocols relies on some sort of address that identifies the nodes in a unique way, for this reason, they assume that mobile nodes are configured a priori with unique addresses before they enter a network. As with routing protocols, the MANET behavior creates challenges to address allocation, which are discussed in the next section.

2.1.3 Address Auto Configuration

A network address serves as a unique identifier for a computer on a network, or more precisely, it identifies a network interface card that belongs to a device, and locates it on the network. When set up correctly, the address allows a device to communicate with other devices connected to the network.

One of the best known forms of network addressing is the Internet Protocol version 4 (IPv4) address, followed by its successor, the IPv6 address. An IPv4 address consists of four bytes (32 bits) while the IPv6 consists of sixteen bytes (128 bits). In order to facilitate routing a data packet across multiple networks, the address is divided into network prefix and host identifier: The first one is something like a network identity represented by a contiguous group of high-order bits that are common among all hosts within a network. The host identifier is the remaining low-order bits of the address and specifies a particular communication interface in the local network.

Such addresses should be unique inside the network and can either be dynamically or statically configured. Static configuration means that some network operator assigns manually all addresses, which can be inappropriate in many situations, for example, in case of large networks. Dynamic configuration means that the network itself is capable of allocating addresses to the devices, what can be done by a central server or in a distributed way.

Traditional network, such as Ethernet or any other network that relies on a previous infrastructure often employs well known dynamic mechanisms such as DHCP [DROMS97], but for mobile ad hoc networks, automatic address configuration is more difficult due to instability of mobile nodes, low bandwidth of wireless links, openness of network, and lack of central administration.
Due to dynamic and unpredictable behavior of mobile ad hoc networks, the problem of addressing in such networks is generally handled in two steps. The first step is the generation of unique addresses and its allocation, which can be done in several distinct ways. The second step is the maintenance of the address space, which means to keep the uniqueness of the addresses over time and ensure that allocate addresses are still in use. Figure 2.3 shows a mobile ad hoc network scenario where a node D is joining the network and another node (F) is departing. In this simple case we already note that the network must be prepared to allocate addresses at any time to nodes that comes from anywhere. Moreover, addresses allocated to leaving nodes may get lost if they are not properly reclaimed.

The nodes departure can actually happens in three different forms that may affect the MANET in distinct ways. Firstly, a node can be in movement and lose contact with the network; in fact it may be only temporary out of reach and may not intent to permanently leave the network. Second, a node really wants to leave the network, in which case it may inform its desire. Finally, a node can run out of energy, get broken, or any other case in which it suddenly disappears from the network and it is not expected to come back. In the first and last cases, we cannot expect the nodes to inform its departure, since they were not expecting it. The difference is that in the former case, the node is expected to be back soon, whereas in the last case, also known as abrupt departure, it is better not to expect that the node will comes back. As we may conclude, if nothing is done to deal with node departures, networks can run out of free addresses.

Figure 2.3. Node joins (D) and node leaves (F) the Network.
Node mobility also introduces the processes of partitioning and merging. As we can see from Figure 2.4, a group of nodes can together move out of range of the main network, thus, forming partitions. In the example of Figure 2.4, node B, D and C move away, breaking the previous connected MANET into two different partitions. Partitions can potentially, at some time, come close enough to merge back, but it is also possible that they will never join back again.

The same problem with the address space that must be dealt with in node departure is also a concern in network partitions. One of the biggest problems about managing the address space is how to identify a lost address and also when it is the best time to recover it. After detecting a leak in the address space, if we act too early to repair it, we may introduce more network overhead, since the lost address owner could be back to the network and needs to request another address. On the other hand, if we delay the detection or recovering process too long, a new incoming node may not obtain an address, if the network has none available.

Figure 2.4. A network gets partitioned and then merges back.

The merging of partitioned networks is the most common type of merge but not the only one possible. The second case involves two completely different networks that have never been in contact to each another. In this case, there is a high probability that some address of one network is also present in the other one, since the address allocation processes were completely independent.

Lack of addresses, possibility of duplication, and network overhead, make the handling of partitions and mergers an important step in the development of an automatic address configuration protocol for mobile ad hoc networks. In the next section some of the current address auto configuration protocols are discussed with a brief categorization of them.
As we already know, the traditional addressing schemes for hardwired networks do not fit well in mobile ad hoc networks, because, among other things, nodes have not the characteristic and resource of servers, the connections are frequently broken, and there is no supporting infrastructure. In the past few years, the research on addressing for ad hoc networks has received much attention from academia, and various schemes for automatic address configuration have been proposed. These solutions try to allocate a unique address to mobile nodes in order to allow unicast communications though the network, and also try to be self-organizing and self-healing in an attempt to better adapt to the dynamic nature of MANETs.

Although keeping many similarities, derived from the common goal, addressing approaches for ad hoc networks differ in a number of aspects, mainly related to how addresses are generated and managed. Generally speaking, auto configuration protocols for ad hoc networks can be classified as stateless, stateful or hybrid approaches. The main features each group are described next, along with some works that can be seen as representative of each one.

2.2 Stateless Approaches

Stateless approaches are mainly recognized by not keeping any kind of information to construct a network view, or to know proactively the state of the address space. In other words, stateless address auto configuration protocols do not keep a list of free or used addresses. They are fully decentralized and use some self-configuration address allocation method in which a node can configure itself using a combination of locally available information. Examples of such information are its hardware address, a pre-assigned UUID (Universally Unique Identifier) [CHES04] [THO98], or even a random numbers [PER01]. For this reason, stateless approaches always make use of some mechanism for Duplicate Address Detection (DAD), which is needed to resolve address conflicts, and to try to ensure address uniqueness over the network. Next, we describe some of these stateless approaches.

2.2.1 Strong Duplication Address Detection (SDAD)

One of the simplest ways to allocate address in ad hoc networks is to let each node choose randomly its own address, and then, flood the network to detect if no other nodes are using the same address. This is the basis of the Strong Duplication Address Detection protocol presented in [PER01].

As the chosen address could introduce routing problems if it was used to send the queries for duplication detection, SDAD states that each node that wants to join the network, must randomly picks two addresses, a temporary address and a tentative address. The flow chart for SDAD is
presented in Figure 2.5. The temporary address is selected from the first 2047 addresses in the network 169.254.0.0/16, and is used only in the initialization phase as a source address of messages sent over the network to discover if the tentative address is already in use. The tentative address is the desired address, and is selected from the last 63487 addresses from the same 169.254.0.0/16 network.

The duplication detection works as follow: firstly, an Address Request (AREQ) contained the tentative address is sent from the starting node; then, it waits for an Address Reply (AREP), which is sent from configured nodes if the tentative address is already known to be in use. If an AREP is received, the starting node must choose another tentative address and send the request again. The AREQ message is sent a number of times to insure that the tentative address is free. If the response is negative, the node releases its temporary address and the tentative address becomes definitive.

![Flow chart of Strong DAD](image)

**Figure 2.5. Flow chart of Strong DAD.**

Despite its simplicity, we can list a number of problems and limitations about SDAD protocol. Even sending the AREQ messages for a certain number of times, the addresses duplication detection is enclosed in the initialization phase. If a configured node is temporarily
disconnected from the network, its address could be misunderstood as not in use. Moreover, partitions and mergers are not considered, which may lead to many undetected address duplication.

As we may notice, due to some limitations, Strong Duplication Address Detection protocol does not guaranteed the address uniqueness, and even if the above mentioned problems are not present, the solution efficiency decrease as network gets larger or in case of limited address space.

2.2.2 Weak Duplicate Address Detection (WDAD)

In contrast with the SDAD protocol, WDAD [VAI02] extends the address duplication to whole lifetime of the network. In this protocol, each node maintains on its routing layer a virtual address that consists of an identification key and its IP address. In a communication, a node is able to send packets to the destination node even if destination node's address is also being used by another node. This is possible because each node uses its identification key for differentiating between potential duplicate IPs.

Each node generates the identification key at initialization phase, and distributes it in all routing messages. IP addresses and keys are maintained on its routing table. If a node receives a routing message with an IP address that exists in its table, it checks if the keys are different. If the keys are different, then address duplication is detected. The node then marks the entry as invalid and informs other nodes about this duplication.

WDAD has the disadvantage of depending on the routing protocol. It requires some modifications on the routing layer to support the introduction of the key identity. This technique can be combined with proactive or reactive routing. However, as this protocol detects address duplication based on local routing information, it is totally dependent on proactive routing where each node maintains a complete routing table.

This protocol does not generate any additional traffic for the network auto configuration, but the use of the key value in routing packets increases the traffic overhead.

2.2.3 Passive Duplicate Address Detection (PDAD)

PDAD [WEN03] detects the address duplication investigating routing information and deducing the problem from events that never occur in case of unique addresses but do occur if there are address duplicates. It is applicable only for proactive routing protocols. In scenarios of proactive routing, each node propagates messages about its neighborhood through the network. These control messages contain sequence numbers to distinguish fresh and old packets. PDAD bases on the
sequence number information to detect address duplicate. Sequence numbers are increased with each packet, and a reset occurs once in a long period of time. If a node receives a message with its IP address as the source address and a sequence number is greater than its own counter value, an address duplication conflict is detected.

The advantage of this protocol is that it does not generate any additional overhead. However, in addition to be applicable only for proactive routing it requires complex analysis of the routing information, in order to detect the addresses duplication.

2.2.4 Ad Hoc IP Address Autoconfiguration

Ad Hoc IP Address Autoconfiguration [JEON05] combines the mechanisms of SDAD and WDAD. Thus this mechanism checks not only if there is address duplication during initialization, but also checks and resolves potential address duplication detected by intermediate nodes using routing messages. Each node must obtain a 128 bits long key and utilizes it to control packets of routing protocol. Intermediate nodes must maintain the key value for each address in routing table or cache.

When receiving a routing packet, a node compares all IP addresses and key values contained in that packet to IP addresses and key values contained in routing table or cache. If the node finds different key values for the same IP address, then an address conflict has occurred. Then the node unicasts an address error message, informing that a conflict has occurred to the node with duplicated address associated with smaller key value. When the message reaches the node with address conflict, then the node releases its address and reinitializes the process of autoconfiguration to obtain a new IP address.

The fusion of the mechanisms SDAD and WDAD allows for smooth handling of network partition and merging. But as a stateless protocol, it still introduces a lot of overhead during the initial stage of address allocation, and also needs changes in the underlying routing protocol.

2.3 Stateful Approaches

Stateful approaches are the ones that maintain information, such as address allocation tables to track assigned and free addresses, so existing nodes can assign unused addresses to requesting nodes. The major motivation for the development of stateful solutions is to guarantee uniqueness of the address, and this is also the greater advantage of stateful approaches over the stateless ones. By ensuring that only free addresses are assigned, stateful solutions do not need to make use of duplication address detection procedures, thus, avoiding the network overhead that comes with them.
The challenge for stateful approaches is to maintain the consistence of the address space. Some sort of information exchange may be needed to synchronize the state of nodes, so they know the list of assigned addresses, or if the previously assigned addresses are still in use. If not carefully planned, the introduced control overhead may not be worth, even considering the efficiency with regard to address conflicts.

The way that the approaches use the address space to allocate the addresses and manage it can be seen as a way to categorize such solutions. If a node is presented as single centralized structure that maintains all information regard addresses allocation, we can say that this is a centralized allocation table solution. On the other hand, if this allocation table is kept by all nodes in a shared way, it is the case of approaches called distributed allocation table solutions. The two other categorizations are distributed disjoint tables and distributed allocation function. Solutions that employ distributed disjoint tables differ from the two previous approaches because there is not one table only; there are many different tables that are mutually exclusive. To finish, solutions based on distributed allocation functions are very distinct from the other approaches due to the fact that there are no allocation tables.

Centralized allocation table solutions fit better in fixed networks since in MANETs it is often difficult to maintain a single centralized structure. In this approach a server acts as a centralized addresses pool. When a node requires an address, it then sends a request and the server replies with a message which contains the address that was assigned to the requesting node. The Agent Based Addressing [GUN02] is one of the few solutions that make use of a centralized allocation table.

The major drawback of centralized allocation table becomes noticeable when the connection with the main node get lost, which means that no address allocation can take part, until this main node comes back or a new one is selected. This problem motivated the development of distributed allocation table solutions, which let all nodes have the same allocation table, and control it consistency by distribute the maintenance of it. MANETconf [NES02] appear as one of such approaches.

Although eliminate the needed for a central structure, distributed allocation table solutions, such as MANETconf generates much control traffic overhead while try to distributed maintain the table consistency. Distributed disjoint table solutions avoid such traffic by using distinct tables for each node. One way to achieve this is to split the address pool of the initiator in half and offer one half to the requesting node, a method that was applied in Buddy [MOH02].

Distributed allocation functions, as we have said, are used to avoid the use of tables, and in the meantime, the traffic and resources needs to maintain then. Such functions are applied to
stateful address allocation mechanisms in an attempt to split the original address space into several distinct subsets and provide them to the nodes. Every configured network node implements the function and uses it to allocate addresses to new nodes. It is not a trivial task to develop a distributed generation function that is conflict free. In other words, a function that allows each node to only generate disjoint set of address. Prophet [ZHOU03] and Prime [HSU05] are two good examples of solutions that apply such functions.

After this general overview of stateful approaches to address allocation, including the categorization and general behavior, we present some details of each of the above mentioned solutions, which we believe are representative for a general understanding.

2.3.1 Agent Based Addressing

Agent Based Addressing [GUN02] consists of a centralized address auto-configuration approach. Only one node in the network is responsible for assigning addresses. This node, known as Address Agent (AA), periodically floods the called Verify packets that hold the address list already assigned in the network. The other nodes when receiving a Verify packet send a Confirm packet indicating that they are present and that is used to refresh the address entry lifetime.

A node that wishes to join to the network should wait a certain time for a Verify packet before requesting an address. The new node requests its address in unicast to the AA. When receiving the request, the AA builds a new 80 bits long IP address based on its MAC address and the requesting node's MAC address. Then the IP address is sent to the requesting node that configures its interface and becomes able to communicate with other nodes in the network.

There is a mechanism to elect dynamically a new AA in case of AA departure. If a node does not receive a “Verify” packet, it concludes that there is no AA in the network and auto considers as the new AA. Each network has a Network ID for differentiating from other networks. This identification is derived from AA's MAC address and is sent with the Verify packets.

A merge between networks is detected when the AAs notice the presence of each other by the reception of the Verify flood. The AA with fewer entries in its table should change its state and concerned nodes should request the new AA for IP addresses with leads to unnecessary address changes.

Despite the assurance of addresses uniqueness that eliminates duplication detection, the Agent Based Addressing presents a greater limitation due to its centralized behavior. The protocol concentrates many functions in a single node for a MANET. Moreover the periodic flood of
“Verify” messages, followed by the nodes answers to them, generates a high overhead, while the list of configured nodes presented in such messages is note used.

2.3.2 MANETconf

The idea introduced by MANETconf [NES02] is to reduce the problems derived from a centralized solution by distributing the addresses allocation between network nodes. This is achieved by using an allocation table that is maintained in a common and distributed way. The synchronization of this allocation constitutes the most important and complex task of this solution.

When a node wishes to join the network, it sends a message to its neighborhood. The first node that replies to the message is chosen by new node to generate its possible address. The new node then requests an address to the indicated node. The last one chooses a free address in its allocation table and floods it through the whole network to have the permission to use it. If all nodes in the network reply positively, it concludes the address is free, informs it to the requester and floods it through the network to confirm the address assignment and let all nodes update their tables. If one or more nodes reply negatively, it concludes the address is already in use and repeats the procedure since the beginning, a certain number of times. If one or more nodes did not reply, it tries a new contact by unicast requesting permission. If there is no answer, it concludes after many attempts that the node left the network and informs the whole network about the departure. If the node replies, the configuration process continues.

Each network has an identifier that consists of two elements: the lowest IP address and one unique number generated by the node with the lowest IP address. When a partition occurs in the network, one of two networks maintains the original identifier and the other one will have to generate its new identifier. On this last case, a new node with the lowest IP address will have to be chosen for generating the network ID and sent it over the network.

When two or more nodes are within the communication range each other exchange their network identifies. If a node receives an identifier that differs the nodes identifies, a merge is detected. When this occurs, these networks exchange their different allocation tables allowing all nodes update their allocation table and detect locally address duplication. When address duplication is detected, one of the two conflicting nodes should generate its new address.

MANETconf finds a way to handler network merges that does not generates unnecessary address changes, as only nodes involved in duplication release their IP addresses. The address allocation is totally distributed and the uniqueness is ensured. However, the maintenance of the distributed allocation table is a complex process and demand high bandwidth. As the final
configuration needs the permission of all network nodes, the time for a new node to assign a free address can be very large.

### 2.3.3 Buddy

In Buddy [MOH02], each node maintains an allocation table that holds a part of the whole address space used to assign addresses to new nodes. At the same time, each node uses the whole allocation table to control the network evolution. The nodes exchanging messages to build their allocation tables and maintain them updated.

The first node in the network detects no neighbors and then auto assigns the first IP of the predefined address range. When a new node wishes to join to the network it sends a broadcast message requesting an IP address. If there is no response, the node then assigns itself the first IP of the predefined address range. If the node gets one or more responses, it reclaims an address from first node who replies. The requested initiator divides its address pool and sends half to the requester node associated to the copy of its address table. Then, the new node assigns the first IP of the received address pool, and sends a confirmation message to its initiator.

If the initiator has no addresses to assign, it then requests its neighbors. It first searches its address table if there is one hop neighbor with free IP address. If there is no response, it repeats the process searching for IP address among two hop neighbors and so on. If the search is not successful, it contacts the node in its address table with the biggest block.

The synchronization of the address table occurs when each node broadcasts its address table. The detection of address leaks is accomplished by “buddy nodes”, i.e. nodes that test each other if they are active. If a node detects the other is missing, it merges its IP range with its own pool. For distinguishing between different networks the first node in a network generates a network ID.

The address allocation in Buddy is less sensitive to network losses, as the control messages involved in such process are limited to neighborhood of the starting node. Moreover, the management of address space allows it to operate in scenarios in which such space is more restrictive in terms of max number of node allowed. On the other hand, the effective of the solutions depends on network concentration and there is no process to balancing the number of available free or reclaimed addresses among the nodes. Furthermore, the synchronization process may need too much time to converge, and it is a little complex.

### 2.3.4 Prophet

Prophet [ZHOU03] is a solution that uses an address allocation method capable to generate with a high probability a unique sequence of addresses without maintaining an allocation table. It
supports partition and merges operations through an advanced address management mechanism. This solution was named Prophet Allocation because the very first node of the network is assumed to know in advance which addresses are going to be allocated.

The base of the solution is the employs of a stateful algorithm in which a sequence consisting of numbers is obtained in a range $R$ through a function $f(n)$. The initial state of $f(n)$ is called a seed, where each seed leads to a different sequence with the state of $f(n)$ updated at the same time. When a new node wishes to join the network, it sends a local broadcast to its neighbors. If it receives no reply, it concludes that it’s the only node in the network and generates a new seed, which is used as the initial function state and to build and configure the IP address. When new nodes broadcast the requisition, the function generates new pair of seed and status. The seed is offered to the new nodes and the state update the function to prevent it to generate the same sequence again. The received seed is used in the new node to configure its address and as the initial state of the function, and from this moment, both nodes can assign IP addresses.

Prophet state that the sequences of $f(n)$ satisfy the following two properties, when considering large enough address spaces

1. The interval between two occurrences of the same number in a sequence is extremely long;
2. The probability of more than one occurrence of the same number in a limited number of different sequences initiated by different seeds during some interval is extremely low.

The algorithm is illustrated as an example in Figure 2.6. Suppose every node is represented by a 2-tuple: (address, state of $f(n)$). Here the range of addresses is [1,8], and $f(n)$ is (address×state×11)
A is the first node in the MANET and uses a random number of 4 as its IP address and seed. When node B joins, node A gets $1 = (4 \times 4 \times 11) \mod 7$. Node A changes its state of $f(n)$ to 1 and assigns 1 to B. When C approaches A and D approaches B, they receive $2 = (4 \times 1 \times 11) \mod 7$ and $4 = (1 \times 1 \times 11) \mod 7$ from A and B, respectively. The reason of conflict with node D and C is due to a small range of available addresses.

Regarding network disconnection and merger, the authors state that Prophet Allocation can easily handle both situations. When a network gets partitioned, the sequences of each network are different and the new allocated addresses are still different among the partitions, moreover no address conflict will occur if the networks merge again.

Nonetheless, for the merger between two or more distinct networks, the Prophet Allocation implements the concept a Network ID (NID). The NID is randomly generated by the very first node in the network and is supposed to uniquely identify this network. If the NID is large enough, two networks will rarely have the same identification. This ID is known to the network during the address allocation process, and the merging can be detected by analyzing routing protocol Hello Messages, which could be signed with this number.

The function proposed in Prophet generates IP addresses with very long repetition intervals when considering larger range of valid addresses and the probability of occurrence of the same number in different sequences (with different seeds) is very low. For this reason, Prophet does not present any mechanism for detection of duplicated addresses. The Prophet approach generates almost no extra traffic and is very simple to be implemented.

Despite its benefits, the Prophet does not provide analytic proof that the described function fulfills properties stated and is clearly only applicable for large address spaces. Furthermore, there is a lack of explanation in the work with regards to what happens after a merger. Additionally, the address space is not very well spent.

### 2.3.5 Prime DHCP

In Prime DHCP [HSU05] each host operates as a DHCP proxy that runs an algorithm for generating a unique address in MANET and thus DAD is not necessary during the process of address resolution. All hosts are eligible to assign addresses and a new host can acquire an address simply from its neighbors, so that allocation happens without broadcasting over the whole network.

The Prime DHCP ensures precisely a unique sequence of addresses for each node. The method stems from the mathematical definition that every positive number can be written as a product of prime numbers in a unique way. The solution states that the first node that creates a network is the root node and has address 1. The root node allocates all prime numbers in sequence
to new nodes. Non-root nodes allocate addresses as their own address multiplied by a prime number which starts from the biggest prime factor of its own address.

When a new node joins a MANET it issues a DHCP_Discover broadcast message. Neighbor hosts generate an address each one and then encapsulate it in a DHCP_Offer message. The new node chooses the smallest address and broadcasts its choice in a DHCP_Request. The chosen proxy updates its address allocation status and then sends a DHCP_Ack to new host for confirmation. Consequently, instead of whole-MANET broadcasts, DHCP_Offer and DHCP_Request incur just a single-hop broadcast.

When a node leaves the network, it should send a DHCP_Release message to its parent proxy. The parent proxy then records the allocation status for its departed child. Each host maintains a recycle list that records the allocation statuses for its departed children in its allocation status. When a parent proxy receives a DHCP_Discover from a new node, it will first give out the lowest recycled address and inform the new node the allocation status associated with the offered address.

For a host that leaves the network gracefulness, its address may become not usable. In order to reclaim leaked addresses, the root proxy periodically broadcasts a DHCP_Recycle message to ask all hosts report their address allocation statuses, including addresses that they have given out or recycled. After gathering the hosts allocation statuses, the root proxy reconfigure the address allocation tree and inform each proxy of updated allocation status.

For detecting network merging and partition, the recycle process can also be used periodically. When a merging of two MANETs happens, a root can receive allocation statuses from hosts in the other network and then detect address conflicts. In this case, the root asks one of the two conflicting nodes to request a new address.

When a network partition happens, the split MANET needs a new root for address recycle. If a node misses several recycle messages, it may claim to be the new root by sending a DHCP_Recycle after a backoff time reversely proportional to its address. This can become the node with the largest address the new root increasing address utilization.

The PNAA introduces a simple and efficient number generator, that ensures the uniqueness of addresses but the address space is not efficient as it is not very well distributed among nodes. Network partition and merge operations is supported but it not very efficient. Moreover, the concentrating of important functions in a single node can prove to be fragile due to dynamic and unpredictable behavior of MANETs.

2.4 Hybrid Approaches

Hybrid approaches, as the name suggests, combine both stateless and stateful techniques in order to improve the overall addressing performance. In general, hybrid solutions tend to use one or
more tables to keep track of what is going on the network and also make use of some sort of
duplicate address detection. In they try to be as efficient as possible, these kind of solutions also
prove to be the more complex ones. We describe next the HCQA [SUN03], and PACMAN
[WEN05], two representative solutions of hybrid approach.

2.4.1 Hybrid Centralized Query-based Autoconfiguration (HCQA)

HCQA [SUN03] combines characteristics of the stateless mechanism SDAD along with a
centralized address autoconfiguration approach. A node that wishes to join to the network waits for
a advertisement of the node “Address Authority” (AA) a certain period of time. When receiving the
advertisement, the new node sends an address registration request to the AA and waits for a
confirmation (ACK message). If the node receives a confirmation message, it may begin to use the
tentative address. After its registration with the AA, the new node runs a timer and reinitializes the
process each time the timer expires.

The AA performs a very important role in detecting address duplication. This is done in
initialization phase when a new node tries to register its tentative address with the AA.

In network initialization, the first node that gets an IP address becomes the AA in the
network. The AA chooses an unique identifier (ex. its MAC address) and broadcasts it periodically
to identify the network. If a node does not hear any advertisement from the AA for a certain time, it
considers that a network partitioning has happened and becomes the new AA and generates a new
unique identifier.

When a node hears a new network ID, it must register its address with the new AA; in this
case, no address change is needed. When the presence of two network IDs is verified, a network
merge is detected. Since AAs hold the state of all assigned IP addresses, the AAs involved in the
verified merge exchange their tables for detecting possible address conflicts.

The protocol performs backup of the address authority's address table to reduce the
centralization at the AA. The AA considers the first node that has registered its address as the
“Address Authority Backup”. When a new node registers its address with the AA, the AA also
sends an update containing the new information to the address authority backup.

The implemented conflict detection mechanism by exchanging AA information generates high
control messages traffic in the network. Moreover, this approach is dependent on a central entity
which may increase the autoconfiguration delay and makes the solution more sensibility on messages
losses.

2.4.2 PACMAN

PACMAN [WEN05] stands for Passive Autoconfiguration for Mobile Ad Hoc Networks and
is an approach for distributed address autoconfiguration that generates a very low protocol overhead by using cross-layer information derived from ongoing routing protocol traffic. It uses elements of both the stateful and the stateless approach. As in stateful approaches, the nodes utilize the routing protocol information to know which addresses assigned in the network. Address conflicts are detected passively based on anomalies in routing protocol traffic. As in stateless approaches, each node assigns itself an IP address enabling the address compression, which contributes for the reduction of routing protocol overhead.

PACMAN supports network partitioning and merging by using a Passive Duplicate Address Detection (PDAD) method. Using this technique, a node observes the incoming routing protocol traffic and checks if there is address conflict, not consuming for that any additional bandwidth.

While greatly reduces the amount of additional traffic to allocate addresses and the probability of duplication by using cross-layer information, PACMAN is a complex solution to be implemented and relies on changing of the underlay protocols.

2.5 Comments

In this chapter we study the challenges presented in address allocation for mobile ad hoc networks, and review the existing approaches for automatic configuration of addresses. These solutions, however, do not cover well all situations that may happen with the dynamic and unexpected behaviors of MANETs or introduces too much network overhead. For example, solutions based on Strong DAD mechanism are not capable to ensure uniqueness and introduces too much overhead. Passive and Weak DAD solutions need changes in the underlay protocols. Even the Prophet and Prime solutions that have some similarities with the proposed protocol have important drawbacks. While Prophet does not really ensure address uniqueness and is only applicable to large scale MANETs, the Prime are very dependent to its root server and the mechanism to handler partitions and mergers are not very effective.

The Table 2.1 bellow presents a brief and comprehensive overview of the previously presented related work, indicating the classification, uniqueness of the address allocation procedure, the necessity of duplication address detection, and also the latency to obtain an address. In Table 2.1, the period of synchronization, flood, or any repetitive procedure if exists is indicated by $T$. $d$ is the average network radius in number of hops. $t$ is the round trip time for one hop communication. $k$ is the number of iteration if exists. The average one-hop latency is $l$. $b$, $D$, and $s$ are the average hop number, the network diameter, and the retry time respectively.
<table>
<thead>
<tr>
<th>Protocol</th>
<th>Approach</th>
<th>Address uniqueness</th>
<th>DAD</th>
<th>Latency</th>
</tr>
</thead>
<tbody>
<tr>
<td>Agent Based Addressing</td>
<td>Stateful</td>
<td>Guaranteed</td>
<td>No</td>
<td>T/2 + d*t/2</td>
</tr>
<tr>
<td>MANETconf.</td>
<td>Stateful</td>
<td>Guaranteed</td>
<td>No</td>
<td>(2 + d)*t</td>
</tr>
<tr>
<td>Prophet</td>
<td>Stateful</td>
<td>Not guaranteed</td>
<td>No</td>
<td>2*t</td>
</tr>
<tr>
<td>Buddy</td>
<td>Stateful</td>
<td>Guaranteed</td>
<td>No</td>
<td>2*t</td>
</tr>
<tr>
<td>SDAD</td>
<td>Stateless</td>
<td>Not guaranteed</td>
<td>Yes</td>
<td>k*T</td>
</tr>
<tr>
<td>WDAD</td>
<td>Stateless</td>
<td>Guaranteed</td>
<td>Yes</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td></td>
<td>with high probability</td>
<td></td>
<td></td>
</tr>
<tr>
<td>PDAD</td>
<td>Stateless</td>
<td>Guaranteed</td>
<td>Yes</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td></td>
<td>with high probability</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Ad hoc IP Autoconf.</td>
<td>Stateless</td>
<td>Guaranteed</td>
<td>Yes</td>
<td>k*T</td>
</tr>
<tr>
<td></td>
<td></td>
<td>with high probability</td>
<td></td>
<td></td>
</tr>
<tr>
<td>HCQA</td>
<td>Hybrid</td>
<td>Guaranteed</td>
<td>Yes</td>
<td>k*T</td>
</tr>
<tr>
<td>Prime DHCP</td>
<td>Stateful</td>
<td>Guaranteed</td>
<td>No</td>
<td>O(2*l)</td>
</tr>
<tr>
<td>PACMAN</td>
<td>Hybrid</td>
<td>Guaranteed</td>
<td>Yes</td>
<td>O(2<em>l</em>D*s)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>with high probability</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Table 2.1. Related works summary.
The objective of this chapter is to propose a new protocol for automatic address organization named Dynamic Node Configuration Protocol (DNCP). The chapter starts with an overview of the solution, including its classification and general behavior. Next, the solution and all its components are described in detail. The control messages in use and its formats are also presented afterward. Finally, we discuss the strengths and weakness of the mechanism and comment possible extensions and improvements.

3.1 Overview

As mentioned before, the problem of address allocation and maintenance in mobile ad hoc networks is not a trivial task, and current solutions needs to evolve to be more performance efficient or to not be much specific, covering small scenarios and forgetting to take care of all addressing challenges of mobile ad hoc world. For this reason we developed a new protocol named DNCP, which is a stateful approach to automatic address configuration that employs a distributed generation function and also some other techniques to overcome the challenges inherited from dynamic behavior of MANET.

To better understanding of the characteristics and behavior of DNCP, we detached this one upon two main parts: address allocation and address maintenance. Together, these parts represent the two major DNCP’s goals, which are: allocate a unique identifier for each node contained in a network, and preserve the consistence of address space during network lifetime.

The address allocation is performed through the exchange of a set of control messages and the use of a distributed function, later defined as Binary Number Generation (BNG) function. The function, based on principles of positional notation system, ensures the generation of unique identifiers; whereon eliminate the necessity of duplicate address detection procedures. Messages are mainly exchanged with direct reachable neighbors, reducing the initial overhead and latency to retrieve an address.

When a node starts up, initially, it should attempt to communicate with reachable nodes to retrieve a unique address. The method of address retrieval is accomplished with the assistance of
those neighbors, responsible for answering or forwarding any address request received from any bootstrapping node. These addresses are constructed at neighborhood nodes that use their internal function (BNG) to generate a unique identifier for each address request. Thus, the neighborhood nodes could be assumed as a set of servers sharing an address space to any bootstrap node requiring an address and in range.

The next step involves the configuration of the bootstrap node, which corresponds to the use of the unique address constructed and transmitted by a neighbor node. After properly configured, a node starts to act as a server, sharing an address space for new nodes, and also participate at addresses maintenance process. Preserve the consistence of address space over time means that the networks must deal with abrupt node departure, temporary disconnections, partitions, and merge. All these events can affect the efficiency of the network address organization, mainly due lost of addresses and possibility of introducing addresses conflicts. DNCP uses a mechanism called address reclamation to deal with loss of address spaces and also other mechanism to deal with network partition and merges, in order to avoid address conflicts.

3.2 Address Allocation

The general address allocation process is fairly simple. As stated in section 2.1.3, the address is composed of a network prefix and a host identifier. The network prefix is a previous statically defined value, while host’s identifiers are calculated with the distributed generation function (BNG). Actually, the solution takes the network prefix 10.0.0.0/8 as the default value, but it is just a value that can be changed. The basic behavior for dynamic address allocation is illustrated in Figure 3.1, where a new node is joining a network composed of three previously configured nodes. The behaviour is very similar to what happens with DHCP When a new node is bootstrapping, it broadcasts a Server Discovery message to ask for an address, and should wait for a response. Upon receiving a Server Discovery message, each previously configured node in range may answer with an Address Offer message. Therefore, the new node might receive multiple offers and may choose the best address suggestion to use. After determined the offer to use, the node must send an Address Request message to the sender of the offer. The chosen server after receive the request should send an Acknowledgment message to the new node. Upon reception of the Acknowledgment message, the new node can configure itself and also starts to acting as an address server.
Actually, not every reachable server may answer to Server Discovery messages. At some time, server may become depleted of addresses to offer, which may often happen due to address space limitation. In this case, the neighbors must not respond to Server Discovery messages, and whether all neighbors of a requesting node are in such situation, no Address Offer message will be received at all. To overcome this problem, DNCP includes also a process to find remote servers with addresses to share, thus address requisition may transpose the neighborhood boundaries.

### 3.2.1 Binary Number Generation Function

The problem of address allocation can be compared with the problem of how to assign different numbers from an integer range to different nodes. The purpose of number generation functions, or distributed allocation functions, is to generate different outputs series from different inputs values called seeds. More specifically, in stateful solutions for automatic address organization that use distributed allocation functions, every node implement the function and should generate only disjoint set of numbers based only on a received seed number, which in general is its own address.

DNCP uses a conflict free distributed generation function called BNG (Binary Number Generation) [ASCH09] that was elaborated based on characteristics of the positional notation system. In such numeral system, a number is represented by an ordered set of characters or symbols where the value of a character depends on its position. The number of different symbols to use, which is the base of the system, is called radix. The binary number system (radix equals to two) is probably our best-known example of positional notation system.

In positional notation systems, the change at one position in the symbols sequence that represents a number is sufficient to change the value of the same. Thus, there are no two distinct sequences that have the same value. This property is useful for the construction of a distributed
generation function, since it can ensure that a particular subsequence is generated exclusively, ensuring uniqueness of addresses. More information about position notation systems and definitions can be found in Appendix A.

Although this property can be used to build functions based on any positional notation system, the binary system greatly simplifies the final result, due the existence of only two symbols and to be the system in use at ordinary computers. Those facts led the selection of the binary system as the base for the distributed function. Generally speaking, the function works generating numbers from an internal binary sequence. The sequence has some positions marked as unchangeable, and the differential in this approach is to change the immediate bit of the remaining positions and expand the number of unchangeable positions, including the immediate bit position for the input and output. In such manner, the internal sequence and the generated one will have the same amount of unchangeable bits, however they will differ in at least one position, which makes the sequence value different and also ensures that these two sequences never generate the same number.

Before introduce the mathematical definition of BNG function, an in-deeper explanation of its behavior and conflict free characteristic is required. Figure 3.2 illustrates the BNG behavior through three examples. The first two of them use the same input binary sequence (seed), differing only on the amount of unchangeable bits (status), and the last one uses the output from the first example as input.

For the first example (Figure 3.2.a), the input seed is the binary sequence “000001” (one), and the status is 2 (two). At the figure, the status is also represented by the highlighted region on grey that indicates the amount of unchangeable bits and the remains ones.

To generate a new sequence, the function split the original seed into two subsequences: the right side, formed by highlight region; and the left side, formed by the remains bits. Next, the left seed part is increased by one unit and attached afterward with the right part to form the generated sequence. Finally the status is updated, increasing it by one unit. The new status and the generated sequence compose the final function output.
Figure 3.2: BNG behavior

Figure 3.2.b shows the function behavior for that new status generated at last example and the same initial seed used before, while Figure 3.2.c shows the function behavior using the output presented in Figure 3.2.a. As noticed, the right seed part from second example (binary sequence “001”) is not the same of the right seed part from third example (binary sequence “101”), since these parts are used to compose the generated sequence, and is well-known that one single different symbol changes the numeric value of the sequences, thus these two seeds, with the assistance of the status variable, will never generate the same number or sequence.

A seed, its generated sequences, and also the sequences produced by these first generated sequences, at any level, do not produce the same number. This is due to the amount of unchangeable bit that is different from generator seed and for generated seed which is also carried along any levels of dependence. We can check this behavior looking at the output presented in Figure 3.2.c. The highlighted area of the output (binary sequence “1101”), contains the same bits that ensures generated sequences to be different from the ones provided by seed “000001”.

Another important observation about the mechanism is the ability to generate all integer numbers from a given interval. This is due to the fact that when the seed is separated, the left part is incremented by just one unit, and since we are dealing with only two symbols (0, 1), it is enough to ensure that every possible combination be achieved. As an example, Figure 3.3 shows the set of
generated numbers (numeric value of the seeds) and its dependencies when considering sequences of five bits, since we not consider 0 as a valid identifier, this mean an integer range from 1 to 31.

The first seed must be self-generated and use the lowest valid identifier, which mean that the mechanism starts with seed 01 and status zero. If we apply the behavior presented in Figure 3.2 to seed 01, we conclude that the first set of generated numbers will be 02, 03, 05, 09, and 17. The next generated number in sequence would be 33, but it cannot be represented with five bits, or in other words, it is out of the valid identifiers range.

![Tree of identifiers](image)

**Figure 3.3: Tree of identifiers.**

The mathematical representation of the proposed function is presented at Erro! Fonte de referência não encontrada., whereon ‘ ’ is the function seed, and ‘ ’ is the function status.

(3.1)

The function receives a seed and a status as its parameters and output one new seed and one new status. To generate new identifiers, we just add up the seed with a power of two, using the status as the exponent. The starter status, which must be used with this new seed, is also provided, and is equal to the input status plus one. As we see in Figure 3.2, although we do not need to change the input seed, the input status is always updated, in order to generate new numbers.

(3.2)
The starting status value to use with a specific seed can be easily obtained from \textbf{Erro! Fonte de referência não encontrada.}, but it can also be acquired from the seed itself, if needed. The Equation 3.2 shows that the initial status ‘‘’ can be computed as the minimum value of a natural number that used as the exponent of a power of two results in a value greater than or equal to seed ‘‘’. The Table 3.1 shows some seed values and its correspondent initial status calculated with Equation 3.2. As we already known from Figure 3.2, the initial status for seed 5 (binary sequence “010”) is 3, which is also verified in Table 3.1. The same is truth to seed 13 (binary sequence “1101”), also presented in Figure 3.2 with the initial status 4.

<table>
<thead>
<tr>
<th>Seed</th>
<th>0</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>Init Status</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1</td>
<td>2</td>
<td>4</td>
<td>8</td>
<td>16</td>
<td>0</td>
</tr>
<tr>
<td>2</td>
<td>1</td>
<td>2</td>
<td>4</td>
<td>8</td>
<td>16</td>
<td>1</td>
</tr>
<tr>
<td>3</td>
<td>1</td>
<td>2</td>
<td>4</td>
<td>8</td>
<td>16</td>
<td>2</td>
</tr>
<tr>
<td>4</td>
<td>1</td>
<td>2</td>
<td>4</td>
<td>8</td>
<td>16</td>
<td>2</td>
</tr>
<tr>
<td>5</td>
<td>1</td>
<td>2</td>
<td>4</td>
<td>8</td>
<td>16</td>
<td>3</td>
</tr>
<tr>
<td>13</td>
<td>1</td>
<td>2</td>
<td>4</td>
<td>8</td>
<td>16</td>
<td>4</td>
</tr>
</tbody>
</table>

Table 3.1. Some examples of seeds and their initial status.

To conclude this explanation about BNG, Figure 3.4 presents a sequence of illustrations showing the use of an embodiment of the function to allocate a configuration (seed, status) to a number of nodes.
3.2.2 Local Addresses Request

The local address allocation process is the way to exchange seeds and status values from BNG function among neighbors’ network nodes, and obviously configure their addresses. To provide a more detailed description of how nodes behave during local address allocation, a flow chart is presented in Figure 3.4, containing the actions performed by clients and servers respectively. For simplicity, all processes related to remote allocation are not presented in these flowcharts.

![Figure 3.4: Sequence of generated seed and the correspondent status.](image)

![Figure 3.5: Flow chart of local address allocation.](image)

At the client side, the process starts with the transmission of a broadcasted Server Discovery message, a very simple message that contains only its own type, and later on the client then should wait for replies. Configured nodes that receive the Server Discovery message call the BNG function to generate a new seed and status. If the generated seed is a valid identifier, an Address Offer message is sent direct to the starting node, containing the generated seed and status. The Address Offers received at the client are buffered until a timeout expires, then, it should choose the offer with the lower seed and send an Address Request message to the server that sent the offer with the chosen seed and status. Upon reception of an Address Request message, the server utilizes the BNG function again and compares the output with the seed and status presented in the received message. Whether the value matches, the server updates its internal BNG status and sends an Acknowledgment message with the other configuration parameters, like the network prefix, the
amount of free bits that can be use to generate the host identifiers, and the network ID, which will has it function presented later. Such update in internal status is what guarantees the uniqueness of generated sequences. When the client receives such Acknowledgment it uses the network prefix plus the received seed to construct the address and configure its network interface.

Since errors may occur, the Server Discovery message should be sent a number of times until a node gives up. Moreover, if a server was discovered and its Acknowledge message gets lost, the retry time must be reset, and the entire process must be started again. The retry time must be chose carefully because low values may prevent nodes to join a network while high values may lead to a longer configuration time, in case where the neighbors really do not have anything to offer.

Figure 3.6 shows an example of the described process, cited above, for three network nodes: node A is the network creator and the others (B and C) are requesting for address. As we already known, the first network node is configured with seed 1 and status 0, and chose the network prefix (refer to section 3.2.3 to an explanation about how first node is detected). When the node C starts, it sends the Server Discovery message that only reaches A. Node A answers with seed 2 and status 1 (remember Equation 3.1). Since no other offers have arrived, node C has chose node A as it server and sends to it an Address Request message. Node A sends the Acknowledgment message and updates the BNG status to 1. Finally, node C configures itself with seed 2 and status 1. When node B starts up, both node A and C were already configured and in direct communication range, which means that two Address Offer messages may be received. From node A, B has received an offer with seed 3 and status 2, and from node C, it has received seed 4 and status 2. As the seed from node A is smaller, node B chooses node A as it server and the process continues as expected.
3.2.3 Remote Address Request

The remote allocation process comes into play when neighborhood nodes do not have address to offers, meaning that other server nodes may be located elsewhere. Nodes that do not have address to offer may act as a Proxy, forwarding the messages from the new node to the rest of the network and vice versa. The process of remote allocation is shown in Figure 3.6.

After local address allocation process failed, the node may send a Proxy Request message in broadcast mode to all its neighbors. The neighbor can replies back a message Address Offer, in case it still has a valid address, or broadcast a Remote Server Discovery message to its own neighbors. Upon receiving the Remote Server Discovery message, each node can send a Remote Address Offer to the proxy or rebroadcast the Remote Server Discovery message, in case it does not have addresses to offer. The proxy forwards the Remote Address Offer messages received directly to the new node, which chooses the lowest seed and sends a Remote Address Request to the proxy. The proxy forwards the request to the server node, which replies with a Remote Address Offer. Finally the proxy forwards the offer to the new node that can configure itself.

In order to allow servers to locate the proxies, the Remote Server Discovery message carries the originator proxy address. With the proxy address, the servers with available address can use any present routing protocol to reaches the proxy. To avoid loops, as well as duplicate responses, Proxy Request message includes a sequence number, which is also used in Remote Server Discovery.
3.3 Address Maintenance

The biggest concern of address maintenance process is do deal with network disconnections and its effects, e.g. the ones described in section 2.1.3.

Despite the remote allocation procedure, we can observe situations where no node in the network has more valid addresses to offer, either because the network has reached its maximum number of devices, either because some addresses have been lost over time. Therefore, addresses not in use must be recovered to relocation. In the best case a node can send a message informing that it will leave the network, so its address and its address space are recovered. However, is not expected that all the nodes will communicate their departure, since they may become out of range, suddenly broken, or in any other condition that may prevents a message to be sent or received.

The situation can be a little more complicated when a group of nodes moves together and disconnect from the main network (i.e. a partition is formed). In such situation, the resultant partitions may turn into networks with limited set of addresses to offer, and may end up needing to retrieve the address space of each other, with also can introduce problems in case of these partitions merge.

The address maintenance process includes two related phases: the address reclamation, and the partition and merge handle. The first one focus onto recover of lost addresses, the second one is concerned with duplication that may arise when partition or networks merge. Both processes are described subsequently.

3.3.1 Address Reclamation

As described in section 2.1.3, despite the importance of address reclamation process, it can be very inconvenient and introduce many networks overhead if performed too early. For this reason, the address reclamation process is triggered only when necessary, or in other words, the addresses are recovered only whether they must be allocated somewhere.

In the best case, a node can send a message informing that it will leave the network, so its address and its address space are recovered. More precisely, the leaving node sends its BNG seed and status to the node that offered its actual seed, which recover the address and are free to allocate it again. These messages aren’t expected or even obligatory, due to the behaviors presented in section 2.1.3, but it speed ups the process of address reclamation, and almost does not introduce network overhead.
To cover the most common cases, in which nodes move away without any report, the network tries to detect the currently unoccupied BNG seeds and status and redistribute them. The process starts when a new node detects the existence of a network but even after try the remote allocation process was not able to receive an address. The network existence is detected though the use of Hello messages, which are sent by all configured nodes periodically. An overview of the entire address reclamation process is presented in Figure 3.8 that contains a small state machine of such process.

![Figure 3.8: Address reclamation state machine.](image)

The reclamation process starts with an Address Reclamation Request message, sent by the new node that is still without address. This message contains a sequence number, generated by the new node, and the Network ID, received in the Hello message. Upon receive the request, each node that has the exactly same Network ID creates an Address Reclamation message, containing the same fields of the Address Reclamation Request, plus its own address, and must flood the message over the network. The node also sends to the requesting node, an Address Reclamation Reply with its seed and fields of Address Reply message used to properly configure the node, such as the network prefix and the amount of free bits.

Upon receive the Address Reclamation each node checks if it already processed the message, looking at sequence number, if the message was not processed yet, an Address Reclamation Reply is sent to the address present in the Address Reclamation message, and the received message is forward. While the Address Reclamation Reply messages are receive, they are forward to the new node. Until this point, the process is very similar to the remote address allocation process, but while
this last tries to minimize the flood, stopping the forward at first encountered server, the address reclamation process tries to reach the entire network.

After receiving all Address Reclamation Reply messages or the timeout expires, the address allocation three is reconstructed. To do that, we first order the received seeds from greatest to smallest, and attach to each seed its initial and recovered status value. The initial status is computed using Equation 3.2, and the recovered status is initially set to be equal to it, but updated according to the algorithm presented bellow. With the received seeds ordered and the initial and recovered status established, the following algorithm recovers all missing address space:

1. Select the next seed from the ordered list and mark it as the current working seed. If the list reaches its end, stop.

2. Compute the father of current working seed and register it if not yet registered. If there is no father go to step 1.

3. If the initial status of current working seed is smaller than recovered status of the father go to step 1, else update the recovered status of the father with the same value as the initial status of current working seed and select the father as the current working seed.

4. Register all unregistered children of current working seed, generated from the initial status up to the recovered status minus one and go to step 2.

The new terms - father and children - refer to the dependencies between generating and generated seeds. After following the above steps we ending up with a list of seeds and recovered statuses that represents the new valid address space. When the pairs of seeds and statuses are reclaimed, the node chooses the lowest seed not presented in the received list of seeds to be its own or fail and starts a new network. To completely configure itself, the node uses the network prefix and amount of free bits available from the messages sent by the firsts’ nodes that answered the reclamation request. Continuing the process, the new node generates a new Network ID and floods it though the network, along with the set of currently used seeds (updated list of received seeds) and the reconstructed addresses three.

The final step occurs when the previously configured nodes receive the reconstructed addresses three. First, they check if they are in the list of currently used seed, in negative case, which may be caused by lost messages, they release their addresses and start the allocation process again. Otherwise, the node updates its internal status to the received one. The node with the first seed in the currently used list recovers the first seed-status in the reclaimed list; the node with the second seed recovers the second seed-status in the reclaimed list, and so on. If the used seed list size is
smaller than reclaimed list size, some nodes only updates its network ID but does not receive any recovered addresses, on the contrary, if used seed list size is larger than reclaimed list size, some nodes will receive more than one recovered address. In this case, the solution states that seeds from reclaimed list go to the nodes in the same index from reclamation list when calculated the modulus of the size of currently used seed list. Actually, the new node uses such algorithm first, considering the size of used seed list plus one, to retrieve for itself a list of reclaimed addresses. For example, if we received the following list of seed \{(4), (5), (10)\}, the reconstruction algorithm will results in the following list of reclaimed seed-status pair \{(1, 3), (2, 4), (3, 2), (4, 2), (6, 3)\}. The new node gets the pair (1, 3), and from the resulting list \{(2, 4), (3, 2), (4, 2), (6, 3)\} it retrieves for itself the reclaimed seed-status list \{(2, 4), (6, 3)\} to itself. From the resulting reclaimed and flooded list \{(3, 2), (4, 2)\} the node 4 retrieves the pair (3, 2), and node 5 the pair (4, 2).

This strategy is very useful to re-arrange the address space among currently connected nodes, also beneficiates the merge of networks, the first node from the other network that merges and had performed address reclamation process, will receive the larger amount of recovered addresses, which will prove to be useful when the other nodes also start to merge.

3.3.2 Partition and Merge

Network partitions and mergers are common in MANETs due to node mobility, mainly. The merger of two networks can lead to address conflicts and cause instability in MANET. To avoid these problems this work includes a mechanism that detects when a merge occurs and determines what to do to maintain the addresses uniqueness.

As stated in section 2.1.3, network partitions for themselves, do not cause any address conflict. In the best case, networks can be partitioned and merge back without any concerns, but the depleted of addresses that may occur may lead to addresses reclamation. In this case, the quiet merge is no longer valid due to address conflict that may arise. For this reason, every address reclamation process generates a new Network ID, so partitions that performed the reclamation process are treated as distinct networks. One may argue that even partitions that performed addresses recovery may become consistent, simple by removing the reclaimed addresses, but since many recovered addresses may have already been allocated on the networks, in general, this procedure does not worth.

With this in mind, the process to handler partition and merge becomes quite simple, and only regards the Network ID, and an estimated network size. Moreover, the merge is handled in a step by step fashion, which avoids the huge amount of message that could be necessary to reallocate an entire network at the same time.
To make a estimation of network size, Hello messages contains the number of neighbors, and the network size becomes the number of detected neighbors plus the number of neighbors informed by each direct detected neighbors. Such information is kept with a timeout value to allow a size estimated from the current moment. This simple technique can result in an estimated size greater than the network itself, and also favors high densities networks, but still is a good estimation, mainly considering the most common case, in which a single or few nodes are merging a bigger network. Hello messages contains also the BNG seed and status of each neighbor that is used in case of single merges of network with the same size, as will be described later in this section.

When a hello message is received, the node checks if the received Network ID is the same as its own Network ID. In affirmative case, the node just updates the neighbors table, to estimate the network size and also helps the reclamation process, as described in section 3.3.1. If the Network IDs mismatches the node checks if the reported size and current size are the same. In affirmative case, the node will release it address and request a new one to the new discovered network, if one of the following sentences are true:

1. The reported status is bigger than its own status.
2. Both statuses are the same, but the reported seed it greater than its own seed.
3. Statuses and seeds are the same, but the reported Network ID is greater than it own Network ID.

This strategy clearly tries to privileges networks with the most number of nodes, which is the best approach to handler merge situations with low possible overhead. Moreover, merge is treated one node at time, instead of release all addresses of one of the networks, which could introduce much traffic at the same time, increasing the probability of corrupted or lost messages, and affecting the network overall performance and stability. To speed up the process, in case a node detects that the Hello sender must change its address, it can anticipate it hello interval, then the other node will detect the merge early.

A special case happens when both estimated size have the value zero. In such situation, instead of send a Server Discovery to a specific network, using the Network ID, the node that chooses to release the address sends the discovery with no restriction. If it is alone, any networks that appear will be good enough. In fact, the local and remote discoveries marked with specific networks are used to prevent a node that leaves a network to enter into another, ending at the same previous network.
4 Evaluation Methodology

“I hear and I forget. I see and I remember. I do and I understand.”

Chinese Proverb

DNCP protocol was implemented in a simulating environment in order to verify its functionalities, and also to check its overall performance. This chapter presents the methodology used to evaluate the DNCP. Firstly, it is presented some key characteristics of the chosen simulator environment. Then, the different scenarios used in simulation are described in details. The final subsection elucidates the metrics chosen to assess protocol performance and behavior.

4.1 Simulation

Network simulators are a good way to test protocols because they are easier to set up and cheaper than real testbed. In simulators, scalability can be seamlessly verified and they also allow reproducibility, fast creation of different scenarios, and fast results collection. In this work, the chosen simulator was NS-3[NSNAM10].

NS-3 is a discrete-event network simulator targeted primarily for research and educational use. It is intended as an eventual replacement for the popular NS-2 simulator [FAL09], and has a strong focus on realism by making models close to real word and easy to validate. NS-3 was developed using good software engineering strategies, what results in a well define architecture, entirely written in C++ programming language, and with support of open source community.

The experiments were run in a HP notebook model Pavillion dv5 1220br with its standard configuration, running the Linux operating system, distribution Ubuntu 9.10 [UBU10], and kernel version 2.6. Although NS-3 allows Python developing through binds, DNCP protocol was entirely implemented in C++ and integrated into the simulation in the scratch folder, which is used for test purpose.

4.2 Scenarios

To better evaluate the proposed protocol, this work focuses on two different scenarios. The first consists of static nodes that are deployed respecting a grid topology. The last is composed of dynamic nodes which can freely move inside a fixed region.
For both scenarios, each node has a Wi-Fi interface with transmission range of a little fifty meters and data rate of one megabit per second. The propagation delay and loss are simulated using two models presented in NS-3, the Constant Speed Propagation Model and Friis Propagation Loss Model. The OLSR routing protocol already implemented in NS-3 was chosen using its defaults parameters.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Retry Attempts</td>
<td>3</td>
</tr>
<tr>
<td>Remote Retry Attempts</td>
<td>3</td>
</tr>
<tr>
<td>Wait Timeout</td>
<td>0.5 seconds</td>
</tr>
<tr>
<td>Remote Timeout</td>
<td>2 * Wait Timeout</td>
</tr>
<tr>
<td>Hello Time</td>
<td>5 seconds</td>
</tr>
<tr>
<td>Neighbors Timeout</td>
<td>10 seconds</td>
</tr>
</tbody>
</table>

Table 4.1. Default DNCP parameters

The default parameters for DNCP are presented in Table 4.1. The local and remote attempts were set to 3 times, which means that a nodes will search for 3 times for local servers before trying another 3 times for remote servers. The timeout to wait for local replies was set to 0.5 seconds, while the timeout for remote messages was set initially to two times the Wait Timeout and is doubled for consecutive tries. Since the time to wait for local messages was set to 0.5 seconds, this means that the first, second and third remote offers are expected to arrive during 1, 2, and 4 seconds respectively. The periodicity for sending Hello messages was set to 5 seconds and the entries in neighbors table expires after 10 seconds.

4.2.1 Scenario A

The main purpose of this first scenario is to examine the protocol correctness, efficiency and scalability in a static environment. Grid topology with stationary nodes gives us a good way to observe DNCP’s functionalities in a simple and controlled way. Moreover, the evaluation metrics can be easily compared to expected behaviors.

In this first scenario, nodes are deployed respecting a regular rectangular grid, starting with node 0 at the center, and spiraling out. Figure 4.1 illustrates such construction that is known as Spiral Grid where the arrows indicate the order of nodes’ deployment. The reason to choose this topology is that even when the number of nodes in the grid varies, the network behavior is nearly the same, despite the additional traffic. This contributes to a better study of proposed protocol stability.
To limit direct communications only to a restrict set of nodes, and to force multi hop when needed, nodes are spaced so that nodes in diagonal stay forty nine meters apart, a little bellow than antenna range. The first node becomes online at zero time, and after that, the subsequent ones start up in order, respecting a uniform random variable which is distributed from 0.5 up to 8.5 seconds. Such delay is calculated only after the previous node has started, which prevents two nodes from become online at the same time, but also allows starting at close intervals. More than that, the range of the uniform random variable was chosen based on the supposed average address configuration time when using the DNCP default parameters aforementioned. Since the first node always delays the same time to create the network, which is in general the greater address configuration latency, the value of uniform variable received for the second node is incremented by five seconds. This configuration allows different situations for the address allocation process because new nodes can boot up before and after the previous node configuration. To better understand the scenario topology, Figure 4.2 presents a spiral grid completely filled with forty nine nodes. The number inside the circle indicates the node number, in other words, the starting order, not the address. We can notice that our topology allows three distinct neighborhood patterns. Nodes at the corner, like node 48, have only three neighbors; nodes at the edges have five neighbors, as node 39; nodes that are neither at edges nor at the corners have eight neighbors.
In this first scenario, we are not trying to study the impact of network partitions and merges, even because the objects are fixed. In the other hand, we are more focused on examining its basic functionalities. For that reason, we preferred to deny all nodes, but the first, the ability to create new networks. So a configuration error is computed whenever a node can't find available servers. Observations were taken from networks with 1, 4, 9, 16, 25, 36, 49, 64, 81, and 100 nodes. To study the impact of address space in address allocation we force the address space size to be the exactly number of nodes, which is the lowest possible address space to fill all nodes, and the must struggle situation to verify the protocol performance. The simulation time was fixed in three hundred seconds, which gives all nodes enough time to receive an address or fail in the process.

4.2.2 Scenario B

The Scenario B is meant to be a more realistic representation of the real ad hoc world. More precisely, we want to analyze the efficiency, stability and robustness of DNCP in situations where nodes can move freely within a region. Unlike Scenario A, links can become broken, and network partitions and merges may occur. In result, more control packets could be lost and the network could suffer with longer latency.

In this second scenario, nodes move dynamically according to a mobility model present in NS-3 named Random Direction 2D. In this model, the node movement is based on random
directions: each node pauses for a specific delay, chooses a random direction and speed and then travels in the specific direction until it reaches one of the boundaries of a specific rectangle area. When it reaches the boundary, it pauses, selects a new direction and speed, and the process starts again in cycle. For our scenario, nodes pause for two seconds and the rectangle is a square of side 300 meters.

Since we are dealing with moving nodes, it would not make much sense to deploy the nodes using the spiral square grid of Scenario A. In this case, a new model was developed to set up nodes initial positions, which in fact is a very simple one. All nodes start at the center of the square, the first node boots up at zero time and all other nodes start respecting a uniform random variable, from 5.5 up to two times the number of nodes in seconds, which means a inter arrival time of two seconds in average. This means that nodes start up on average every two seconds, but in practice, they can start even at the same time.

Unlike Scenario A, nodes are allowed to create new networks, configuration errors do not occur anymore, instead, we compute the number of created networks, the number of distinct networks that remains at the end of simulation, and also the node addresses changes. The simulation time was set to three hundred seconds and observations were taken from 25, 50, 75, and 100 nodes.

4.3 Metrics

The proposed protocol was analyzed considering three different criteria to study how it behaves under the restrictions of the above mentioned scenarios. The first criterion is the efficiency, which indicates the time required for a node to obtain an address and the following overhead. The second criterion is scalability, which requires that the increase of the numbers of nodes does not have a major impact on the overall performance of the protocol. Lastly, the stability indicates if the network reaches a stable condition with regards to the control overhead. The following metrics were used to study the above criteria:

- **Latency**: This latency represents the time from the moment that a node starts up until it is assigned an address.

- **Overhead**: The communication overhead measures the number of control packets generated to allocate and manage the addresses. It is considered in packets and in Bytes (packets size).

- **Configuration Error**: This metric is only applicable to the first scenario, and represents the amount of nodes that did not receive an offer after the local and remote tries.
- **Network Numbers**: This metric computes the total number of Network IDs generated through the simulation time and the number of Network IDs that remains at the end of simulation. Such IDs are generated by nodes that are creating a network or in case of address reclamation. This metric is only applicable to Scenario B.

- **Address Change**: Measures the frequency of addresses changes during the simulation, which only in case of network merge. The metric is only applicable to Scenario B.

To compute the average values of the metrics in both scenarios, a number of replications for each node population were done. The static characteristic of Scenario A makes the result varies less, compared to Scenario B, which has the results more dependent to network conditions. In order to makes the results a confident representation of the performance of the protocol under the analyzed conditions, a preliminary run of 20 simulations were executed to determine the sample average and standard deviation. Using this sample statistics in the Raj Jain formula [JAIN91], the quantity needed to have a 95% trust interval was discovered.

### 4.4 Static Network Results (Scenario A)

As above mentioned, the purpose of Scenario A is to understand the general behavior of DNCP protocol and verify primarily the correctness of the address allocation mechanism. We start the study by presenting the overall mean latency to configure an address. Next we describe the network overhead introduced by the protocol. Finally we comment the configuration errors and point some strategies to mitigate them.

#### 4.4.1 Latency

Figure 4.3 shows the average latency for a node to obtain its address in Scenario A varying the number of nodes. Only the nodes that get configured before the end of simulation ends were computed.
Figure 4.3. Latency for address configuration (1 up to 100 nodes)

The high latency presented by the first network node is due to the result of going through the configuration process without receiving any answer to its address requests. In this case, the node sends three Server Discover messages with timeout of 0.5s and then three Proxy Discover messages, with timeout 1, 2, and 4 seconds respectively that results in 8.5 seconds in total (refer to section 4.2). As the number of nodes increases, the average latency drops to values below 2.5 seconds. This can be best observed in As stated before, the DNCP is part of a bigger project in context, which presents a closer view of network latency, without the well-known behavior of network with only one node.

Almost all simulations resulted in a latency value close to 1.5 seconds, with the exception of the networks sized 4 and 16. In the case of network with four nodes the high latency is the result of the great impact of first node latency. In fact, with only 4 nodes, there is no need to contact remote

45
servers, which makes the latency close to 0.5 seconds, if we exclude the first node. The impact of the latency from network’s creator is attenuated as the number of nodes increases, as noticed looking at the result of network sized 9.

The good result of network sized 16, with regards to address latency, is due to a favorable topology, matching the address space, and the attenuated latency from the first node. In other words, the behavior of network sized 16 is similar to network sized 4 with respect to address allocation, since no remote offers are needed. Moreover, the former one has a lower latency due to the great number of nodes, which pulls down the latency, even with the high value presented by the first node.

For the others nodes population analyzed, the increase of the number of nodes has little effect in the mean latency, but they do not diverge too much, which lead us to believe that the protocol has some stability, or in other words, it does not present majors changes in the expected behavior while the network increases in size.

As we know, the latency for each node is different and directly affected by the arrival time, the network topology during configuring time, and the amount of free addresses from direct and remote neighbors. To illustrate this idea, Figure 4.5 presents the address configuration latency of each node in a single simulation of 25 nodes.

![Address latency per node for a network sized 25.](image)

Although the majority of nodes have a latency value near 0.5 seconds, there are some nodes with latency that varying from 2.5 seconds to more than 8 seconds. This particular simulation case was chosen because it presents some of the interesting patterns that may happen in Scenario A. The first thing that maybe catches attention in the flow chart is the high value of latency for node 1, but we already expected such value for node 0. For this singular simulation, the node 1 starts about time...
5.6 seconds, and since the node 0 only gets configured in time 8.5 seconds, the Server Discovery messages sent by node 1 are not answered until that time. Only with the third Proxy Discovery message node 1 receives an offer from node 0, which explains the high latency and also the similar value with network creator. It is important to note that even if Proxy Discovery messages were used, the process is still local, since a direct neighbor is responding and no Remote Server Discovery is sent. The node 3 starts in about 13.7 seconds, just slightly ahead from when its predecessor gets configured, which explains the lower latency of node 2 compared with node 1.

From node 3 to node 17 the network presents almost the same latency, near 0.5 seconds. These low values appear because the network already has a sufficient number of nodes to provide addresses and this number is still not large enough to compromise the address space. In other words, every time the node starts up, there is at least one addresses server on the neighborhood, which is not true for nodes 18 up to 24. These nodes enter in a network range with few addresses to offer and need to use the remote allocation process to get an address. For nodes 18, 19, 20, 21, and 24 the address is retrieved with the first attempt of remote allocation. Node 22 only gets configured in its second attempt of remote allocation and node 23 does not even receive an address. The node 23 had two problems that prevents it from receives an addresses. First, in one of its attempts to get configured, node 22 was faster and requests the address first, which forced node 23 to restart the remote process, and in the meantime node 24 also configures itself. Second, a reply message from one of the nodes was lost, and since we are using the address space of the exactly same size of the total amount of nodes and are not triggering the address reclamation process, node 23 has no option but to raise a configuration error. For more explanations about configuration errors check section 4.4.3.

4.4.2 Communication Overhead

In order to allocate addresses in a dynamic and distributed way, DNCP needs to exchange control messages, introducing overhead in the network. Figure 4.6 shows the number of broadcast and unicast messages, and also the number of these messages when subtracting the number of periodic messages Hello.
In Figure 4.6, the number of messages is plotted in a monolog scale to facilitate the visualization of values from all sized networks at the same time, since there is a great different in the number of messages sent in network with few nodes comparing with networks with high number of nodes. Figure 4.6 shows that Hello messages have a huge impact in the total amount of messages sent over the network by DNCP to configure or manage the node address. This is due to the fact that such messages are always sent from the instant time when the nodes are configured until the end of simulation. Moreover, considering the same simulation time for all network sizes, the contribution of Hello messages is relative bigger in simulations with few nodes, since in these cases, nodes get configured first, and after that, only Hello messages are sent.

The first node does not send unicast messages, since there are no other nodes besides it, this explains the absence of “Unicast” bar in the chart. From 4 up to 16 nodes, the number of unicast messages is higher than broadcast messages, and from 25 up to 100 nodes, it is the contrary, when considering only non-periodic messages. This happens because, with few nodes, the need for remote allocation is smaller, and the impact of forwarding the Remote Server Discovery messages is reduced. The network with 16 nodes has the best ratio when compared the relation between unicast and broadcast messages, due to the same reason for it low latency as the presented in Figure 4.3, i.e. the network size, arrangement and address space make no need of relaying process.
In order to better comprehend the communication overhead, Figure 4.7 presents the amount of message sent without the influence of Hello messages, also using a logarithm scale. In this new chart, the communication pattern was separated in 1-hop broadcast, $N$-hop broadcast, and Unicast. The first one corresponds to messages exchanged with neighbors in direct communication range. The second communication pattern ($N$-hop broadcast) groups the messages that can be flooded. Finally, the unicast messages are the ones destined to a particular node.

The first thing to note in Figure 4.7 in the one node configuration, beside the absence of unicast messages, is the same amount of 1-hop and $N$-hop broadcast messages. This is evident because the node tries to discover local servers the same a number of times as it tries to discover for remote servers. The smallest number of $N$-hop messages in networks sized 4 and 16 are due to their lack of need to use remote address allocation process, or more precisely the proxies and the Remote Server Discovery messages.

Another fact that can be obtained from Figure 4.7 is that 1-hop broadcast messages have a little effect in communication overhead comparing to unicast and $N$-hop broadcast messages. While 1-hop broadcast messages depends basically on the total amount of nodes in the network, unicast and $N$-hop broadcast messages heavily depends on network topology and more specifically, the address space and the amount of free addresses at the neighborhood of income nodes. Considering the bottlenecked situation imposed by the small address space, we may conclude that DNCP performs well considering the introduced overhead.

![Figure 4.7. Network overhead without periodic messages for Scenario A.](image)
Similar to the latency changes, reflecting the network condition, the communication overhead also changes. Figure 4.8 presents the average throughput in bytes per second during the simulation time, for networks with 1, 4, 9, 16, and 25 nodes. The results were sliced after 140 seconds, because all nodes had already been configured and the network had already reached its stability, as we can see by the contiguous throughput line. As we might expect, the good configuration of network sized 16, which does not need remote address reclamation, makes it performs better than network sized 9, even with more nodes. The high throughput from 100 to 125 seconds of the network sized 25 is the result of the small address space. Even with good distribution among the nodes, the address space is so small that the configuration process needs to cover almost all network nodes to find a free address.

4.4.3 Configuration Error

Operating with its normal behavior, when a DNCP node starts the configuration process and do not receive answer, it starts the address reclamation process or creates a new network. For the purpose of analyses of the protocol’s efficiency in allocating the address, we do not consider the address reclamation process and deny the capability of creation of new networks, allowing only the first node to do so. In this way, when a node boots up, and for any reason, it does not receive an address, we compute such event as a configuration error. Such errors mainly occur in situations where messages get lost and new incoming nodes believe that there are no address servers available.

The graph in Figure 4.9 presents the amount of configuration errors for the Scenario A. With few nodes, there was no configuration error, and only when the number of nodes is greater than or
equal to 25 the errors begin to appear, and as we might expect the number of errors increase as the network gets bigger. The amount of errors varies from one simulation to another, even with the same number of nodes. From 25 up to 49 nodes, the results clearly show that not all executions have errors, since the averages are below 1. For 64 and 81 nodes, the executions resulted in an amount of error that varied from 0 up to 4 with a mean near 1.5. The network sized 100 presented the largest number of errors, varying from 1 to 7, and with a mean of 4.3.

![Figure 4.9. Address configuration errors varying the number of nodes.](image)

These results may seem to be a problem at first analysis, but they only represent a small percentage of the total amount of nodes, and only happen in particular conditions, such as very restricted address space or lost Acknowledge messages. The most crucial factor that affects the configuration errors metric is the address space size. Since we are working with the lowest possible address space, analyzing the worst case, the result in fact appears very attractive. Moreover, by carefully adjusting DNCP parameters - like the number of retries or the time to wait for responses – we can reduce the amount of errors, but it comes with a cost. By increasing the number of retries we are also delaying the time to create a new network, and by increasing the wait time, we may catch the delayed responses, but it will surely impact the average address latency time.

### 4.5 Dynamic Network Results (Scenario B)

The purpose of Scenario B is to assess the protocol performance with regards to real environments, mainly the effect of nodes mobility. As with the previous presented study for static nodes, we start the study by presenting the mean latency to configure an address, followed by the network overhead. The configuration errors is not present, instead we present the metrics Network Number and the Address Change.
4.5.1 Latency

Figure 4.10 shows the average latency for a node to obtain its address in Scenario A for networks with 25, 50, 75, and 100 nodes. The network sized 25 presented the high throughput. As nodes moves freely, within the same region size, there is a greater likelihood of a node that starts in a network with few nodes to appear in a region poorly connected, with few or no addresses available, or even no other nodes nearby. For this reason, the higher the network density, the lower the latency for obtaining the address.

![Figure 4.10. Address latency for Scenario B varying the number of nodes.](image)

This statement clearly explains the decadently values presented in Figure 4.10, however, does not explain why the network with 25 nodes have an average latency that exceeds the greater value latency in Scenario A. As we may expect, this is due to the new reclamation process that takes places. It cans actually doubles the maximum amount of time that a node may takes to get configured. Since the network with 25 nodes is much spaced, the nodes may passes by all retries for address allocation more often, resulting in a large latency.

4.5.2 Communication Overhead

Figure 4.11 shows the number of broadcast and unicast messages, and also the number of these messages when subtracting the number of periodic messages Hello. The number of unicast messages presents the most regular behavior when considering the incrementing in the number of nodes, which is similar to the results presented in Scenario A. The Hello messages still has the major impact, and proportionally it is in fact a little greater than in Scenario A, since the nodes may starts early in simulation.
The number of broadcast messages in network with 25 nodes is out of the linear progression with values from the other networks due to its lower density. As the allowed space to move is the same for all networks, the network sized 25 may present more gaps and partitions, with makes the allocation process a little harder.

Figure 4.12 presents the amount of message sent without the influence of Hello messages, with the same communication pattern presented in Figure 4.7. The number of messages sent for networks with 50 up to 100 nodes, strongly respect the linear progression of the number of nodes. Starting from 50 nodes, the possible moving area is better covered by the nodes, allowing the protocol to work in a more attractive environment; this explains the similarities between the results of the more densities networks and the contrast with the network sized 25.
In the network sized 25, nodes have a greater probability to start with no neighbors, which increases the number of 1-hop broadcast messages. In this network, the disconnection appears more often, which may cause the restarts of the local and remote address allocation processes. Such disconnections also lead to more partitions and subsequent merges, which require some nodes to leaves its address and request a new one. Partitions may also trigger the address reclamation process more often, causing more network flood.

### 4.5.3 Network Creation and Address Changes

The network creation represents the number of distinct Network IDs generated by nodes that are creating a new network because they do not detect an existent network or when the address reclamation takes place, producing a new Network ID. Moreover, as network mergers, the Network IDs are disappearing and the nodes converges to a single and bigger network.

Figure 4.13 shows the average amount of Network IDs generated in the simulations and also the number of final active IDs. The number of final network IDs are really small than the number of created ones, as expected. The number of generated and also the final Network IDs are as lowers as bigger are the network, this is due to the more coverage area of larger networks, which results in less partitions, and less address reclamations.

![Number of Network IDs generated and remained varying the number of nodes.](image)

The same reason can be applied to the number of addresses changes presented in Figure 4.14. The greater difference from network sized 25 and network sized 50 is due to the very low connected characteristic of the former one.
Figure 4.14. Number of changes in the network varying the number of nodes.

We can also relate the number of networks created and the number of network that remains at the end of simulation with the number of addresses change. A simple observation is that, greater difference between these first two numbers, leads to greater amount of addresses that have changed. On the other hands, it only an initial to point out what is happening in the network, and does not explains the big difference between the network sized 25 and the network sized 50. We must remember that most changes of addresses, especially in the network with 25 objects, are due to merges of networks, which does not necessarily incur in address reclamation and the resulting creation of new Network IDs.

4.6 Comments

In this chapter we presented the methodology to evaluate the proposed protocol and discuss the results of the experiments done. The analyses show that our solution is scalable and performs well even in struggle scenarios. Many events prove to interfere in the behavior of the proposed protocol, like the network topology, starting time, and mobility, but the most determinant fact is the size of address space. Our experiments put the DNCP in a stressful environment where the amount of valid addresses are only the number of nodes, and yet find a way to allocate the available address to all nodes. In such scenario, solutions that rely on randomly identifiers, such as DAD approaches could prove to be very ineffective.
5 Conclusions and Future Work

“Using no way as way, having no limitation as limitation.”

Bruce Lee

This chapter presents the conclusions about the research project, including the proposed protocol, related work and the results of experiments done. The chapter also includes some future works to improve the DNCP and also the contributions of the research.

5.1 Final Considerations and Contributions

This work presented a new solution for automatic organization of addresses that take care of all challenges imposed by communications in mobile ad hoc networks without incurs in too many overhead. The solution employs a distributed generation function that is very efficient while still very simple. The strategy used in the developed function ensures the generation of unique addresses that makes solutions for duplicate detection unnecessary, and provides a good distribution of address space among nodes. Moreover, the proposed solution does not rely on any central or main servers, and the responsibility for address allocation is shared among all network nodes.

With regards to address leak, the solutions tries to delay as much as possible the reclamation of lost addresses, which greatly decreases the introduced overhead compared to approaches that recover the address as early as the leak is detected. Furthermore, by delaying the reclamation process we are giving a chance to departure nodes to come back again and join the network without change its address. This approach is aligned with the proposed mechanism for handling network merges. If no recovery is made, networks can get partitioned and merges back without any special treatment. In addition, the merge of distinct networks or from partition that already reclaimed addresses, are performed in a local and step by step way, which attenuate the problems that may emerge in case of a larger number of nodes floods the networks at the same time, which is more common in other approaches. The solutions also distinguish the case of single node joining from really networks merge that is not always taken into consideration in previous existent solutions.

The presented solution aimed at to be independent of the underlying routing protocol but the integration with it can improves the overall performance of the solution and also reduce the introduced overhead. Most routing protocols use some time of periodic messages in which our Hello masses could be piggybacked. With proactively routing approaches, the network size can be
estimated in much more precise way, and the reclamation process can also be improved, since the network maintains the vision of currently connected nodes.

We tried to understand the functioning of the chosen simulator and implementing the underlying functionalities need to start the implementation of the proposed protocol. Moreover, it was a very demanding task to find a function that ensures uniqueness, and developing a solution to handles partitioning, merging, and to reclaim the addresses without suffer with too much overhead. As a biggest challenge, matches all solutions together and cover all situation presented by the MANET environment.

The NS-3 proves to be a very challenging environment since it was not yet adjusted to work with addressing mechanism. A new layer of addressing was developed, including functions to boot up nodes in different times. The algorithm to deploy nodes based on spiral grid was offered to the community and increments the sets of previously existent NS-3 nodes position allocation.

As a major contribution of this work, we achieve an international patent application focused in the distributed generation function and the allocation method of DNCP [ASCH09]. Moreover, there are two papers under developing pretended to be published in scientific conferences and symposiums. The protocol was also embraced by Ericsson Research Lab to be used in a project related to routing in mobile ad hoc networks. The DNCP proves to be an important step to boot up the network providing unique identities that are need to the routing protocol, as well as deal with addressing challenges presented by the targeting mobile scenario of the routing protocol.

5.2 Future Works

There is some work to be developed regarding to DNCP. Our intended future work is focused on improving the performance of DNCP. We are dividing our efforts in two different lines in order to evaluate our approach:

- Extends the experiments to include more metrics and new scenarios.
- Implementation of DNCP in real environment.

This thesis presented the evaluation of the basic functionalities of the protocol in very limited address space scenario. Now we are intend to evaluate our protocol with more variables, like background traffics, and also varies some of the currently study metrics to verify the impact of their change in the overall protocol performance. Moreover, we need to evaluate the protocol with other scenarios, including bigger networks merges, and battery depletion or broken nodes.
In fact, we are trying to better understand how the protocol handles different situations and the resulting performance. Such study may be useful to do some improvements in the proposed protocol. Two main focus points of improvement are the remote allocation process and the network estimation size. The larger number of messages generated from the remote allocation process can be seen as the most fragile part of the approach, as they represent a by far the most weight in relation to control overhead. The Hello messages, and more precisely the neighborhood status table construct with it, could be used to detect available server and redirect the requisitions without result in flooding the network. The network estimation plays an important role in the networks merge handler, and more researches can point out that currently strategy to such estimation is not well enough to some specific cases. Furthermore, we are planning to evaluate the reclamation process with some little changes, like to use one hop broadcast messages in response to reclamation request instead of direct unicast messages. In such way, every node that receives a request should set a timeout for its reply, and when it expires, send a one hop broadcast message with its status along with received status from neighbors’ nodes. The timeout should be set up initially according to network diameter, and after that, decreasing as reclamation requisitions hop away.

We also pretend to evaluate and to compare our solutions with other approaches, mainly, stateful ones. This can improve our first analytical analyses of the studied related works, and enforces the supposed DNCP ability to perform better in distinct scenarios. Actually, we already start the implementation Prophet and Prime stateful solution, and planning the implementation of at least one representative approach for stateless addressing.

As stated before, the DNCP is part of a bigger project in context of routing and deploy of MANETs. Such project is target at real environment and we have been implementing some of the core functionalities of the proposed solution to meet its requirement, chiefly the BNG function and the local allocation process, which are already completely done. With such implementation, we can advance the studies of the protocol’s functionality and performance by evaluating and validating it in real operations, which is in fact the final goal.
6 References


Ata de Defesa de Dissertação de Mestrado do
Centro de Informática da Universidade Federal de Pernambuco, 28 de maio de 2010.

Ao vigésimo oitavo dia do mês de maio do ano
dois mil e oito, às dez horas, no Centro de Informática da Universidade Federal de
Pernambuco, teve início a *octingentésima octogésima nona* defesa de dissertação de
mestrado em Ciência da Computação, intitulada "DNCP: Dynamic Node Configuration
Protocol" do candidato Rafael Roque Aschoff, o qual já havia preenchido anteriormente
as demais condições exigidas para a obtenção do grau de mestre. A Banca Examinadora,
composta pelos professores Paulo Roberto Freire Cunha, pertencente ao Centro de
Informática desta Universidade, Stênio Flávio de Lacerda Fernandes, pertencente ao
Departamento de Ensino do Instituto Federal de Alagoas da e Djamel Fawzi Hadj Sadok,
pertencente ao Centro de Informática desta Universidade, sendo o primeiro presidente da
Banca Examinadora e o último orientador do trabalho de dissertação, decidiu, em
exigência, dar o prazo de trinta dias para a entrega da versão final do trabalho, com as
devidas correções propostas pela banca examinadora. E para constar lavrei a presente ata
que vai por mim assinada e pela Banca Examinadora. Recife, 28 de maio de 2010.

 tổ !
Mária Lilila Pinheiro de Freitas
(Secretaria da Pós-Graduação em Ciência da Computação)

Prof. Paulo Roberto Freire Cunha
Centro de Informática / UFPE

Prof. Stênio Flávio de Lacerda Fernandes
Departamento de Ensino / IFAL

Prof. Djamel Fawzi Hadj Sadok
Centro de Informática / UFPE