www.euro6ix.net

Title:	Document Version:
Technical Report TR4.1A.1 **Mobility over IPv6 Networks**	2.4

Project Number:	Project Acronym:	Project Title:
IST-2001-32161	Euro6IX	European IPv6 Internet Exchanges Backbone

Contractual Delivery Date:	Actual Delivery Date:	Deliverable Type* - Security**:
31/12/2002	04/03/2003	R – PU

* Type: P - Prototype, R - Report, D - Demonstrator, O - Other
** Security Class: PU- Public, PP – Restricted to other programme participants (including the Commission), RE – Restricted to a group defined by the consortium (including the Commission), CO – Confidential, only for members of the consortium (including the Commission)

Responsible and Editor/Author:	Organization:	Contributing WP:
Tomas P. de Miguel (UPM) Ivano Guardini (TILAB)	TILAB - UPM	WP4

Authors (organizations):

Miguel A. Díaz (Consulintel), Francisco Fontes (PTIN), Victor Marques (PTIN), Carlos Parada (PTIN), Carlos Rodrigues (PTIN), Jacinto Vieira (PTIN), Mario Morelli (TILab), Antonio F. Gómez-Skarmeta (UMU), Maria J. Perea (UPM).

Abstract:

This Annex summarizes the research done during the first year in the context of the Euro6IX project regarding the selection, configuration and test of many solutions on the field of macro and micro mobility over IPv6. During this period two groups of experiments have been developed: partner tests and multi-partner tests. The obtained results invite to continue the work during the next period; deploying the mobile service over Euro6IX network and testing new developments.

Keywords:

Handover, HMIPv6, implementation, interoperability, macro mobility, micro mobility, MIPv6, performance.

© Euro6IX Consortium

Revision History

Revision	Date	Description	Author (Organization)
v1.0	15/11/2002	Document creation	Tomas P. de Miguel (UPM) Maria Jose Perea (UPM)
v1.1	29/11/2002	UMU contribution integration	Antonio Skarmeta (UMU)
v1.2	19/12/2002	PTIN contribution integration	Carlos Parada (PTIN) Francisco Fontes (PTIN) Victor Marques (PTIN) Carlos Rodrigues (PTIN) Jacinto Vieira (PTIN)
v1.3	19/12/2002	TILAB contribution integration	Mario Morelli (TILab)
v1.4	19/12/2002	CONSULINTEL contribution integration	Miguel A. Díaz (Consulintel)
v1.5	20/12/2002	TID contribution integration	Ruth Vázquez Cerro
v2.0	08/1/2003	TID corrections to the previous document	
v2.1	9/1/2003	CONSULINTEL minor corrections to the previous document	Miguel A. Díaz (Consulintel)
v2.2	13/1/2003	UPM abstract and executive summary	Tomás P. de Miguel (UPM)
v2.3	11/1/2003	Style corrections	David Fernández (UPM)
v2.4	04/03/2003	Logos added an PDF generated	Jordi Palet (Consulintel)

Executive Summary

This Annex deliverable summarizes all the tasks planned for the first year of Euro6IX project in the context of WP4 subactivity A4.1-1 (Mobility over IPv6). During this period, several macro and micro mobility implementations have been tested.

The work has been divided in two tasks: local partner tests and multipartner tests. The first one was devoted to evaluate some implementation alternatives with different operating systems not only in macro but also in micro mobility field. The second task has been devoted to demonstrate the availability of Euro6IX network to support such type of services (distributed tests).

MIPv6 operation has been tested in local testbeds of several partners, testing Home Agent (HA), Mobile Node (MN) and Correspondent Node (CN) functionality, using different implementations and testing interoperability and performance. In addition, multipartner mobility tests have involved visiting mobile nodes from a home to a foreign network geographically distant in order to test the basic ‘roaming’ service. Additional tests have been comprised mobility within foreign networks, so handovers and performance of re-registration to far HA have been tested. Another important issue that has been treated is the interoperability among different MIPv6 implementations: Linux, Windows, CISCO, 6WIND, etc.

The document finalizes with a detailed description of the tests carried out and the plan for the next period.

Table of Contents

1. Introduction 10

1.1 TILab comparison of MIPv6 implementations 11

2. Objectives 12

2.1 Micro mobility 12

2.1.1 BCMP Implementation 13

2.1.2 Test Bed Topology for Micromobility Testing 13

2.1.3 Results of the experiments 14

2.2 Macro mobility 15

3. Description of the Tests Performed 16

3.1 Micro mobility tests 16

3.1.1 Micro mobility tests in UMU 16

3.1.1.1 Installing FreeBSD 3.4 and INRIA IPv6 Stack 16

3.1.1.2 INRIA's HMIPv6 implementation. 17

3.1.1.2.1 Base Station and simple PC-routers 18

3.1.1.2.2 Home Agent 18

3.1.1.2.3 MAP (Mobility Agent) 18

3.1.1.2.4 Mobile Host 18

3.1.1.2.5 How to use 19

3.1.1.2.5.1 Base Station. 20

3.1.1.2.5.2 MAP (Mobility Agent) 20

3.1.1.2.5.3 Mobile Host 20

3.1.1.2.5.4 Home Agent 20

3.1.1.2.6 Drawbacks 20

3.2 Local mobility tests 21

3.2.1 PTIN 21

3.2.1.1 Implementing the tests 22

3.2.1.1.1 LANs testing 22

3.2.1.1.2 WLANs testing 23

3.2.1.2 Conclusions 23

3.2.1.2.1 MIPL tests 23

3.2.1.2.2 Windows, Cisco and MIPL tests 24

3.2.2 UPM 25

3.2.3 TILAB MIPv6 implementations comparison 26

3.2.3.1 MIPv6 imlementation analysis 26

3.2.3.2 Overview 27

3.2.3.2.1 ScenarioA: LandMARC Mobile IPv6 for Windows 2000 28

3.2.3.2.1.1 LandMARC Test bed 28

3.2.3.2.2 Scenario B: MIPL Mobile IPv6 for LINUX 29

3.2.3.2.2.1 MIPL test-bed 30

3.2.3.3 Testing Activity 31

3.2.3.3.1 Network layer test 31

3.2.3.3.2 Transport Layer Test 32

3.2.3.3.3 Application Layer Test 33

3.2.3.3.3.1 W2000 implementation 33

3.2.3.3.3.2 MIPL implementation 33

16. See the previous test 34
3.2.4 Consulintel 34
3.2.4.1 Mobility tests 34
3.2.4.1.1 Introduction 34
3.2.4.1.2 Designing the test bed 35
3.3 Distributed mobility tests 36
3.3.1 Distributed mobility tests between UPM and BT 36
3.3.2 Distributed mobility tests between UPM and UMU 37
4. Future Work 40
4.1 MIPv6 area 40
5. Summary and conclusions 42
6. Appendixes 43
6.1 Appendix A: Annexes to HMIPv6 Tests in UMU 43
6.1.1 INRIA's HMIPv6 IMPLEMENTATION TEST. 43
6.1.2 CONFIGURATION FILES 48
6.2 Appendix B: Detailed MIPv6 tests in PTIN description 49
6.2.1 Introduction 49
6.2.2 Technical Analysis of the MIPv6 49
6.2.2.1 Introduction 49
6.2.2.2 Implementations 50
6.2.3 Testbed Design 51
6.2.4 Implementing the tests 51
6.2.4.1 LANs testing 52
6.2.4.1.1 Scenario 1 52
6.2.4.1.2 Scenario 2 54
6.2.4.1.3 Scenario 3 59
6.2.4.1.4 Scenario 4 67
6.2.4.1.5 Scenario 5 73
6.2.4.2 WLANs testing 83
6.2.4.2.1 Scenario 1 84
6.2.4.2.2 Scenario 2 86
6.2.4.2.3 Scenario 3 91
6.2.4.2.4 Scenario 4 100
6.2.4.2.5 Scenario 5 105
6.2.5 Handover on Applications 115
6.2.5.1 Ping 116
6.2.5.2 WWW 117
6.2.6 Interoperability tests 120
6.2.6.1 Cisco HA; Windows MN; Windows CN 121
6.2.6.1.1 Cisco HA; Windows MN; Windows CN 121
6.2.6.1.2 Cisco HA; Windows MN1; Windows MN2 124
6.2.6.1.3 Conclusions 127
6.2.6.2 Linux HA; Linux and Windows (MN and CN) 127
6.2.6.2.1 Linux HA; Linux MN; Windows CN 127
6.2.6.2.2 Linux HA; Windows MN; Linux CN 131
6.2.6.2.3 Linux HA; Windows MN2; Linux MN1 135
6.2.6.2.4 Conclusions 138
6.2.6.3 Cisco HA; Linux and Windows (MN and CN) 139
6.2.6.3.1 Cisco HA; Windows MN; Linux CN 139

6.2.6.3.2 Cisco HA; Linux MN; Windows CN 142
6.2.6.3.3 Conclusions 145
6.2.7 Conclusions 146
6.2.7.1 Linux tests....146
6.2.7.2 Windows, Cisco and Linux tests 147
6.2.8 Future work....148

6.3 Appendix C: PTIN MIPv6 tests annexes 149
6.3.1 MIPL Tests 149
6.3.1.1 HA configurations 149
6.3.1.2 CN configurations....150
6.3.1.3 MNs configurations 152
6.3.2 Windows, Cisco and Linux interoperability Tests 155
6.3.2.1 CISCO HA configurations....155
6.3.2.2 Windows CN configurations 155
6.3.2.3 Windows MN configurations 155
6.3.2.4 Windows MNs configurations....156
6.3.2.5 Linux CN configurations....156
6.3.2.6 Linux MN configurations 157
6.3.2.7 Linux HA configurations....158

6.4 Appendix D: Detailed analysis of MIPv6 implementations in TILAB 159
6.4.1 Linux MIPL 159
6.4.1.1 Introduction 159
6.4.1.1.1 Testbed description....160
6.4.1.1.2 Measurements....160
6.4.1.2 Results 161
6.4.2 Windows 2000....165
6.4.2.1 Introduction 165
6.4.2.2 Overview on LandMARC implementation for W2000....166
6.4.2.2.1 Main Features 167
6.4.2.2.2 Limitations....167
6.4.2.2.3 Known Bugs 167
6.4.2.3 Testing Activity 168
6.4.2.3.1 Network Layer Mobility Test....170
6.4.2.3.2 Transport Layer Test 171
6.4.2.3.3 Application Layer Test 172
6.4.2.4 Results 173

6.5 Appendix E: Detailed MIPv6 tests in Consulintel 174
6.5.1 Mobility tests 174
6.5.1.1 Introduction 174
6.5.1.2 Objectives 175
6.5.1.3 Test Bed....175
6.5.1.3.1 MIPv6 implementation on Windows....175
6.5.1.3.2 ICR IPv6 Stack Features....176
6.5.1.3.3 Installing and configuring the ICR IPv6 Stack....176
6.5.1.4 Designing the test bed....181
6.5.1.4.1 Designing principles 183
6.5.1.4.1.1 Home Agent....184
6.5.1.4.1.2 Correspondent Node 187
6.5.1.4.1.2.1 Installing the IPv6 Stack....187
6.5.1.4.1.2.2 Configuring mobility 188
6.5.1.4.1.3 Mobile Node 189

6.5.1.5 Running the test-bed 191
6.5.1.6 Conclusions 195
6.5.1.7 Future work 196
6.5.1.8 Glossary 197

7. References 198

Table of Figures

Figure 2-1: BCMP Overview ... 12
Figure 2-2: Networks Structure ... 14
Figure 3-1: General Testbed used for MIPL tests. ... 22
Figure 3-2: Scenario for local mobility tests at UPM ... 25
Figure 3-3: Traffic received by the Mobile Node while changing its location ... 26
Figure 3-4: Time elapsed after handover in 30 consecutive handovers ... 26
Figure 3-5: Windows 2000 testbed ... 29
Figure 3-6: Linux test bed ... 30
Figure 3-7: Basic Mobility Windows Test-bed ... 36
Figure 3-8: Scenario for BT – UPM Multipartner IPv6 Mobility Tests ... 36
Figure 3-9: Traffic received by the Mobile Node ... 37
Figure 3-10: Time elapsed after handover in 30 consecutive handovers ... 37
Figure 3-11: Scenario for UMU-UPM Multipartner IPv6 Mobility Tests ... 38
Figure 3-12: Traffic received by the Mobile Node ... 38
Figure 3-13: Time elapsed after handover in 25 consecutive handovers ... 39
Figure 6-1: HMIPv6 basic testbed ... 43
Figure 6-2: Average time between switch of interfaces ... 44
Figure 6-3: Results tests ... 46
Figure 6-4: General Testbed used for MIPL tests. ... 51
Figure 6-5: LAN: Scenario 1: MN in the HN. CN is located in other network. ... 52
Figure 6-6: LAN: Scenario 2: MN starting in a FN. ... 54
Figure 6-7: LAN: Scenario 3: The MN starts in its HN, goes to a FN and backs to the HN. 60
Figure 6-8: LAN: Scenario 4: The MN1 is on the HN and the MN2 is on a FN. ... 68
Figure 6-9: LAN: Scenario 5: The MN1 is on the FN1 and the MN2 is on a FN2. ... 74
Figure 6-10: WLAN: Scenario 1: MN in the HN. CN is located in other network. ... 84
Figure 6-11: WLAN: Scenario 2: MN starting in a FN. ... 86
Figure 6-12: WLAN: Scenario 3: MN starts in its HN, goes to a FN and backs to the HN. .. 92
Figure 6-13: WLAN: Scenario 4: The MN1 is on the HN and the MN2 is on a FN. ... 100
Figure 6-14: WLAN: Scenario 5: The MN1 is on the FN1 and the MN2 is on a FN2. ... 106
Figure 6-15: Start of the ping application (from CN to MN). ... 116
Figure 6-16: Middle of the ping application (handover is happening). ... 116
Figure 6-17: End of the ping application (with Ctrl+C). ... 117
Figure 6-18: Start of the web Application (download a 10MB text file). ... 118
Figure 6-19: Middle of the web application (Handover is happening). ... 119
Figure 6-20: End of the web Application (10MB text file downloaded successfully). ... 120
Figure 6-21: Cisco HA; Windows MN; Windows CN. ... 121
Figure 6-22: Cisco HA; Windows MN1; Windows MN2. ... 125
Figure 6-23: Linux HA; Linux MN; Windows CN. ... 128
Figure 6-24: Linux HA; Windows MN; Linux CN. ... 132
Figure 6-25: Linux HA; Windows MN2; Linux MN1. ... 136
Figure 6-26: Cisco HA; Windows MN; Linux CN. ... 139
Figure 6-27: Cisco HA; Linux MN; Windows CN. ... 143
Figure 6-28: Test bed with delay simulator ... 160
Figure 6-29: Throughput between CN and MN ... 163
Figure 6-30: comparison table between different throughputs ... 163
Figure 6-31: Test bed scenario for transmission in both directions ... 164
Figure 6-32: Throughput between CN and MN, depending on traffic direction ... 165
Figure 6-33: Test Bed used for the testing of LandMARC implementation ... 169
Figure 6-34: LAN properties ... 177
Figure 6-35: Selecting protocol. ... 178
Figure 6-36: LAN properties ... 179
Figure 6-37: ICR IPv6 stack properties ... 180
Figure 6-38: IPv6man application ... 181
Figure 6-39: Basic Mobility Test-bed ... 182
Figure 6-40: Basic Mobility Windows Test-bed ... 183
Figure 6-41: Consulintel's network architecture test-bed ... 184
Figure 6-42: Installing IPv6 Protocol ... 188
Figure 6-43: Starting mobility from Mobile Node ... 190

Figure 6-44: Running test-bed 192
Figure 6-45: CN communicating to MN 195
Figure 6-46: Mobile Node losing connectivity 196

1. INTRODUCTION

The Internet is today an environment broadly used for many different purposes. In the last years, with the increasing number of available access technologies, with particular relevance for the wireless ones, and the use of new small devices such as laptops, PDAs or mobiles phones, it becomes each time most important that the Internet Protocol of the Next Generation can deal with this mobility in an appropriate way. The IPv6 protocol provides a more appropriated support of mobility than IPv4, bringing clear benefits in efficiency and simplicity.

The uses of IPv6 native mechanisms like auto-configuration, or Extension Headers like Routing Header or Destination Option Header, are facilities that improve the simplicity of how the protocol are implemented and, at the same time, the efficiency.

The aim of this subactivity is to test both basic micro and macro mobility protocols operation. Both kinds of protocols complement each other, but macro mobility implementations are much more matured than micro mobility ones.

MIPv6 is not a closed issue but a “work in progress” field traditionally bound to developing environments like Linux. HUT [mipl] is an excellent and well known implementation example of MIPv6 on Linux.

Other platforms like Windows did not have any implementation until now because the source of this operating system is not liberated. However more and more advances and new releases are developed every day, some of them based on Windows.

MIPv6 operating on Linux is well known thanks to previous works made by Consulintel, UMU, UPM and PTIN and documented on the Deliverable D4.1 Annex I. However, there is not any kind of experience neither on this field on Windows nor interoperability between MIPv6 Linux and MIPv6 Windows.

So, it seems very interesting to make any kind of trials on this field in order to check the “state of the art” nowadays and get a lot of experience that will be useful to us in the next works.

These tests have been performed both in a local scenario (within UPM’s premises) and in a distributed one (multipartner mobility tests).

Basic MIPv6 operation has been tested in local testbeds of several partners, testing Home Agent (HA), Mobile Node (MN) and Correspondent Node (CN) functionality, using different implementations and testing interoperability and performance. In addition, multipartner mobility tests have involved visiting mobile nodes from a home to a foreign network geographically distant in order to test the basic ‘roaming’ service. Additional test will have comprised mobility within foreign networks, so handovers and performance of re-registration to far HA can be tested. Another important issue that has been focused is the interoperability of different MIPv6 implementations: linux, windows, cisco, 6wind, etc.

From the point of view of micro mobility, INRIA implementation of HMIPv6 protocol has been tested by UMU.

1.1 TILab comparison of MIPv6 implementations

The tests carried out in the TILab IPv6 laboratories inside the A4.1 sub-activity take into account two different Mobile IPv6 implementations: the first one for Linux operating system and the other one for Windows 2000 operating system.

For each of these, the test-beds and the relatives tests carried out will be shown.

The experimentation regards, first of all, tests at network-layer to verify the degree of conformance of the implementations under test to the MIPv6 draft. Then transport layer tests have been carried out to understand if the transport layer protocols are transparent to the mobility of the mobile node. At the end, application-layer tests have been carried out to analyze what happens to the applications already active in an IPv6 mobility environment when a mobile node moves from a subnet to another one.

The tests permit us to make also a comparison about two different implementations and to understand the maturity and reliability level of both of them.

2. OBJECTIVES

2.1 Micro mobility

Micro-mobility concerns the user's movement inside a domain while macro-mobility is related to the movement of the user from one administrative domain to another.

Mobility differentiates between care-of address as the access address and home address as the identifier address. When a mobile node moves from its home network to its foreign network, the home address remains the same but the care-of address changes. This is known as macro-mobility. Micro-mobility is when both addresses remain unchanged because the mobile node is moving inside the same administrative domain.

Besides other features as paging and security, the most relevant feature that micro-mobility tries to solve is the fast handoff support. There have been several proposals that deal with micro-mobility of which the three main ones are: HAWAII, Cellular IP and HMIP.

Some Micro-mobility proposals:

- INRIA Hierarchical Mobile IPv6 Proposal (HMIPv6)
  http://www.inrialpes.fr/planete/people/bellier/hmip.html
- MobiWan: NS-2 extensions to study mobility in Wide-Area IPv6 Networks:
  http://www.inrialpes.fr/planete/mobiwan/
- Hierarchical Mobile IPv6 Research at CTIE:
  http://www.ctie.monash.edu.au/ipv6/hmipv6.htm
- Cellular IP by Columbia:
  http://www.comet.columbia.edu/cellularip/
- Mobile IP Paging:
  http://www.comet.columbia.edu/pmip/
- Columbia IP micro-mobility software:
  http://www.comet.columbia.edu/micromobility/

The objective of this subactivity has been focus on the evaluation of BRAIN Candidate Mobility Management Protocol (BCMP), designed for local mobility management in the MIND networks. King College London (KCL) has developed the implementation version of BCMP used in these experiments.

The key features of the protocol are summarized below, while its outline operation is illustrated in Figure 2-1. A detailed description of the original definition of BCMP is given in [BD22].

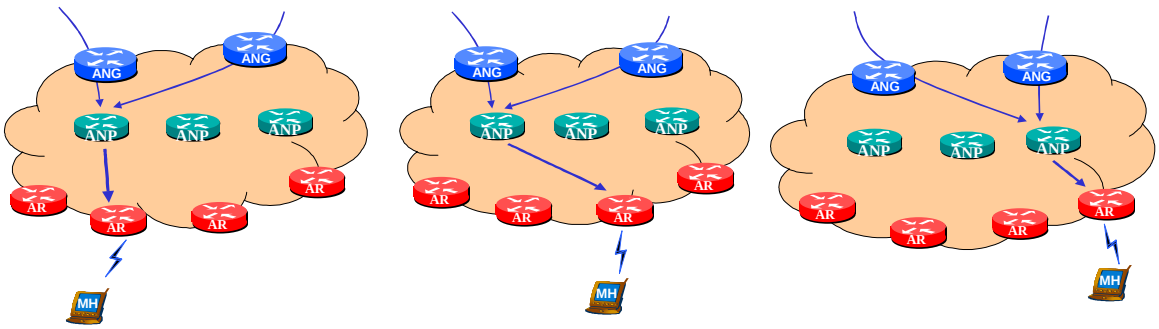


Figure 2-1: BCMP Overview

Generally, the access network consists of legacy IP routers. In order to support the protocol, mobility aware functionality is embedded in key components of the AN:

- The Access Routers (ARs) which are located at the edge of the AN and offer IP connectivity to the MN. They act as the default router to the MNs that they serve.
- The Anchor Point (ANP) which are located 'inside' the AN at different selected positions. ANPs own and allocate IP addresses, authenticate users, maintain user records, and tunnel packets towards MNs. The AR terminates tunnels from ANPs and forward packets to/from MNs.

In this context, the Access Network Gateways (ANGs) do not need to have mobility specific functionality – their role is that of a standard border router, shielding the rest of the AN from the exterior routing protocols and distributing traffic within the AN to the correct ANPs.

The functions that are supported by the protocol are:

- **Login** (Address Management and Security) for the MN to securely attach to the AN,
- **Handover and Path Updates** to ensure that the MN can seamlessly change ARs in a planned or unplanned way,
- **Inter-Anchor Handover** so that the MN can change ANP if it has moved far away from its original ANP, and
- **Paging** in order to effectively track the location of idle MNs attached to the AN.

2.1.1 BCMP Implementation

BCMP was implemented in Linux and both IPv4 and IPv6 versions were produced. The main functions of the available implementation were:

- **Initial Login**: A MN will attach to the network through an AR by acquiring an IP address from the ANP. It can then establish communication with a Correspondent Host (CH) outside the AN.
- **Handover Management:** The MN can then move and change AR, executing handovers with or without prior handover preparations (planned and unplanned handovers), while maintaining all its data connections.
- **Path Updates:** After every handover, the path update function will take place, updating the path from the ANP to the MN, by changing the tunnel to point to the new AR.
- **Change of Anchor:** The MN can optionally perform a change of Anchor thus registering and getting an address from a new ANP. This however requires the existence of a global mobility management protocol (like Mobile IP), which is also coupled with the BCMP implementation.
- **Logout:** Finally the MN can also logout and free all the resources that it has been using in the network.

2.1.2 Test Bed Topology for Micromobility Testing

The network topology in UPM used for the experiments is due to change depending on the tests being performed. UPM has a 'stable' part of the network, basically connecting to external sites and networks and several different networks attached to a layer 2 switching backbone, so it can be easily customized. The 'experimental' part is likely to contain several different subnetworks.

As we can see in the figure, the stable part contains two different networks running only MIPv4/MIPv6. In the case of IPv4 Cisco routers (2600) are used as Home Agents and WLAN hubs as Foreign Agents. For IPv6 a PC act as a HA. There are other nodes that will take part in experiments acting as traffic generators (noise) or conference participants (Isabel).

The customisable part is based in a gateway that may be a Cisco router or a Linux/FreeBSD (depending on the implementation) and several PCs forming an access network. This network will form a micro-mobility domain where several experiments can be performed, depending on the tests. For example, in the figure, we can see a test on a simple handover where one of the nodes on the same domain (it could it be even the gateway) maintains a communication with a moving laptop.

The addressing and the operating systems depend also on the tests being performed and the availability of implementations. This platform allows UPM to have a very flexible environment both for macro and micro mobility experiments and performance evaluation.

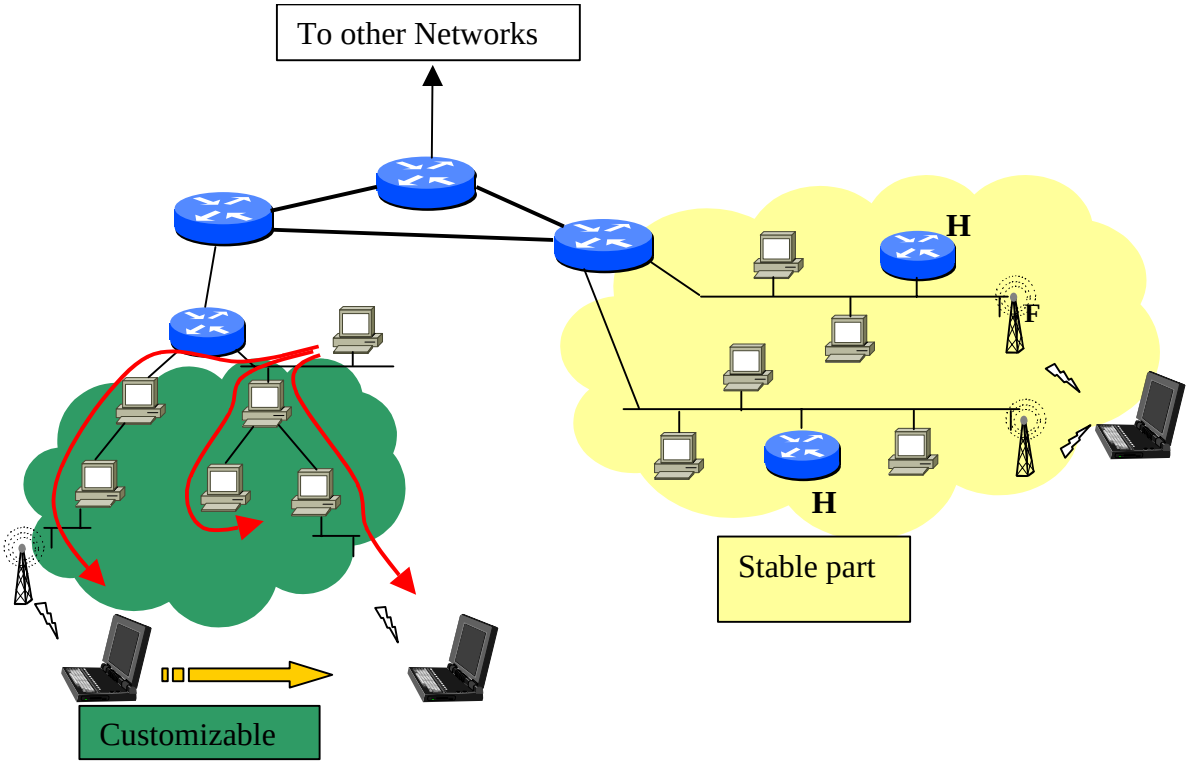


Figure 2-2: Networks Structure

2.1.3 Results of the experiments

Different experiments can thus be performed but for the results presented here a single AN with one ANP is considered and the focus is on BCMP operation, implementation validation and performance evaluation.

Qualitative experiments were performed in order to integrate the different elements and to get an informal evaluation of the user perceived QoS when terminal moves while running one videoconference application. Those experiments revealed a satisfactory quality for supporting videoconference services.

Only planned handovers over BCMP have been experiments during this phase. More experiments have to be performed in order to compare planned and unplanned handovers, as well the use of different configuration modes of BCMP.

2.2 Macro mobility

The objectives for this subactivity related to macro mobility have been:

- To know the existing mobility over IPv6 implementations.
- To evaluate and to test this MIPv6 implementations.
- To test the interoperability between MIPv6 implementation on Windows environment and MIPv6 implementation on Linux environment.

Regarding multipartner tests, this experience was intended to test basic MobileIPv6 operation in a distributed scenario, testing the basic 'roaming' service. The intention is to provide mobility service to visiting nodes, from a Home to a Foreign Network geographically distant.

Also, mobility within Foreign Networks will be tested, so handovers and performance of re-registration to far HA can be tested.

3. DESCRIPTION OF THE TESTS PERFORMED

3.1 Micro mobility tests

3.1.1 Micro mobility tests in UMU

3.1.1.1 Installing FreeBSD 3.4 and INRIA IPv6 Stack

During this document, we are not going to explain the complete installation of FreeBSD 3.4 but how we can turn FreeBSD 3.4 into a FreeBSD 3.4 v6 by using INRIA IPv6 stack. This is needed because INRIA HMIPv6 implementation works under FreeBSD 3.4 and they use the Francis Dupont IPv6 Stack (INRIA IPv6 stack). This stack can be downloaded from the [ipv6stack]. The file is called `New.tar.gz` and related documentation in [stackdoc](file `docs.tar.gz`) though it is not really good.

After FreeBSD 3.4 installation [freebsd34] (we installed it with `Developer`, `User`, `Kern-developer` options) we must proceed to install IPv6 stack. First place, we must gunziped an untar `New.tar.gz` file.

We execute:

```
gzip –d < New.tar.gz | (cd /; tar xvBpf -)
```

The content of this package is placed in `/usr/src/IPv6` directory. If we have a look at files contained inside, we will see files called `XXX-new`. They are the new files that support IPv6 protocol. To install them we must only execute the script file `inst-new`.

Due to an error in files inside the package, we must modify `inet.c` file placed on `/usr/src/usr.bin/netstat/` . In lines 100-102 there is next content:

```
define SRC__FreeBSD_version 340000        /* FreeBSD 3.3 */
#if __FreeBSD_version != SRC__FreeBSD_version
#error "FreeBSD version change, please control the generic IPv4 inpcb
structure"
#endif
```

If we execute next step make `world` from `/usr/src`, an error is shown during this process because our FreeBSD version is 3.4 and these lines test if FreeBSD version is 3.3. For example, we can change the condition and turn it into an always true condition.

After this, `make world` (it takes long time) is executed without problem.

Next, we must execute this command

```
config KERNEL_FILE
```

`KERNEL_FILE` is a personalized kernel file configuration. In order to know much, it is interesting to consult FreeBSD handbook about this issue[fbsdHbook].

Our kernel configuration file (`KERNEL_FILE`) contains a new option to support IPv6 and another to define a pseudo-device called `tug` (The MAP and HA opens a tug per managed mobile host. So you need a lot of tug if you want to manage many Mobile Hosts (MH). `X` is a number that signals the number of defined tugs. Furthermore, one tug must be defined in MH's kernel configuration file.

```
options INET6
option pseudo-device tug X
```

This handbook explains two methods to compile the kernel. We must use the first option, so we execute:

```
/usr/sbin/config KERNEL_FILE
cd ../../compile/KERNEL_FILE
make depend
make
make install
reboot
```

Now, we have a kernel with IPv6 support. But we must configure network interfaces with IPv6 addresses. To do this, there are two interesting files in `/etc/` directory called `rc.ipv6-ottawa` and `rc.ipv6-duvel`. First one configures IPv6 support for a host and the second one configures a server. We have made our own script file based on these one (`rc.startIPv6`) that meets all our needs.

We put this file on `/etc/` and an example for MAP machine is:

```
#!#!/bin/sh
/*This line enables kernel IPv6 forwarding*/
sysctl -w net.inet6.ipv6.forwarding=1
/*This line configures all intefaces' IPv6 link address and start up
interfaces*/
autoconf6 –av
/*These lines configures for each network interface a  IPv6 address*/
ifconfig fxp0 inet6 firstalias 2001:720:1710:10::10/64
ifconfig fxp1 inet6 firstalias 4080:0:7b2:100:4080:0:7b2:100/64
/*Configure the routes to testbed network. This information is different
between nodes*/
route -n add -inet6 3ffe:720:1710:12::/64 4080:0:7b2:100:4080:0:7b2:101
route -n add -inet6 3ffe:720:1710:16::/64 4080:0:7b2:100:4080:0:7b2:101
route -n add -inet6 2000::/3 2001:720:1710:10::1
```

The rest of files are shows in 6.1.2 annex.

3.1.1.2 INRIA's HMIPv6 implementation.

Next step is to install HMIPv6 package (`hmipv6.tgz`). This one can be downloaded from [hmipv6inria]

Firstly, we uncompress and untar the package (for example in `/usr/src` directory)

```
tar xzvf hmipv6.tgz
```

a directory called `HMIPv6.34` is created. Inside it, there are four directories and three files

`ndpd-host`. ndpd with mobility support for mobile host.

`ndpd-router`. ndpd with mobility support for routers (ie. home agent, base station, mobility agent).

`ndpd-mobility`. Mobility functions.

`ndpd-uhmma. HMIPv6 functions.`

`ip6_opts.c. A hacked kernel file.`

`in6_proto.c. A hacked kernel file.`

`ip6_opts.h. A hacked kernel file.`

Depending on node role (Base Station, MAP, Home Agent, Mobile Host) we must do different steps.

3.1.1.2.1 Base Station and simple PC-routers

We must go to the `ndpd-router` directory. We can find two `makefile` files and we must execute:

```
make –f makefile.b
```

This process produces a daemon named `HA_BS`. We must remove all `*.o` files too.

3.1.1.2.2 Home Agent

We must do the above same step. Furthermore, we must create a file called `gateway6` in `/etc` directory. This file is read by `HA_BS` daemon and it contains several configuration parameters(see annex B).

3.1.1.2.3 MAP (Mobility Agent)

To prepare `MAP` agent , we must execute next commands in ndpd-router directory

```
make –f Makefile.b (it creates HA_BS daemon)
rm *.o
make –f Makefile.a
rm *.o
```

The second `makefile.a` produces a daemon named `MA` and only this program must be executed but does not `HA_BS` daemon.

By the way, we must create the file `gateway6` under `/etc`. This file must contain:

```
#Mandatory. These lines are needed because MAP will not start if you do not
include them.
virtual prefix=4080:0:7b2:100::/64
virtual if=fxp1
virtual BS=4080:0:7b2:100:4080:0:7b2:101
virtual Root=fxp1
#These lines are optionals.
virtual Domain=8
virtual SHRet=4
timeout Forward=15
timeout Annonce=10
timeout Home=30
timeout Livetime=300
timeout Refresh=250
timeout Smooth=1
```

`virtual prefix` is the virtual prefix address and it is prefix announced by MAP agent in the MAP option. This MAP option will be send to one or several Base Stations (BS) inserted here. In our example there is only one BS. The rest of parameters are explained in [hmipv6inria], but it has not importance. To avoid problems we have included these optional lines in our `gateway6` file.

3.1.1.2.4 Mobile Host

We must go to `ndpd-host` directory and execute `make` to compile the daemon named `MH`.

Moreover, we must create a file named `mobile6` under `/etc/` directory. This file contains some configuration parameters needed to Mobile IPv6. This file has next lines:

```
#To configure MH's Mobile IPv6 behaviour
mode multi=n
mode scope=link
home agent=2001:720:1710:18::1
home addr=2001:720:1710:18:202:b3ff:fe1c:e65b
movable=fxp0
#Next parameters must be added to support HMIPv6 Micro-mobility
timeout Forward=15
timeout Position=10
timeout Home=30
timeout Lifetime=300
timeout Refresh=250
timeout Smooth=1
timeout retransmission=300
Domain Len=8
Domain RtReg=5
```

Where (HMIPv6 parameters)

- `timeout Position`. Time between two old position packet. Old position packets are sent to the Agent to refresh the registration. If the Agent does not receive a refresh, it will remove the mobile host from its managed list.
- `timeout Retransmission`. Time between two unacknowledged registration. If the mobile host does not receive an acknowledgement (after registering) it has to resend a registration. The time between two registration is specified here.
- `Domain Len`. Length of the domain prefix in bytes. The length is used to know if a `CH` is in the same domain that the MH (i.e. to send Binding Update with the GCOA or the LCOA).
- `Domain RtReg`. Number max of registration retransmissions. If the mobile performs the number of registration without to be acknowledged, it must swap in Mobile IPv6 mode.

And where (MIPv6 parameters):

- `mode multi`. Set the multi-homed multi-ISP binding management.
- `mode scope`. Set the scope of the mobility management. With `link`, connections to visited networks are not kept across movements, with `prefix`, connections to nodes sharing a visited prefix will use the whole mobility stuff, ie. will use the home address and can be kept across movements. Scopes `site` and `global` are not yet implemented nor defined.
- `home agent` . Set the home-agent address (home agent discovery is not supported)
- `home addr`. Set the home-address (currently only one home-address is supported).
- `movable`. Gives the list of interfaces which can be moved. When a totally new router advertisement (from a new router announcing new prefixes) is received on an interface then this interface is restarted (i.e. router solicitations are sent in order to know more), old prefixes and routers will be flushed if they are not announced again.

3.1.1.2.5 How to use

All daemons (`HA_BS`,`MA`, `MH`) has a new option named `-V` which turns the daemon in verbose mode. `-v` option is used to dump MIPv6 information, while `-V` is used to dump HMIPv6 information

3.1.1.2.5.1 Base Station.

```
./HA_BS –B –V –d  –D4/1
```

We must always start all base stations before MAPs. With `–Dx/y` option, we can set the router advertisements period (developers recommend 4/1). This parameter enables faster handoffs. With `–d` (debug mode) option if we use `CTRL+C` sequence dump statistics are shown on the screen.

3.1.1.2.5.2 MAP (Mobility Agent)

We must launch `MA` daemon with –a option. Furthermore, we use other options:

```
./MA -a -V –d
```

3.1.1.2.5.3 Mobile Host

We must lauch `MH` daemon with `–H` option. The mobile node can swap to Mobile IPv6 mode if no `MA` information is available, but it will always try to use HMIPv6 mode. Another interesting option is `–F` which enables fast handoff. Therefore, we execute:

```
./MH -H -V -d –F
```

3.1.1.2.5.4 Home Agent

In this case all Mobile IPv6 information is found in `/etc/gateway6` (see annex B) so we must only execute:

```
./HA_BS –d
```

with `–d` option to show debug information (`CTRL+C` sequence must be used)

3.1.1.2.6 Drawbacks

In spite of all above steps of configuration, the implementation fails in some situations. In this section we are going to explain some these bugs and solutions used.

For example, `Makefile.b` file has a compilation flag

```
 CFLAGS += -DKAME_ND6
```

Due to this value, the format of MAP option contained in router advertisement and sent by MAP is not well processed by MH. It dues to MH is able to process a MAP information with next format:

```
+-------+-------+-------+-------+
| type  | length|  na   | plen  |
+-------+-------+-------+-------+
|                               |
|            MA address         |
|                               |
|                               |
+-------------------------------+
```

But with a different structure is used `–DKAME_ND6` flag. So we had to modify `Makefile.b` file and comment this line.

But it is not enough, because the field `plen` should be a 64 value, but it is always a 0 value. So, we must solve this problem. The solution has been to modify `mb-req.c` file in `ndpd-uhmma` directory where `plen` is assigned. So we put a 64 value instead of 0.

By the way, two additional problems have been found when we executed our tests (see annex A):

- MAP hangs and reboots when several Mobile Node movements. Problem has not identified.
- All MH's IPv6 addresses except link local address must be deleted each time MH is restarted otherwise MH does not works (it does not detect network changes) in subsequent execution.

Apart from this internal problem there is some external problem that should take in account:

- The implementation is not supported by the developers anymore. Therefore, the code is pretty old now (about 2-3 years) and there is not a new version from 1999
- It uses an old SO (FreeBSD 3.4).
- It uses an old IPv6 stack (august 2000). This stack supports Mobile IPv6 but it full conformance with Mobile IPv6 draft 12 [d-mipv6-12]. Currently, the last draft is version 19. Furthermore, KAME IPv6 stack [kame] is the de-facto standard one. In fact, there are not many IPv6 tools that supports INRIA stack.
- The implementation is based on first HMIPv6 design. Later revision about it has not been taken in account (last draft version about HMIPv6 is [d-hmipv6-07]).
- The code is not completely checked (several found bugs)

These reasons are enough to conclude INRIA's implementation is not suitable as HMIPv6 implementation and it should be discarded. However, in annex A we show some information about handover time tests made. This information has been obtained to show us that implementation works in a suitable way at least. The results show that it takes around 1 second in each handover (this information is also confirmed by developers in its own documentation [hmipv6inria]).

At least, a positive aspect of this implementation it is free source code and it could be a reference for future implementations (mainly for FreeBSD).

3.2 Local mobility tests

3.2.1 PTIN

Here is an introduction to local mobility tests performed in PTIN. A more detailed description can be found as an appendix.

The general testbed used in the tests described in the next sections is presented in Figure 3-1. There are 3 different networks with prefixes 3ffe:3103:0:8::/64, 3ffe:3103:0:9::/64 and 3ffe:3103:0:10::/64. Between the 3ffe:3103:0:8::/64 and 3ffe:3103:0:9::/64 networks, we have a Linux router acting as HA entity, and between the 3ffe:3103:0:9::/64 and 3ffe:3103:0:10::/64 networks, we have a Telebit router. Both, Linux and Telebit routers are announcing their respective networks, sending Router Advertisements (RAs). I.e., the Linux HA router is announcing to the 3ffe:3103:0:8::/64 network (*radvd* daemon), while the Telebit router is announcing in the 3ffe:3103:0:9::/64 and 3ffe:3103:0:10::/64 networks.

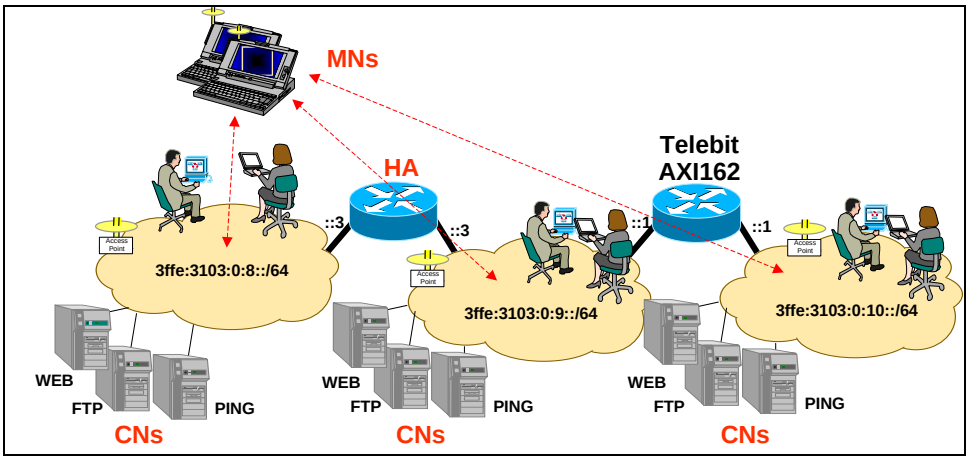


Figure 3-1: General Testbed used for MIPL tests.

In each network we have some simple applications like pings and traceroutes, and other services like www, or ftp (servers and clients). Either the MNs or the CN we use, have support of these applications/services. The MNs used, have their HNs on the 3ffe:3103:0:8::/64 network.

This general scenario allows us to test different technologies such as LANs (802.3) and WLANs (802.11). The attachment to all networks can be made either via wired LANs or wireless WLAN. There are two sets of scenarios: the first ones where MN are attached to LANs and the second ones where the MN uses WLAN.

3.2.1.1 Implementing the tests

In this section we describe the scenarios we have tested using the previously described testbed. Not all scenarios are represented, but just the ones that really represent different and interesting situations.

The configuration used for each entity was always the same and can be viewed in the Annexes **6.3.1**. The Annex 6.3.1.1 shows the HA configuration, the Annex 6.3.1.2 shows the configuration for the different CNs used and, finally, the Annex **6.3.1.3** shows the configuration used for the MNs.

To analyze the flow of the protocol information exchange, we have used the Microsoft Network Monitor (*netmon*), which allow us to recognize some of the new information related to mobility like Destination Option Header codes (e.g. Home Agent or Binding Updates), among others.

The tests can be divided in two categories. In the first ones, a wired technology is used to attach the MN to the links, while in the second ones, the wireless technology has been used to attach the MN. For these both scenarios, similar tests have been performance, in order to allow assess the differences between of them.

3.2.1.1.1 LANs testing

In these tests, we use the wired LAN technology to attach all the nodes to the network. The hand-off is made changing the cable from the hub of one network to another.

It is important to notice that the external behaviour we present in the following tests, must be analyzed taking into consideration that some time was spent on these operations and, actually, should not be considered as a handoff.

3.2.1.1.2 WLANs testing

In these tests, we use both, the wired LAN technology and the wireless WLAN technology in order to attach the nodes to the network. The WLAN technology will be always used to perform the handoffs in the MN, in order to establish a comparison between the previous tests.

The way the handoff is made is using the command:

iwconfig ethX essid NAMEWLAN key XXXX-XXXX-XX ap YY:YY:YY:YY:YY

where:

- *ethX* is the wireless interface;
- *essid* is the network name identification (SSID);
- *key* is the WepKey encrypt the data to be sent; and
- *ap* is the Access Point MAC address of the AP.

When the handoffs are performed, the MN is covered by the radio signal of both AP (Access Points) of each links. So, the command will attach and detach to one or another network depending on the parameters used.

3.2.1.2 Conclusions

The IP mobility capabilities are very important in order to provide to the devices the necessary transparency to the upper layer protocols like TCP, UDP, and applications when changing their poit of attachment to the network. This way, the movement of a Mobile Node (MN) between different links doesn't imply either a restart of a connection, or the reconfiguration of the network parameters. On the other hand, the node could be always contacted, thanks to the use of the same IPv6 address as the MN identification in each link. Actually, the mobility does not provide the "all-IP" capabilities at all, based in the fact that to implement this it is necessary to reduce considerably the handover time, which can be made using Fast-Handover mechanisms over the Mobility.

3.2.1.2.1 MIPL tests

In this document we described the tests of one MIPv6 implementation, running on the free Linux platform. The MIPL (Mobile IPv6 for Linux) implementation is one of the most tested in the world right now, and offers an interesting set of functionalities based on the draft version number 15 (version 0.9.1).

This document has started talking about how the Mobile protocol, defined by the IETF *mobileip* Working Group works. After that, a typical testbed used for the tests has been presented, referring the main existent equipment and links. The tests have been separated in two main components: the first one where the handover was made with LAN technologies (Hubs), and the second one, where the handover was made using Wireless LAN (WLAN) technologies. For each component, five main scenarios were tested, representing the main situations of where IP Mobility can play an important role. From the situation where the MN is in the HN, to one where the MN is in a FN, or another where two MNs are present, and each one either in its HN or in a FN.

As expected, the main results we achieved between the two sets of scenarios were the handover times. In fact, these times are very different, being higher for the LAN technology. Of course, this is because the WLAN technology is faster than the LAN recovering the signal and starting the communication. In the LAN, when the cable is unplugged, the interface goes "down" and the restart is longer. Anyway, the approximated time interval for the handover presented, can hide the truth, because we just refer the time during which the application are not responding. However, in the LAN handover, some time should be subtracted in order to take in consideration the time in which we change the cable from one Hub to another.

In terms of functionality the tests performed, show that the implementation is working properly. However, the implementation still has some instability in some situations. Sometimes, after a period of tests, mobility seems that goes down, and in some cases the connection between the entities are not restarted, and is necessary to restart the mobility or even the machines in order to have them working properly again. For example, one thing that many times makes the mobility crash is when the handover is performed very fast consecutively. If the times that a MN is attached to each link is less than 4 - 5 seconds, usually the MN crash easily.

Finally, and despite some instability, we can conclude that in general, the Linux implementation is working correctly. Another interesting property, which is very important for researchers, is that this implementation follows very near the IETF evolutions.

3.2.1.2.2 Windows, Cisco and MIPL tests

The interoperability tests showed to us what are the implementation and the functionalities that can work together. The interoperability tests had since the beginning and extra-complexity, taking into account that not all the platforms implement the same draft version - Linux implements the draft 15 and, Windows and Cisco implement draft 13.

Basically the tests were made covering almost all the possible combinations between the platform and the functionalities.

Firstly, we started the interoperability tests using the Cisco HA, and testing the Windows CNs and MNs. What we concluded was that in every situation the scenarios work as expected and the packet exchange were the correct. We can say that for the main functionalities at least, the implementations are completely interoperable. However, this is an expected result, taking into account that both implement the same draft version - 13.

The second interoperability tests we performed used a Linux HA, and the CNs and MNs were either Linux or Windows. The results obtained were very interesting, because we concluded that even implementing different draft version, with considerable differences between them; the implementations are compatibles in all circumstances. Of course this result can only be achieved if we don't use the authentication mechanisms on the Linux (authentication is optional in draft 15).

Finally, we have tested a Cisco HA, with CNs and MNs, either been a Linux or Windows implementations. In this case not all the functionalities inter-operate with each others. What we concluded was that the Linux CN can work with Windows MN and Cisco HA, but the Linux MN doesn't work with the Cisco HA. This happen because the fields related to the BU header changed from the draft version 13 to 15, and the value of the prefix length are bad interpreted. We have already noticed that the Windows, implementing already the draft 13 interprets correctly the Linux Bus. This is due to the fact that the Windows doesn't check some fields, using just the CoA of the packets. One doubt left about this fact. Perhaps the Windows implementation was changed in order to inter-operate with draft 15. This is still unknown.

3.2.2 UPM

The scenario used for local tests at UPM is shown in Figure 3-2:

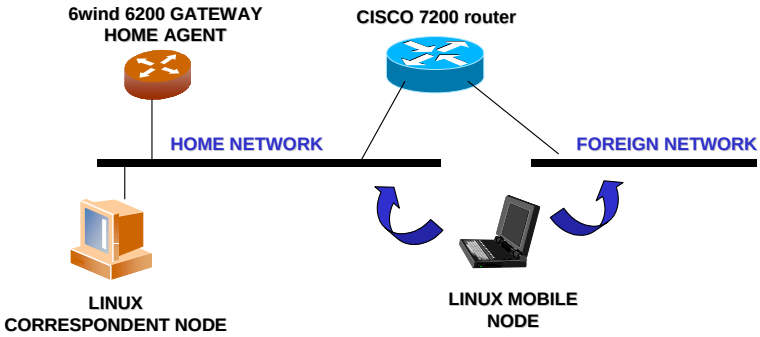


Figure 3-2: Scenario for local mobility tests at UPM

In this scenario it has been used the linux MIPL [mipl] implementation for Mobile and Correspondent Node (mipv6-0.9.2-v2.4.18 release) and the Home Agent implementation of the 6wind 6200 gateway. The objective of these tests was to test the interoperability and performance of both implementations.

The Home Agent [6wind-ha] is configured as follows:

```
r6w6200{}edit conf mipconfig
WARNING : The home-agent function is an advanced feature ( not fully tested )
r6w6200{mipconfig}mip
r6w6200{mipconfig-mip(advanced and not tested feature)}mobility home_agent
WARNING : The home-agent function is an advanced feature ( not fully tested )
```

The Linux Mobile Node is able to register with the 6wind Home Agent when it visits the Foreign Netwtork, as it can be seen in the log of the Mobile Node:

```
mnode:~ # mipdiag -m
Home Address                                    Home Agent Address
          Care-of address                                 Registered
2001:800:40:2b01:2e0:7dff:feba:3913             2001:800:40:2b01::a088
          2001:800:40:2b03:2e0:7dff:feba:3913           1
```

It means that the Linux MN interacts with the 6wind Home Agent, getting registered and going on with its connections. As a result, the 6wind MIPv6 implementation is compatible with MIPL one. The Mobile Node is not using Authentication Headers, since the 6wind Home Agent provides draft 13 supports, where the use of this kind of authentication is not included.

The Mobile Node is not using reverse tunneling in this case, and the Router Solicitation interval is configured to 1 second.

Two applications have been used for performance measurements. The first one is a traffic generator, mgen, ported to IPv6 within the Long IST Project [mgen6]. Using this tool, the Correspondent Node acted as traffic emitter and the Mobile Node as receiver. Figure 3-3 shows the traffic received in a 180 seconds test, where the emitter was sending a constant bit rate traffic, and the receiver (Mobile Node) was changing its location every 20 seconds.

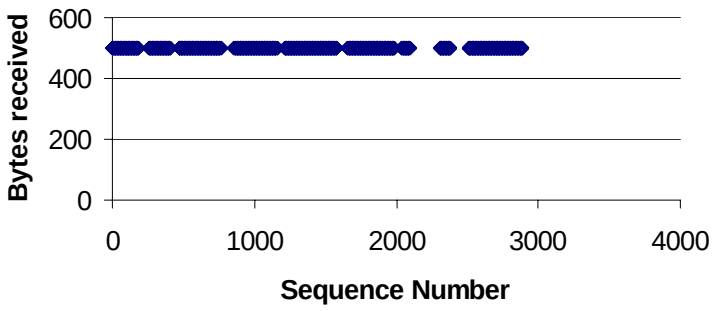


Figure 3-3: Traffic received by the Mobile Node while changing its location

After each handover, the Mobile Node stops receiving traffic, as it can be appreciated in the graphic. After registration has succeeded a few seconds later goes on receiving. The time elapsed without receiving traffic after each handover is variable. Since the kind of traffic is UDP, the traffic corresponding to this period of time is lost.

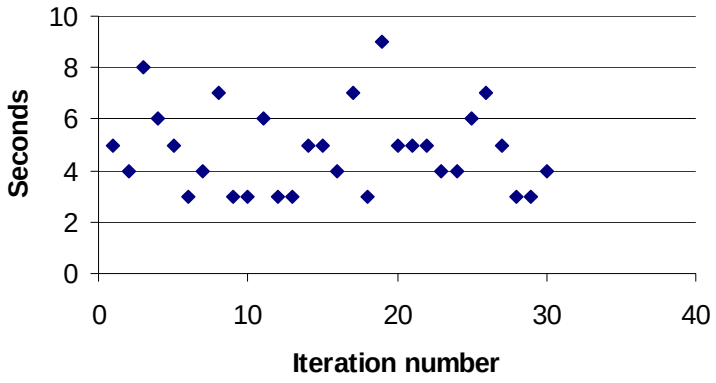


Figure 3-4: Time elapsed after handover in 30 consecutive handovers

The value of this time in seconds is represented in Figure 3-4, for 30 consecutive movements from Home Network to Foreign Network and vice versa.

It can be seen that this time is not constant, and ranges from 3 to 9 seconds. The gateway in both visited networks is a CISCO 7204 router. The router advertisement interval and lifetime at both networks is set to the minimum according to the related RFC, 3 seconds. It means that the minimum theoretical value for the handover time would be 3 seconds for this testbed, so the results obtained are coherent. The average value of the handover time according to the tests is 5 seconds.

The other application used for performance measurements has been a multiconferencing application, Isabel. These tests are described in the annex of the correspondent subactivity.

3.2.3 TILAB MIPv6 implementations comparison

3.2.3.1 MIPv6 imlementation analysis

This document takes into account the tests carried out during the first semester activity inside the A4.1 sub-activity and concerning the analysis of different MIPv6 implementations and the related tests.

During our activity, the implementations that have been tested are: the MIPL implementation (developed in Linux software environment) and the LandMARC implementation (developed for Windows2000).

Both of them have been analyzed from many points of view. The goal was infact both to understand if the respective implementations were effectively conform to the MIPV6 standards, both also to verify that they were application transparent. Infact it is important in the same way that the applications were always active although the mobility of node inside the network.

Dislike the IPv6 protocol, Mobile IPv6 implementations are not still stable and they can be mostly considered as prototypes always updated to be more conform to the new versions of the protocol.

The reasons why two above implementations have been chosen among those one already available are basically that MIPL, for LINUX environment, is a mature implementation that guarantees a good conformance level to the MIPv6 standards.

The LandMARC implementation has been selected, although the maturity level is not as good as the other one, why Windows is now the most popular end-user operating system all over the world and it could be useful to verify the stability and the functionalities already supported.

3.2.3.2 Overview

This chapter if mainly focused on the description of the MIPv6 implementations used in our tests, underlining the main features and the main known bugs or limitations. The test bed used for the experimental activity will also be shown.

In a mobility scenario the entities that have to support the MIPv6 protocol are the Mobile Node, the Correspondent Node and the Home Agent. Generally speaking, the operating systems that can be used are Windows 2000, FreeBSD and Linux. Our choice was to take into account the MIPL implementation for LINUX and LandMARC implementation for Windows 2000.

MIPL is an implementation widely tested and conforms to the new version of the MIPv6 protocol. The LandMARC implementation is, on the other hand, the first experimental version developed by Microsoft Research, and it is not as mature as the MIPL. However, as it was argumented before, we decided to test it because Windows is the most popular end-user operatying system and it could have been interesting to test its functionalities.

Note that MIPL implementation used is conformed to draft-ietf-mobileip-ipv6-13.

Note that in both of tests we made, we have installed not the patch but the stable versions of the implementations.

The guiding criterion of the tests was to demonstrate that, moving from an home network to a foreign network, the transport layer connection and application were always active. Both of the MIPv6 implementations support the functionalities of Mobile Node, Correspondent Node and Home Agent. Being two different implementations and conform to different drafts, we decided to test them using two different test-bed, one for Linux scenario and the other for Windows 2000 scenario.

The test scenario is characterized by an home network (i.e. a link where Mobile Node and its HA belong), a foreign network (where the mobile node arrives, when it leaves the Home Network), a third network where Correspondent node is placed (to use as foreign network, when testing the movement of Mobile Node from a foreign network to the other foreign network).

Note that in both test-beds the Home Agent functionality has been implemented not in a router but in a host with only one network interface. In other words, the HA is a node placed inside the home network that sends over the link the Neighbor Advertisement messages and acts as Proxy Neighbor Discovery for those nodes out of home network, creating a tunnel towards the mobile nodes out of the home network. This way we have realized a more realistic scenario where access routers have an operatying system independent of that one supported by the mobile nodes.

The implementations under test provide the security support for the signaling procedure security, but this support has not been enabled during our tests since this support has been implemented in a different way in the next MIPv6 draft (version14 and 15).

All the tests have been carried out using wired link and Ethernet interfaces, consequently the movements of the mobile node have been realized disconnecting by hand the laptop and connecting it to different networks. Infact, to verify the conformance to the implementation, is not important to use wired or wireless link.

3.2.3.2.1 ScenarioA: LandMARC Mobile IPv6 for Windows 2000

This implementation developed by Microsoft Research, together with Lancaster University, has been tested with the stack IPv6 Windows 2000 Technology Preview. Among the main supported features of this implementation we remind:

- Mobile Node, Correspondent Node and Home Agent functionalities are implemented
- Binding cache is implemented on Home Agent and Corresponding Node
- Binding Update List is implemented on Mobile Node
- Route Optimization mechanism is supported
- Movement detection mechanism is based upon the Router Advertisement message and NDIS media connect signaling
- Multiple home address configuration
- Interoperability with MSRIPv6 IPSec to guarantee the protection of the control signalling

Known bugs and limitations are:

- Hand-off times strongly depending on the driver and network adapters
- IKE is not supported
- Multi-homed nodes are not supportd
- High RTT when IPSec is enabled
- Low performances in route optimization when handing-off during a TCP or UDP session with high load
- On some machines the autoconfiguration procedure has to be activated by hand.

3.2.3.2.1.1 LandMARC Test bed

In the next picture the test scenario for Windows 2000 implementation is shown:

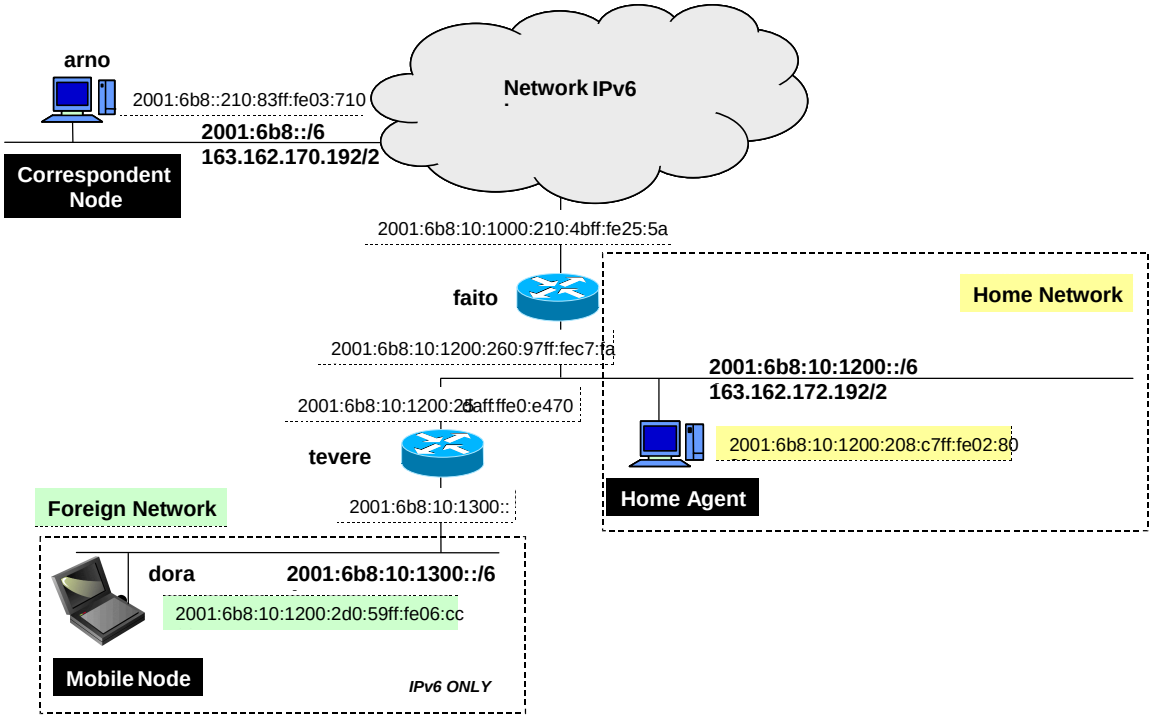


Figure 3-5: Windows 2000 testbed

3.2.3.2.2 Scenario B: MIPL Mobile IPv6 for LINUX

The version we tested in our experimentation was the 0.9.

In the following we summarize its main features:

- Conform to the draft-ietf-mobileip-ipv6-13
- It has been tested only on RedHat distributions 6.1 up to 7.1 with kernel 2.4.4 up to 2.4.7, but it should work wit any kernel 2.4.x with IPv6 support enabled
- Mobile Node, CN, and HA functionalities are implemented. CN functionality is available also in the MN and HA
- IPSec Autentication Header is used to assign the keys in order to authenticate the signalling procedures. Secret Key management shared by CN-MN is obtained by a suitable key management tool (Pluto), belonging to the FreeS/WAN IPSec packet
- Dynamic Home Agent Address Discovery mechanism is supported

Main bugs and limitations are reported in the following:

- Dynamic key assignment between MN and CN is not implemented
- Home Agent does not support the Proxy DAD related to the home address of Mobile Node

3.2.3.2.2.1 MIPL test-bed

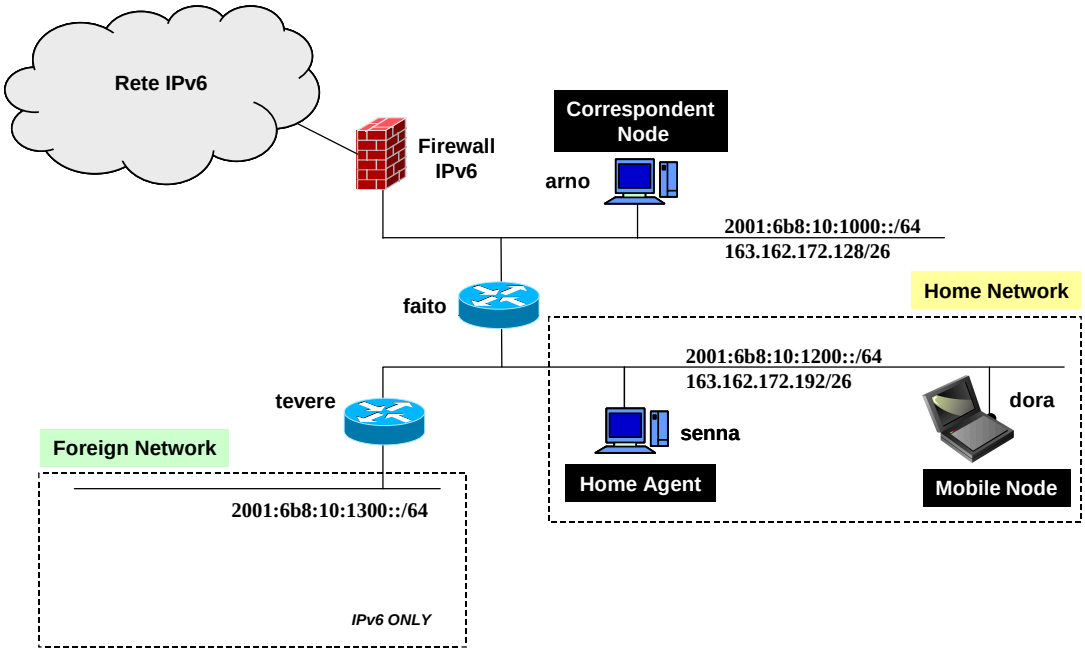


Figure 3-6: Linux test bed

MN, HA e CN:	Linux RedHat-7.1-kernel 2.4.7
	MIPL Mobile IPv6 for Linux-0.9-v2.4.7
Home Network Access Router	NetBSD 1.5
(`faito`):	
	Zebra Routing Daemon v-0.89a
	Router Advertisement Daemon radvd-0.7.0
Foreign network Access Router	*Dual-boot*:
(`tevere`):	- FreeBSD 4.3
	Zebra Routing Daemon v-0.92a
	- Linux RedHat-7.1

After analyzing two implementations, we can individuate the difference between two implantations in order to understand the difference. Test we realized with MIPL implementation are almost the same as the Windows 2000 implementation. Only difference is that with MIPL we made one more test where Mobile Node moves from to a foreign network to another foreign network. This is not possible with W2000 since it does not support the situation where MN e CN are in the same network.

3.2.3.3 Testing Activity

Main goal of Mobile IPV6 is to let the connection active when the mobile node changes its Internet access point in order to avoid that the connection already active goes down. Hence the goal of the experimentation was to verify the reachability of mobile node after a movement and to verify what happens at application and transport layer.

We can individuate three types of test:

- Network layer mobility tests
- Transport layer mobility test
- Application layer mobility test

3.2.3.3.1 Network layer test

Description	W2000	MIPL
The Mobile Node is inside the foreign network while a ping6 from CN to the MN (and vice-versa) is active.	OK[1]	
The MN is inside the home network. While a ping6 from CN to the MN (and vice-versa) is active, the mobile node moves to foreign network.	Some problems[2]	OK[3]
MN is inside the *foreign network*. A `ping6` from CN to MN (and vice-versa) is executed, and during the ICMPv6 messages exchange, MN is disconnected from *foreign network* and come back on the *home network*.	OK[4]	OK[5]
MN is on the *home network*. During a `ping6` between MN and CN, mobile node moves to *foreign network1*, and subsequently on the *foreign network2*		OK[6]

Notes:

1. OK: Route Optimization mechanism works properly for both implementations

2. HA does not implement properly *Proxy Neighbor Discovery procedure*, infact HA starts to receive the packets going to the MN, forwarding them to the CoA via *tunneling*, with some delay referring to the Binding Update (from MN) reception time. *Route Optimization* mechanism works fine. If a `ping6` is executed from MN to CN, we note a second problem. Infact, following to the movement of MN, MN sends Echo Request on packets that have as source address the *home address*,instead of COA

3. As soon as movement is dected BU/BAck are exchanged with HA, and then packets exchange from MN to CN starts again immedately. Both Route Optimization and Proxy Neighbor Discovery mechanisms work fine.

4. Following to this movement, there is a temporary loss of connectivity at network layer, but subsequently the exchanges of Echo Request and Echo Reply messages work again properly

5. Following to this movement, there is a temporary loss of connectivity at network layer, but subsequently the exchanges of Echo Request and Echo Reply messages works again properly.

6. ping6 is active during both movements, and in both cases Route Optimization mechanism works properly.

3.2.3.3.2 Transport Layer Test

Description	W2000	MIPL
MN is inside *foreign network*. A `ttcp6` is executed from CN to the MN.	OK[7]	
A `ttcp6` from CN to the MN is executed, while MN moves from *home network* to the foreign *network*.	Some problems[8]	OK
A `ttcp6` from CN to MN (and vice-versa) is executed, while mobile node goes from *foreign network* to the *home network*.	OK	OK[9]
MN is inside the *home network*. During the execution of a `ttcp6` from MN to CN, MN moves to *foreign network 1*, and subsequently to *foreign network 2*		OK[10]

Notes:

7. Routing optimization mechanism works properly.

8. When time between the transmission of two consecutive TCP packets is not defined, session usually breaks. When this time increasing, TCP session remains active and mobile node receives all the packets transmitted from CN. Route Optimization mechanism has some problems due to the fact that MN inserts in the packet sent to foreign network, its home address as source address, instead of Care-Of Address.

9. TCP session remains active and ends correctly, although the MN has the same problems with ping6 in the same scenario.

10. In both cases TCP session remains active and Route Optimization mechanism works in the right way.

3.2.3.3.3 Application Layer Test

3.2.3.3.3.1 W2000 implementation

Description	Result
HTTP: in this test, on the mobile node, an HTML page is opened (with an automatical redirection to another page) while MN moves from *home network* to the *foreign network*, and then came back to the *home network*.	Some problems[11]
HTTP: in this test, we try to download a file in the MN from CN, while MN moves from *home network* to *foreign network*, and vice-versa.	NO[12]
HTTP: audio streaming via http of a MP3 file stored inside the CN, while mobile node mobile moves from *home network* to the *foreign network*.	NO[13]
TELNET: MN is the telnet *server* and CN is the *client*. While MN is in the home network, a telnet session is opened. Then the mobile node moves to *foreign network*, and at the end comes back to *home network*.	NO[14]

Notes:

11. Following to the various movements application is always opened and delays are well-accepted. In other words, Internet Explorer continues to show a page, and to ask for the other one to the server, up to connectivity is not recovered and the other page is shown. Route Optimization mechanism does not work.

12. The application is not usually tolerant to the delays for recovering the transport layer connectivity and it stops. When application has remained active, following that movement of mobile node , it has been noted that the exchange of the packets is similar to that one observed with html pages download.

13. See the previous test

14. This application usually is not tolerant to the delays to recover the connectivity at transport layer and it stops (unsolicited Neighbor Advertisement is not sent by HA).

3.2.3.3.3.2 MIPL implementation

Description	Result
HTTP: in this test, we try to download a file in the MN from CN, while MN moves from *home network* to *foreign network*, and viceversa.	OK[15]

TELNET: MN is the telnet *server* and CN is the *client*. While MN is in the home network, a telnet session is opened. Then the mobile node moves to *foreign network*, and at the end, comes back to *home network*.	OK[16]

Notes:

15. Session remains open during the mobile node movements and after the updating of binding cache in Home Agent and CN, the file *download* is recoverd correctly.
16. See the previous test

3.2.4 Consulintel

3.2.4.1 Mobility tests

3.2.4.1.1 Introduction

IPv6 includes many features for streamlining mobility support that are missing in IP version 4, including Stateless Address Autoconfiguration and Neighbor Discovery [rfc2460], [rfc2461] and [rfc2462].

Mobility support in IPv6, as proposed by the Mobile IP working group, follows the design for Mobile IPv4. It retains the ideas of a home network, home agent, and the use of encapsulation to deliver packets from the home network to the mobile node's current point of attachment. While discovery of a care-of address is still required, a mobile node can configure its care-of address by using Stateless Address Autoconfiguration and Neighbor Discovery. Thus, foreign agents are not required to support mobility in IPv6. IPv6-within-IPv6 tunnelling is also already specified.

All IPv6 nodes are expected to implement strong authentication and encryption features to improve Internet security. This affords a major simplification for IPv6 mobility support, since all authentication procedures can be assumed to exist when needed and do not have to be specified in the Mobile IPv6 protocol. Even with the security features in IPv6, however, the current working group draft for IPv6 mobility support specifies the use of authentication procedures as infrequently as possible. The reasons for this are twofold. First, good authentication comes at the cost of performance and so should be required only occasionally. Second, questions about the availability of Internet-wide key management are far from resolved at this time.

Mobility in IPv6 (MIPv6) is a “work in progress” field, so all the documents about this topic must be considered just like a snap-shoot of that’s moment status [ietf-mip19]. Revisions and updates of mobility’s documents are essential.

With this in mind, we have to establish an actual status either of the protocols (drafts) and the implementations (that follows a concrete version of the drafts).

There is a lot of literature covering the basic concepts involved in Mobility over IP, and over IPv6. Further information in [ietf-mip] and [rfc2002].

3.2.4.1.2 Designing the test bed

For the purpose of trials and test-beds in MIPv6 in this contribution, we only have considered basic Mobility.

Basic mobility allows a MN to move from one network (home) to other (foreign) without loosing the possibility of being accessible by a CN that knows the home network address of the MN. Also involves the mechanisms for Route Optimization using binding updates.

Unfortunately **security issues** related to mobility could not be tested in this contribution due to the limitations in this field of the ICR IPv6 stack.

Our basic test-bed had three different networks, connected by mean of an IPv6 “cloud” (See next figure). With this scenario several functionalities worked.

The CN (Correspondent Node) is placed in one of them, the CNN (Correspondent Node Network). The MN (Mobile Node) has its own HN (Home Network) with his HA (Home Agent) on it. The third network, FN (Foreign Network), should be used to test the movement of the MN to it, and also the return to HN, also the movement of the MN to the CNN from HN or from the FN.

This contribution wants to be a survey about MIPv6 on Windows, so all above functionalities should be implemented on Windows. However, as were already said there is not a complete MIPv6 on Windows that supports all functionalities.

Until now, only MN and CN have been developed on Windows as follow:

- MN. Entity supported on the ICR IPv6 stack. Also Windows CE supports it but it has not been tested in this contribution.
- CN. Entity supported on the new operating systems Windows XP and Windows .NET.
- HA. Entity implemented on Linux.

Therefore, in order to do tests of MIPv6 under Windows it is necessary to make an interesting mixture of different implementations which let us check interoperability with them.

Finally, the next implementations have been chosen in order to do MIPv6 tests on Windows:

- MN: ICR IPv6 stack.
- CN: Windows .NET.
- HA: HUT Linux, version 0.9.4

and the test-bed diagram remains as follows:

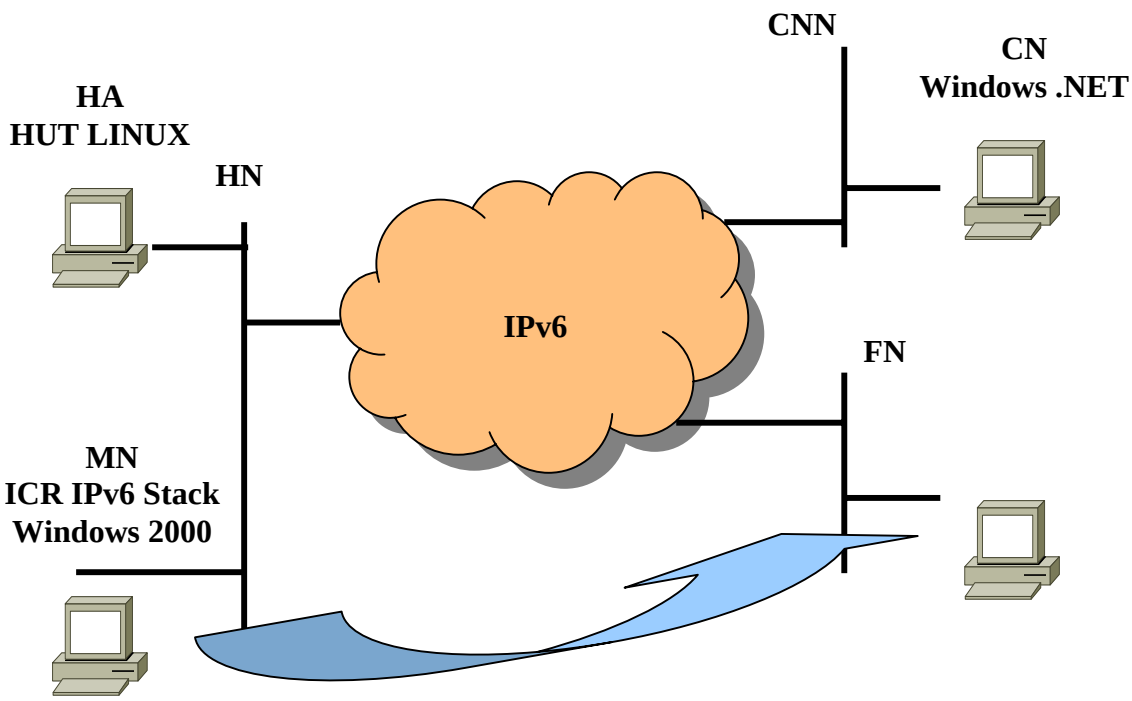


Figure 3-7: Basic Mobility Windows Test-bed

3.3 Distributed mobility tests

3.3.1 Distributed mobility tests between UPM and BT

The scenario for these tests is shown in

Figure 3-8. The Home Agent is provided by BT. The implementation used for all the nodes has been MIPL v0.9.4.

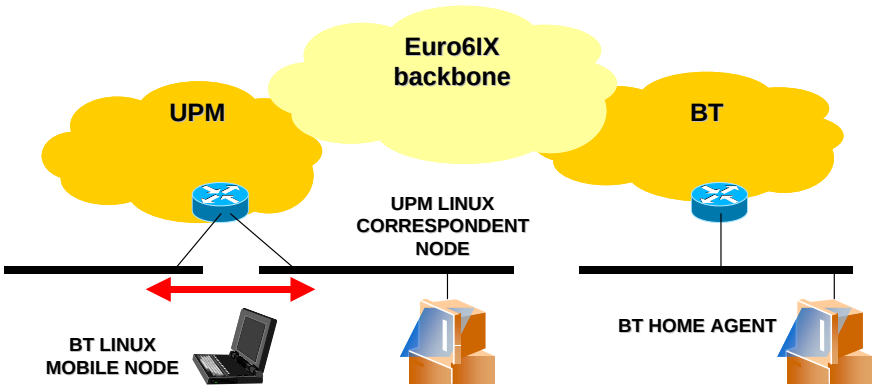


Figure 3-8: Scenario for BT – UPM Multipartner IPv6 Mobility Tests

UPM has configured a Mobile Node with a Home Address belonging to BT, and has attached it to its network. As a result, the Mobile Node has obtained a CoA belonging to UPM's network from the Router Advertisements, using the IPv6 auto configuration mechanism. The gateway in both visited networks is a CISCO 7204 router. The router advertisement interval and lifetime at both networks is set to the minimum according to the related RFC, 3 seconds.

The performance measurements have been exactly the same as those performed in the previous scenarios.

Mgen6 utility has been used again, acting the Correspondent Node as emitter and the Mobile Node as receiver. Figure 3-9 shows the traffic received by the Mobile Node, when the Correspondent Node is sending a constant bit rate and the Mobile Node is receiving it while it changes its allocation.

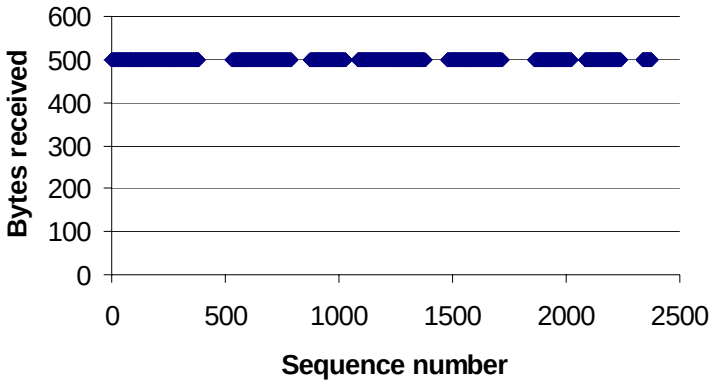


Figure 3-9: Traffic received by the Mobile Node

Figure 3-10 shows the handover times in successive movements of the mobile node. Again this time is variable, and the values it ranges are the same as before.

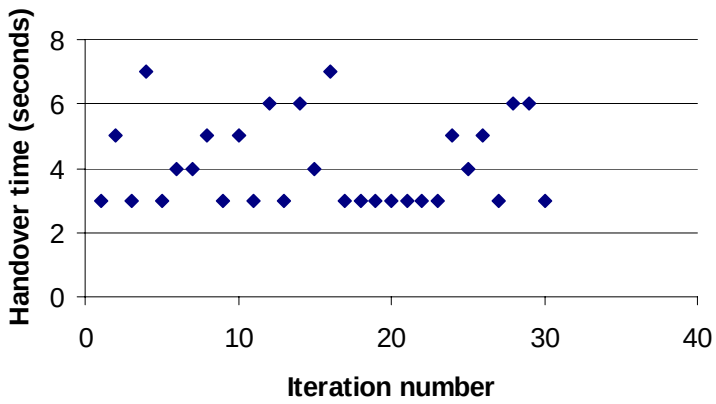


Figure 3-10: Time elapsed after handover in 30 consecutive handovers

The Mobile Node, once has detected it is not on its Home Network, sends a Registration Request to its HA, located in BT. It checks that the MN's Home Address is on its Access List, and then sends back a Registration ACK to the Mobile Node.

3.3.2 Distributed mobility tests between UPM and UMU

The scenario for these tests is shown in Figure 3-11.

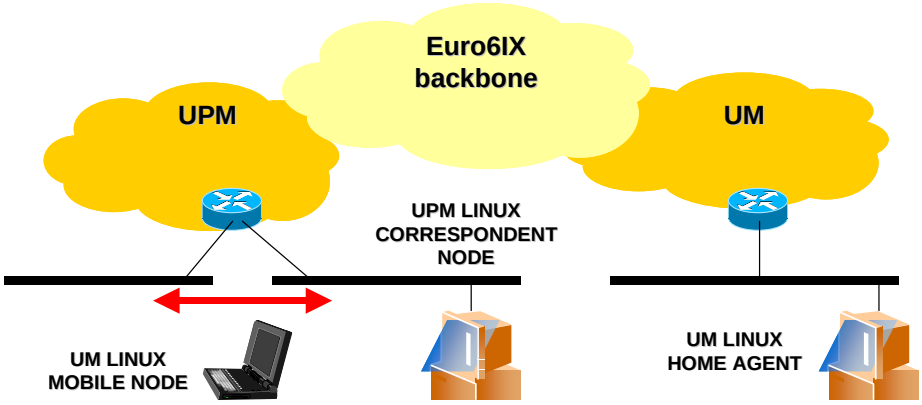


Figure 3-11: Scenario for UMU-UPM Multipartner IPv6 Mobility Tests

The implementation used for all the nodes has been MIPL v0.9.4. The situation is very similar to the previous one, the only change is that the Home Address is now in UMU, and therefore the Home Address of the Mobile Node belongs to UMU.

The performance measurements have been exactly the same as those performed in the previous scenarios.

Mgen6 utility has been used again, acting the Correspondent Node as emitter and the Mobile Node as receiver. Figure 3-12 shows the traffic received by the Mobile Node, when the Correspondent Node is sending a constant bit rate and the Mobile Node is receiving it while it changes its allocation.

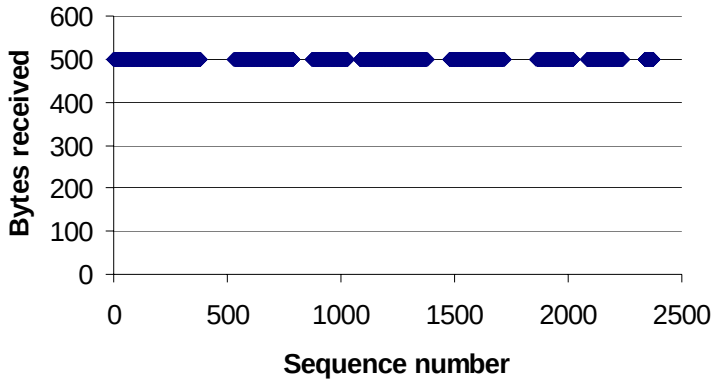


Figure 3-12: Traffic received by the Mobile Node

Once again, the UDP traffic is lost for a few seconds after the handover.

Figure 3-13 shows the handover times in successive movements of the mobile node. Again this time is variable, and the values it ranges are the same as before.

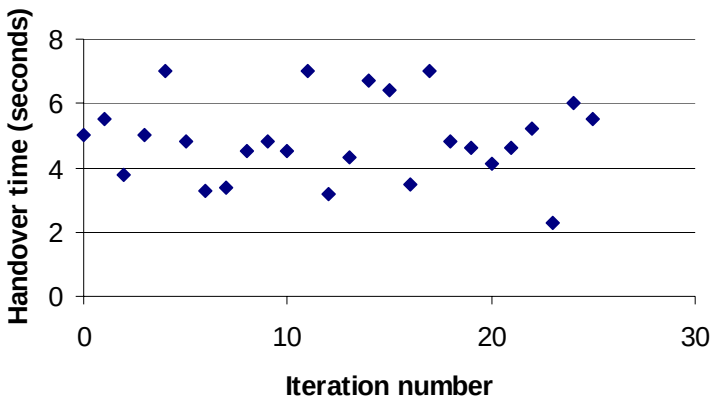


Figure 3-13: Time elapsed after handover in 25 consecutive handovers

4. FUTURE WORK

Several issues must be handled in next semesters:

- Since several partners have developed a MIPv6 testbed, more multipartner tests are being planed as part of internal trials. The idea is to configure a Mobile Node with a Home Address of other partner, as it has been done in this tests. Performing these tests will be a test for the project's network as well.

- Several implementations based on different operating systems have been used in different partner's testbeds: Linux's HUT, CISCO, Windows, 6wind, etc. Some partners have performed interoperability tests between them. Analyzing and comparing features between all of them, and some others. Mobile IPv6 implementations and criteria will have to be defined.

- Obtaining throughput measurements from this implementation.

- Analyzing IPv6 micro mobility and different implementations

- To design and implement a first micro mobility testbed based on the experience obtained from these first testbeds about macro mobility.

- To analyze how micro mobility could be integrated with Mobile IPv6 and how it could help to increase MIPv6 performance.

- To analyze AAA framework and study and how could it improve MIPv6 security.

4.1 MIPv6 area

The future work to be made in this area will be to test the remaining main IP Mobility implementations. At the same time, the test of interoperability between them is very important in order to assess the always-important capacity of the different equipment to work together. So far, this is a very hard work, due to the large amount of draft versions, and the different stages of development done by the vendors. This work, can be perhaps more effective when the MIPv6 becomes a RFC, and the implementations converge to the same point.

Important issues for MIPv6 are other technologies, defined by the IETF that complements the main protocol. A good example are the Fast Handover and the HMIPv6 implementations, which allow the protocol to minimize the handover time, being an aspect of critical importance in applications with real-time constraints, like voice, video, among others. The test of both implementation is a work that we intent to do in the near future.

On the other hand, IP mobility will be a very powerful technology, supported by the evolution of the public wireless networks. However, it is necessary to integrate this technology with many others, making it inter-operate, and taking advantage of mixing technologies. One example can be the capacity to provide mobility facilities for VPN clients. This kind of mix scenarios are not too much explored, but could be extremely interesting in the future. Actually, the mobility facilities for VPN clients, is a work that we intend to perform in the near future.

Already very important and not very much studied yet, is the business model that the ISPs can use to provide mobility services to their clients. Usually the mobility tests are performed at a low scale, and in local networks. However there it is not clear how it can be made in a higher scale, and how the deployment of the different mobile entities can be. This is also a study that we will perform in the near times. Also the unavoidable billing issues are not very clear yet. The mobility will increase the difficulty to bill the customers, because it became very difficult to know the amount of bandwidth they are using.

Our intention is to continue to work in the Mobility area during this project in order to accomplish the items above referred.

5. SUMMARY AND CONCLUSIONS

The tests carried out by PTIN over Linux MIPL and Windows implementations have shown that the analyzed implementations have a different stability and maturity level.

Particularly, the Microsoft implementation is still at a basic stage and far from providing a mobility support transparent to the applications. Many bugs have been individuated and it is still unstable.

On the other hand, test carried out over MIPL showed that this implementation has a good maturity level, since it is in accordance to the MIPv6 drafts and provides a mobility support transparent to the high layer protocols.

The main conclusion in Consulintel contribution is clear: **There is no compatibility between ICR IPv6 Stack for Windows 2000 and HUT implementation for Linux** because first one is based on Mobility Internet draft 13 whereas last one is based on draft 15. Different formats in Binding Update and Binding Acknowledgment messages in both drafts makes care-of-address updates between Home Agent and Mobile Node impossible. Furthermore, a strange behavior in the ICR IPv6 Stack was detected when mobility functionality is started. After a few seconds the stack seems losing connectivity and no packets are sent/received from/to Mobile Node and the global care-of-address becomes to value `0:0:0:0:0:0:0:0`. The “feeling” after the tests is that the mobility support in Windows is not a maturity product but experimental in most cases. At present, there is not any implementation with all mobility functionalities but only implementations with Correspondent Node or Mobile Node and any with Home Agent.

About UPM results, 6wind and MIPL implementations compatibility has been proved. Also, interoperability tests with other partners have been performed, concretely UMU and BT.

TILAB trials have demonstrated that Windows 2000 implementation has more problems and bugs than declared from Microsoft Research. It seems that are mainly the functionalities that work wrong:

- HA, after receiving a Binding Update from MN, moving from HN to a FN, doesn’t send an *unsolicited* Neighbor Advertisement in charge of MN to all nodes on the *link*;

- MN, coming back to Home Network and receiving a Binding Acknowledgement from Home Agent, doesn’t send an unsolicited Neighbor Advertisement to all nodes on the link;

- MN ignores the messages Redirect sent from the router;

- Immediately after the move of MN to the *foreign network*, there is a transient time during it uses the *Home Address* as source address of packets transmitted, even if it has correctly configured its new COA and sent the Binding Update to the HA

Probably with the testbed suggested by Microsoft, where MN and CN are on different subnets linked to the same router which works as HA too, and the movements are only on these two links, the implementation works better. In fact, that scenario doesn’t have the first two problematic items.

6. APPENDIXES

6.1 Appendix A: Annexes to HMIPv6 Tests in UMU

6.1.1 INRIA's HMIPv6 IMPLEMENTATION TEST.

To test INRIA's HMIPv6 implementation we have made a basic test that consists of sending icmpv6 packets (ping6) several times per second. Our interest is focused on handover time in order to verify the information offered by the developers.

The test has been done by using schema shown by Figure 6-1. It is very simple testbed but it has been enough to obtain the results and study the rough behaviour of the implementation. Furthermore, as you can see we are only interested in micro-mobility and, therefore, we have centered on the three entities that need modifications in its function by adding micro-mobility behaviour : Gateway (need a modified router advertisement) Mobile Host and MAP.

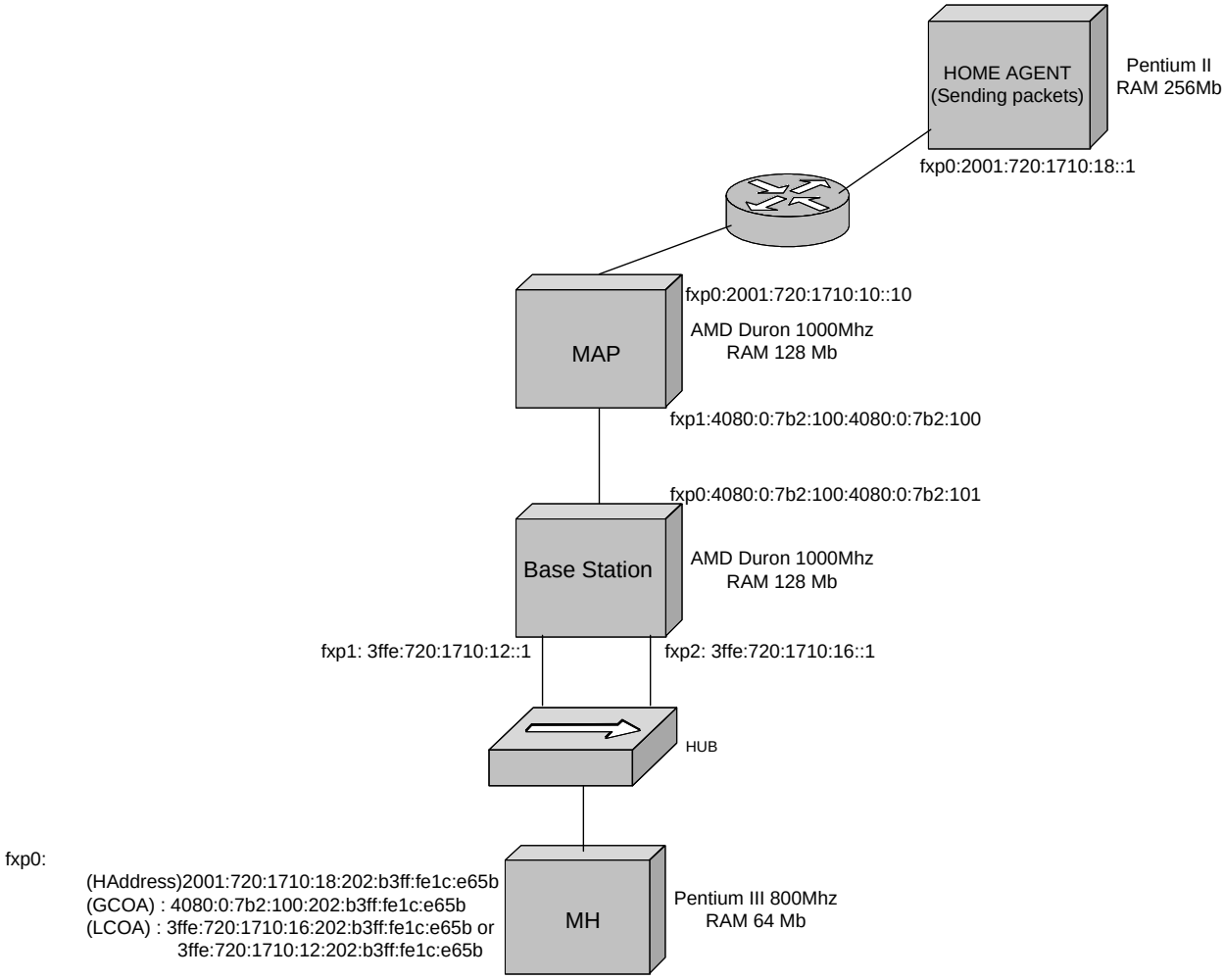


Figure 6-1: HMIPv6 basic testbed

In fact, Home Agent constantly sends ping6 packets to MH therefore it works as a correspondent node too.

MH continuously moves between networks advertised by Base Station (ten movements each test). The movement is simulated by the Base Station by disconnecting and connecting each interface alternatively. We have made simple C program to do this:

```
#include <stdio.h>
#include <stdlib.h>
#include <sys/time.h>
#include <unistd.h>

int main(int argc, char **argv)
{

 int nHandOver=15;
 u_int lengthHandOver=3000000; /*Micro-second*/
 int i,c;
 extern char *optarg;

 while ((c=getopt(argc,argv,"t:n:?")) != EOF)
 {
        switch (c)
        {
                  /*time between handover*/
                case 't':
                 lengthHandOver=(u_int)atol(optarg);
                 break;
                  /* number of handover */
                case 'n':
                 nHandOver=atoi(optarg);
                 break;
                default:
                fprintf(stdout,"Bad Input\n");

        }
  }
/*Alternately each interface is connected and disconnect.It supposes
interface fxp1 is initially down and fxp2 is up*/
 for (i=0; i < nHandOver;i++)
 {
        system("ifconfig fxp2 inet6 down");
        system("ifconfig fxp1 inet6 up");
        usleep(lengthHandOver);
        system("ifconfig fxp1 inet6 down");
        system("ifconfig fxp2 inet6 up");
        usleep(lengthHandOver);

 }
}
```

Furthermore, we have calculated the spent time when a switch between interfaces: shown in next table:

Test Number	1	2	3	4	5	6	7	8	Average
Down(usec)	2297	2272	2073	2314	2311	2302	2365	2335	2283,625
Up(usec)	2551	2619	2131	2557	2595	2555	2579	2536	2515,375
Average (usec) switching									**4799**

Figure 6-2: Average time between switch of interfaces

As we can see it approximately lasts 5 milliseconds and this time could be disregarded in the final results.

The obtained results can be seen in next tables:

Executed Command: `ping6 –i time_ping MH_Home_Address`

TEST 1	Time Ping(sec)=	**0,04**										Average
Nº handover		1	2	3	4	5	6	7	8	9	10	
NS final		87	187	287	388	488	588	688	788	889	989	
NS initial		45	146	246	346	446	546	647	747	847	947	
Lost Packets		42	41	41	42	42	42	41	41	42	42	**41,6**
Avg HandOver Time(sec)		**1,664**										
TEST 2	Time Ping(sec)=	**0,02**										Average
Nº handover		1	2	3	4	5	6	7	8	9	10	
NS final		462	596	729	863	1130	1264	1398	1531	1665	1798	
NS initial		393	527	661	794	1062	1195	1329	1463	1596	1730	
Lost Packets		69	69	68	69	68	69	69	68	69	68	**68,6**
Avg HandOver Time(sec)		**1,372**										
TEST 3	Time Ping(sec)=	**0,01**										Average
Nº handover		1	2	3	4	5	6	7	8	9	10	
NS final		1125	1325	1526	1726	1926	2127	2328	2528	2729	2929	
NS initial		1022	1223	1423	1624	1824	2025	2225	2426	2626	2827	
Lost Packets		103	102	103	102	102	102	103	102	103	102	**102,4**
Avg HandOver Time(sec)		**1,024**										
TEST 4	Time Ping(sec)=	**0,009**										Average
Nº handover		1	2	3	4	5	6	7	8	9	10	
NS final		1202	1389	1578	1766	1954	2142	2330	2518	2706	2894	

© Euro6IX Consortium

NS initial		1100	1288	1476	1664	1852	2040	2228	2416	2604	2792	
Lost Packets		102	101	102	102	102	102	102	102	102	102	**101,9**
Avg HandOver Time(sec)		**0,917**										
TEST 5	Time Ping(sec)=	**0,008**										Average
Nº handover		1	2	3	4	5	6	7	8	9	10	
NS final		841	1029	1217	1405	1593	1780	1968	2156	2345	2533	
NS initial		738	926	1114	1302	1490	1678	1866	2054	2242	2430	
Lost Packets		103	103	103	103	103	102	102	102	103	103	***102,7***
Avg HandOver Time(sec)		***0,822***										
TEST 6	Time Ping(sec)=	**0,005**										Average
Nº handover		1	2	3	4	5	6	7	8	9	10	
NS final		5493	5694	5894	6095	6295	6496	6696	6896	7097	7297	
NS initial		5390	5591	5791	5992	6192	6393	6593	6794	6994	7195	
Lost Packets		103	103	103	103	103	103	103	102	103	102	***102,8***
Avg HandOver Time(sec)		***0,514***										
TEST 7	Time Ping(sec)=	**0,001**										Average
Nº handover		1	2	3	4	5	6	7	8	9	10	
NS final		897	1085	1273	1461	1649	1837	2025	2213	2401	2589	
NS initial		794	982	1170	1358	1546	1734	1922	2110	2298	2486	
Lost Packets		103	103	103	103	103	103	103	103	103	103	***103***
Avg HandOver Time(sec)		***0,103***										

Figure 6-3: Results tests

NS: Number of sequence

Several comments about these tables:

To obtain the lost packets we only need to see what icmpv6 packages' number of sequence are lost.(NS final - NS initial)

By the way, we can see that the time between each ping is higher, the accuracy of measurement is worse and the time between handover is higher though this information is not really valid. Therefore, the time between pings must be put right. After several experiments, we can see under 0.009 seconds the lost packets are approximately the same (around 103 packets). The found explanation is machine sending packets is not able to sends pings on higher rate. Therefore, although we decrease the time between packets under 0.009s, they are sent to the same rate than pings sent each 0.009s.

Note each handover is done each 3.75 seconds. Under this value we have notice the Mobile Node is not able to detect the movement and it does not send any Binding Update to the MAP. Therefore, MAP has not updated information about Mobile Node's position and then the icmpv6 packets (ping6) cannot be delivered. In fact, Mobile Node remains in a weak state and it is not able to detect changes in the network anymore.

To conclude we can see the fittest value is 0.917 seconds (TEST 4) as average time between handovers. More or less, it coincides with the data obtained by the developers (aprox. 1 second).

6.1.2 CONFIGURATION FILES

Home Agent

/etc/rc.startIPv6

```
#!/bin/sh
sysctl -w net.inet6.ipv6.forwarding=1
autoconf6 -a
ifconfig fxp0 inet6 firstalias 2001:720:1710:0::/64 eui64
#ifconfig fxp1 inet6 firstalias 2001:720:1710:17::1/64
ifconfig fxp1 inet6 firstalias 2001:720:1710:18::1/64
route -n add -inet6 2000::/3 2001:720:1710:0:207:ebff:fe3a:1161
route -n add -inet6 4080:0:7b2:100::/64
2001:720:1710:0:207:ebff:fe3a:1161
```

/etc/gateway6

```
#Do not send router advertisement through fxp0 ethernet interface fxp0
option rtadv=n if=fxp0
#Send router advertisement with Home Agent flag signaling this machine
is a Home Agent.
#Avoid sends rip information through the network.
option haadv=y flag=H rtadv=40/60 ripin=n ripout=n if=fxp1
#Prefix information sent through fxp1 ethernet interface. R flag must
be activated.
prefix prefix=2001:720:1710:18::/64 flag=LAR if=fxp1
```

MAP

/etc/rc.startIPv6

```
#!/bin/sh
sysctl -w net.inet6.ipv6.forwarding=1
autoconf6 -a
ifconfig fxp0 inet6 firstalias 2001:720:1710:10::10/64
#ifconfig fxp1 inet6 firstalias 2001:720:1710:17::1/64
ifconfig fxp1 inet6 firstalias 4080:0:7b2:100:4080:0:7b2:100/64
route -n add -inet6 3ffe:720:1710:12::/64
4080:0:7b2:100:4080:0:7b2:101
route -n add -inet6 3ffe:720:1710:16::/64
4080:0:7b2:100:4080:0:7b2:101
route -n add -inet6 2000::/3 2001:720:1710:10::1
```

Base Station

/etc/rc.startIPv6

```
#!/bin/sh
sysctl -w net.inet6.ipv6.forwarding=1
autoconf6 -a
#ifconfig fxp0 inet6 firstalias 2001:720:1710:17::2/64
ifconfig fxp0 inet6 firstalias 4080:0:7b2:100:4080:0:7b2:101/64
ifconfig fxp1 inet6 firstalias 3ffe:720:1710:12::1/64
ifconfig fxp2 inet6 firstalias 3ffe:720:1710:16::1/64
#route -n add -inet6 2001:720:1710:10::/64
4080:0:7b2:100:4080:0:7b2:100
route -n add -inet6 2000::/3 4080:0:7b2:100:4080:0:7b2:100
```

6.2 Appendix B: Detailed MIPv6 tests in PTIN description

6.2.1 Introduction

The Internet is today an environment broadly used for many different purposes. In the last years, with the increasing number of available access technologies, with particular relevance for the wireless ones, and the use of new small devices such as laptops, PDAs or mobiles phones, it becomes each time most important that the Internet Protocol of the Next Generation can deal with this mobility in an appropriate way. The IPv6 protocol provides a more appropriated support of mobility than IPv4, bringing clear benefits in efficiency and simplicity.

The uses of IPv6 native mechanisms like auto-configuration, or Extension Headers like Routing Header or Destination Option Header, are facilities that improve the simplicity of how the protocol are implemented and, at the same time, the efficiency.

6.2.2 Technical Analysis of the MIPv6

6.2.2.1 Introduction

The Mobile IPv6 protocol (MIPv6), defined by the Mobile IP Working Group in the IETF [ietf-mip], describes the required operations necessaries that support the mobility of the devices. Each Mobile Node (MN) is always identified by its Home Address (Ha), regardless of its current point of attachment to the Internet.

While the MN is attached to its Home Network (HN), the communication is made in a standard IPv6 way. However, when the MN moves to a different network, it gets a new address, the Care-of Address (CoA), which provides the necessary information about the MN point of attachment. This way, the packets are routed transparently to the CoA, and then delivered to the MN. There, the MN reads the additional information contained on the Destination Option Header (DH), realizes that it is currently attached to a Foreign Network (FN), translate the CoA to the Ha address making a local loop, and then deliver the packet transparently to the upper layers.

The main entity that supports this operation, besides the MN, is the Home Agent (HA), which is located in the HN. This is the entity responsible for tracking the association between the CoA and the Ha for each MNs it is responsible for, in each moment. Furthermore, when any packet destined to the MN is sent to the HN, the HA catches the packet and delivers it to the MN, using IPv6 over IPv6. When the MN receives an encapsulated packet from its HA, it knows that the node that have sent this packet, called Correspondent Node (CN), is not aware of the actual point of attachment of the MN to the network. For this reason, the MN sends the necessary information about the actual association (CoA, Ha), in order to allow that the next time the CN sends a packet, do it directly using a Routing Header (RH). The information about the current association (CoA, Ha) owned by the HA and also by the CN (plus some other information, like Lifetimes, etc.), is saved in a structure called Binding Cache (BC). The MN requests operations, either to the HA or to the CN, to add the new association, is called a Binding.

When the CN is aware of the point of attachments of the MN, the communication is made directly, using the optimal path. The packets from the CN to the MN, are sent to the MN using a RH, while the packets from the MN to the CN are sent using a DH (*Home Address*), the later one to allow the CN to know what are the real IPv6 in use. Every time the MN moves its point of attachment to the network, going to another FNs, the process is repeated. When the MN returns backs to the original network (HN) the process is stopped and the normal IPv6 packets are exchanged between the nodes.

6.2.2.2 Implementations

Despite the MIPv6 protocol is not still a RFC, there are many implementations for different platforms, either for free or for commercial purposes. Because it is very difficult to work with all of them, we have selected the ones that have a strategically interest to PTIN: The free Linux platform, using the MIPL implementation of the University of Helsinki and, a commercial implementation, using the Cisco and Windows platforms.

In this semester we will just work with the Linux implementation, while the Cisco and Microsoft Windows will be used in the second semester. Thus, all descriptions of the tests and results contained in this document are just related to the MIPL implementation (MIPL version 0.9.1 for 2.4.16 kernel version).

This version implements the draft15 of the MIPv6 [ietf-mip], where the issues related to security are defined (even if still not completely). At the moment, the Cisco and Microsoft Windows platforms just implement the draft13, which makes more hard the test of all of them, and makes not possible to perform interoperability tests, because the draft MIPv6 versions have not backward compatibility.

The next sections will describe the general testbed used and the results achieved.

6.2.3 Testbed Design

The general testbed used in the tests that we will describe in the next sections is presented in Figure 3-1. We have 3 different networks with prefixes 3ffe:3103:0:8::/64, 3ffe:3103:0:9::/64 and 3ffe:3103:0:10::/64. Between the 3ffe:3103:0:8::/64 and 3ffe:3103:0:9::/64 networks, we have a Linux router acting as HA entity, and between the 3ffe:3103:0:9::/64 and 3ffe:3103:0:10::/64 networks, we have a Telebit router. Both, Linux and Telebit routers are announcing their respective networks, sending Router Advertisements (RAs). I.e., the Linux HA router is announcing to the 3ffe:3103:0:8::/64 network (*radvd* daemon), while the Telebit router is announcing in the 3ffe:3103:0:9::/64 and 3ffe:3103:0:10::/64 networks.

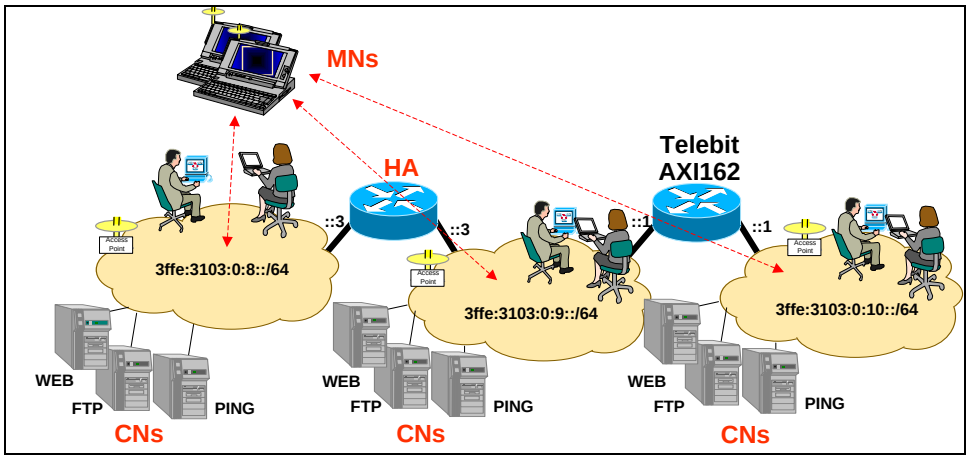


Figure 6-4: General Testbed used for MIPL tests.

In each network we have some simple applications like pings and traceroutes, and other services like www, or ftp (servers and clients). Either the MNs or the CN we use, have support of these applications/services. The MNs used, have their HNs on the 3ffe:3103:0:8::/64 network.

This general scenario allows us to test different technologies such as LANs (802.3) and WLANs (802.11). The attachment to all networks can be made either via wired LANs or wireless WLAN. In the latter case the

Thus, we have split the tests in two sets of scenarios: in the first ones, the scenarios when we use LANs to attach the MN and, in the second ones, when we use WLAN.

6.2.4 Implementing the tests

In this section we describe the scenarios we have tested using the previously described testbed. Not all scenarios are represented, but just the ones that really represent different and interesting situations.

The configuration used for each entity was always the same and can be viewed in the Annexes 6.3.1. The Annex 6.3.1.1 shows the HA configuration, the Annex 6.3.1.2 shows the configuration for the different CNs used and, finally, the Annex 6.3.1.3 shows the configuration used for the MNs.

To analyze the flow of the protocol information exchange, we have used the Microsoft Network Monitor (*netmon*), which allow us to recognize some of the new information related to mobility like Destination Option Header codes (e.g. Home Agent or Binding Updates), among others.

The tests can be divided in two categories. In the former ones, a wired technology is used to attach the MN to the links, while in the latter ones, the wireless technology has been used to attach the MN. For these both scenarios, similar tests have been performance, in order to allow assess the differences between of them.

6.2.4.1 LANs testing

In these tests, we use the wired LAN technology to attach all the nodes to the network. The hand-off is made changing the cable from the hub of one network to another.

It is important to notice that the external behaviour we present in the following tests, must be analyzed taking into consideration that some time was spent on these operations and, actually, should not be considered as a handover time.

6.2.4.1.1 Scenario 1

In the first scenario, the MN is located in its HN and the CN is located in another network (Correspondent Node Network - CNN). The Figure 6-5 shows graphically this scenario.

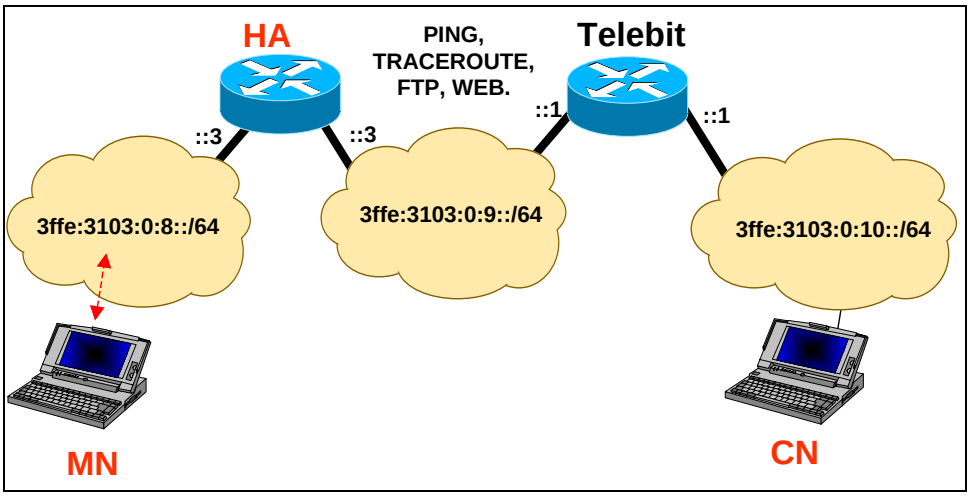


Figure 6-5: LAN: Scenario 1: MN in the HN. CN is located in other network.

Characteristics of equipment involved:

Equipment Characteristics	HA	MN	CN	FN Router
Platform	*i386*	*i386*	*i386*	*Telebit*
Model	*"Blank" PC*	*"Blank" PC*	*"Blank" PC*	*AXI 462/3*
CPU	*Pentium II 400 MHz (cache 512 KB)*	*Pentium III 650 MHz (cache 256 KB)*	*Pentium II 266 MHz (cache 512 KB)*	*RISC (TBC 103)*
Memory	*64 RAM (swap 256 MB)*	*128 RAM (swap 256 MB)*	*64 RAM (swap 256 MB)*	*16 MB*
Network Interfaces Card	*2 Ethernet cards 3COM series 59x (driver 3c59x)*	*1 Ethernet card 3COM series 59x (driver 3c59x)*	*1 Ethernet card 3COM series 59x (driver 3c59x)*	*2 Ethernet cards (TCB 120) Motorola Power PC 750*
Operating System / Kernel	*Linux 7.2 (kernel 2.4.16)*	*Linux 7.2 (kernel 2.4.16)*	*Linux 7.2 (kernel 2.4.16)*	*Telebit OS Swtbcpaxos 2.6a14*

MN location:

Home Network (HN) - 3ffe:3103:0:8::/64 (via LAN).

CN location:

Correspondent Node Network (CNN) - 3ffe:3103:0:10::/64 (via LAN).

Applications:

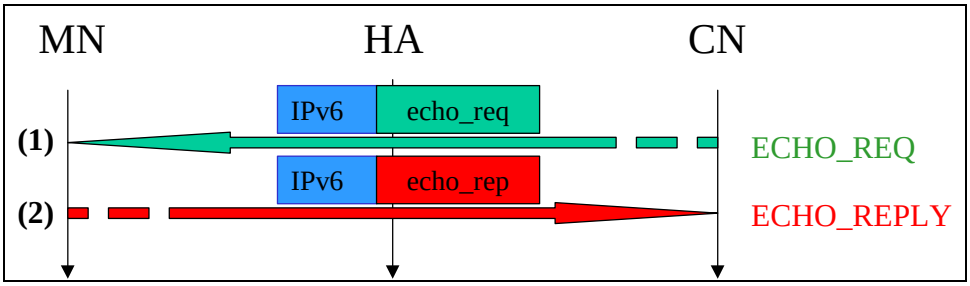
PINGs, FTPs and WWW.

External Behaviour:

In this scenario all packets are processed as an IPv6 standard way. For this reason, all the applications/services work as usually.

Schematic DataFlow:

* Ping from the CN to MN.



Packet Analysis:

A normal IPv6 Echo Request. Equal as if the mobility has not been enabled.

A normal IPv6 Echo Reply.

Sniffed packets:

* The particular packets here presented are from the PING application.

- Echo Request

```
ETHERNET: ETYPE = 0x86DD : Protocol = Unknown
IP6: Proto = ICMP6; Len = 40
    IP6: Version = 6 (0x6)
    IP6: Traffic Class = 0 (0x0)
    IP6: Flow Label = 0 (0x0)
    IP6: Payload Length = 40 (0x28)
    IP6: Next Header = 58 (ICMP6)
    IP6: Hop Limit = 126 (0x7E)
    IP6: Source Address = 3ffe:3103:0:10:260:8ff:fe46:46e7
    IP6: Destination Address = 3ffe:3103:0:8:201:2ff:fe2a:93b9
    IP6: Payload: Number of data bytes remaining = 40 (0x0028)
ICMP6: Echo Request; ID = 23, Seq = 60579
    ICMP6: Checksum = 0x8208
    ICMP6: Type = 128 (Echo Request)
    ICMP6: Code = 0 (0x0)
    ICMP6: Identifier = 23 (0x17)
    ICMP6: Sequence Number = 60579 (0xECA3)
    ICMP6: Data: Number of data bytes remaining = 32 (0x0020)
```

(2) - Echo Reply

```
ETHERNET: ETYPE = 0x86DD : Protocol = Unknown
IP6: Proto = ICMP6; Len = 40
    IP6: Version = 6 (0x6)
    IP6: Traffic Class = 0 (0x0)
    IP6: Flow Label = 0 (0x0)
    IP6: Payload Length = 40 (0x28)
    IP6: Next Header = 58 (ICMP6)
    IP6: Hop Limit = 64 (0x40)
    IP6: Source Address = 3ffe:3103:0:8:201:2ff:fe2a:93b9
    IP6: Destination Address = 3ffe:3103:0:10:260:8ff:fe46:46e7
    IP6: Payload: Number of data bytes remaining = 40 (0x0028)
ICMP6: Echo Reply; ID = 23, Seq = 60579
    ICMP6: Checksum = 0x8108
    ICMP6: Type = 129 (Echo Reply)
    ICMP6: Code = 0 (0x0)
    ICMP6: Identifier = 23 (0x17)
    ICMP6: Sequence Number = 60579 (0xECA3)
    ICMP6: Data: Number of data bytes remaining = 32 (0x0020)
```

6.2.4.1.2 Scenario 2

In the second scenario, the MN starts working in a visited network (FN). The CN is located in another network. The Figure 6-6 shows graphically this scenario.

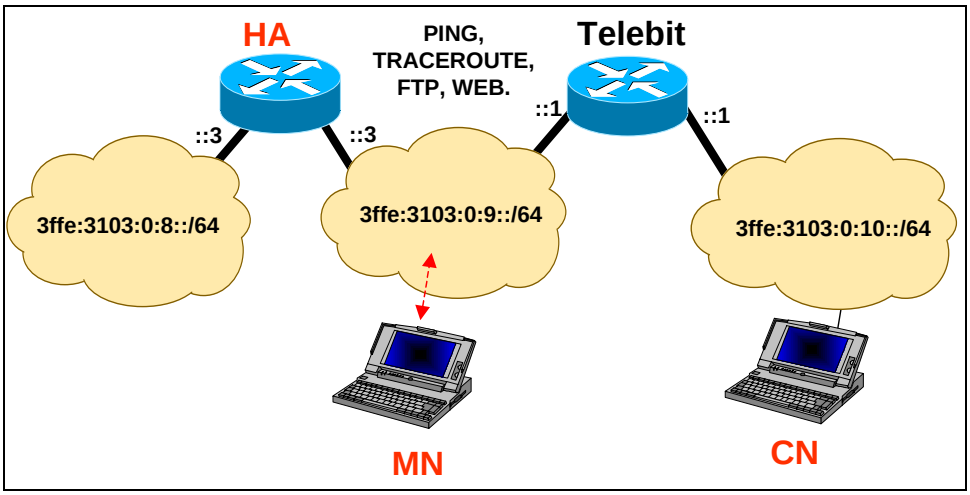


Figure 6-6: LAN: Scenario 2: MN starting in a FN.

Characteristics of equipment involved:

Equipment Characteristics	HA	MN	CN	FN Router
Platform	*i386*	*i386*	*i386*	*Telebit*
Model	*"Blank" PC*	*"Blank" PC*	*"Blank" PC*	*AXI 462/3*
CPU	*Pentium II 400 MHz (cache 512 KB)*	*Pentium III 650 MHz (cache 256 KB)*	*Pentium II 266 MHz (cache 512 KB)*	*RISC (TBC 103)*
Memory	*64 RAM (swap 256 MB)*	*128 RAM (swap 256 MB)*	*64 RAM (swap 256 MB)*	*16 MB*
Network Interfaces Card	*2 Ethernet cards 3COM series 59x (driver 3c59x)*	*1 Ethernet card 3COM series 59x (driver 3c59x)*	*1 Ethernet card 3COM series 59x (driver 3c59x)*	*2 Ethernet cards (TCB 120) Motorola Power PC 750*
Operating System / Kernel	*Linux 7.2 (kernel 2.4.16)*	*Linux 7.2 (kernel 2.4.16)*	*Linux 7.2 (kernel 2.4.16)*	*Telebit OS swtbcpaxos 2.6a14*

MN location:

Foreign Network (FN) - 3ffe:3103:0:9::/64 (via LAN).

CN location:

Correspondent Node Network (CNN) - 3ffe:3103:0:10::/64 (via LAN).

Applications:

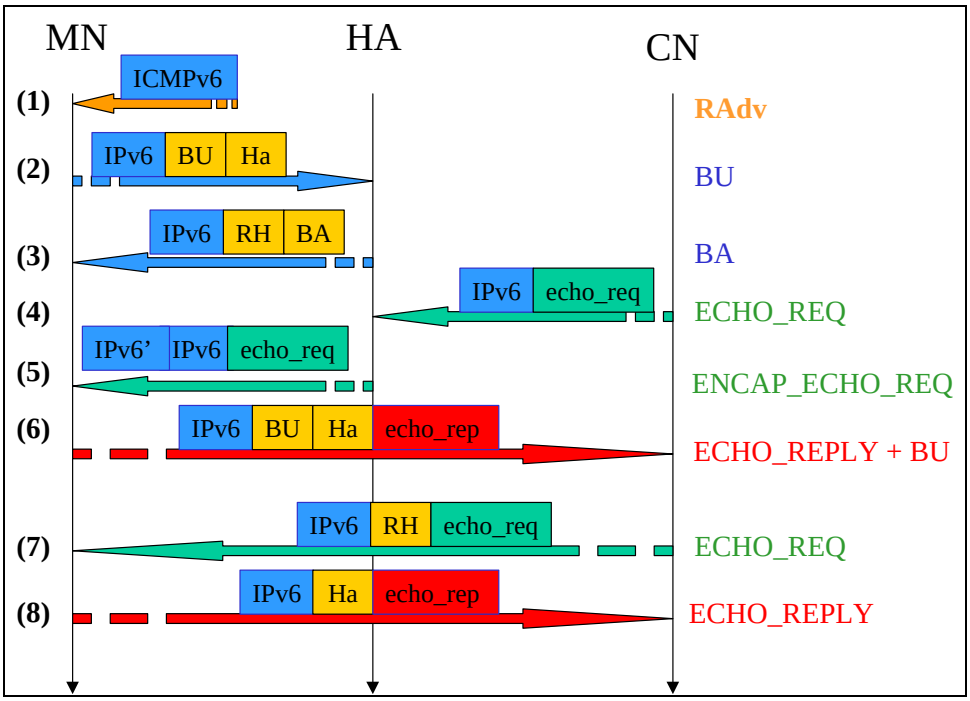
PINGs, FTPs and WWW.

External Behaviour:

The MN works with all applications/services transparently as if it was in its HN. After the movement to attach the nodes as specified in the Figure, when the MN is down about 2 - 5 seconds, it works properly.
No any relevant difference is noticed in comparison to the previous scenario, except that sometimes is noticeable an increase of the round trip times in the ping.

Schematic DataFlow:

* Ping from the CN to MN.



Packet Analysis:

The Router Advertisements (RAs) are being sent periodically by the Telebit. The prefix announced is the 3ffe:3103:0:9::/64.

A Binding Update (BU) is sent from the MN to the HA, in order to register the CoA acquired, using a Ha Destination Option (DH).

A Binding Acknowledge (BA) from the HA confirms the BU. The HA use a Routing Header (RH) to forward the packet.

The first time that the MN sends an *Echo Request* to the MN, it is sent to the MN Home link.

The packet is there intercepted by the HA and, is encapsulated, using IPv6 over IPv6, and sent to the MN.

When the MN receives a packet encapsulated by his HA, it knows that there is a CN that doesn't know it's actual location. For this reason a BU is sent to the CN to actualize the CN Binding Cache (BC) - for this is used again the Ha DH.

The steps are completed and, the next packets from the CN to the MN are sent directly using a Routing Header (RH).

The packets from the MN to the CN are sent directly using a Home address (Ha) DH.

Sniffed packets:

* The particular packets here presented are from the PING application.

(1) - Router Advertisement

```
ETHERNET: ETYPE = 0x86DD : Protocol = Unknown
IP6: Proto = ICMP6; Len = 56
    IP6: Version = 6 (0x6)
    IP6: Traffic Class = 0 (0x0)
    IP6: Flow Label = 0 (0x0)
    IP6: Payload Length = 56 (0x38)
    IP6: Next Header = 58 (ICMP6)
    IP6: Hop Limit = 255 (0xFF)
    IP6: Source Address = fe80::210:5aff:fe67:55bc
    IP6: Destination Address = ff02::1
    IP6: Payload: Number of data bytes remaining = 56 (0x0038)
ICMP6: Router Advertisement
    ICMP6: Checksum = 0x5AA8
    ICMP6: Type = 134 (Router Advertisement)
    ICMP6: Code = 0 (0x0)
```

```
    ICMP6: Current Hop Limit = 64 (0x40)
    ICMP6: 0....... = Not managed address config
    ICMP6: .0...... = Not other stateful config
    ICMP6: Router Lifetime = 600 (0x258)
    ICMP6: Reachable Time = 0 (0x0)
    ICMP6: Retransmission Timer = 0 (0x0)
    ICMP6: Prefix = 3ffe:3103:0:9::
        ICMP6: Type = 3 (0x3)
        ICMP6: Length = 4 (0x4)
        ICMP6: Prefix Length = 64 (0x40)
        ICMP6: Site Prefix Length = 0 (0x0)
        ICMP6: 1....... = On-link determination allowed
        ICMP6: .1...... = Autonomous address configuration
        ICMP6: ..0..... = No router address
        ICMP6: ...0.... = Not a site prefix
        ICMP6: .......0 = Not a route prefix
        ICMP6: Valid Lifetime = 2592000 (0x278D00)
        ICMP6: Preferred Lifetime = 604800 (0x93A80)
        ICMP6: Reserved
        ICMP6: Prefix = 3ffe:3103:0:9::
    ICMP6: Source Link-level Address = 00 10 5A 67 55 BC
        ICMP6: Type = 1 (0x1)
        ICMP6: Length = 1 (0x1)
        ICMP6: Source Link-level Address = 00 10 5A 67 55 BC
```

(2) - Binding Update

```
ETHERNET: ETYPE = 0x86DD : Protocol = Unknown
IP6: Dest Opts; Auth SPI = 0x40a; Dest Opts; Len = 0
    IP6: Version = 6 (0x6)
    IP6: Traffic Class = 0 (0x0)
    IP6: Flow Label = 0 (0x0)
    IP6: Payload Length = 64 (0x40)
    IP6: Next Header = 60 (Destination Options Header)
    IP6: Hop Limit = 255 (0xFF)
    IP6: Source Address = 3ffe:3103:0:9:201:2ff:fe2a:93b9
    IP6: Destination Address = 3ffe:3103:0:8::3
    IP6: Destination Options Header
        IP6: Next Header = 51 (Authentication Header)
        IP6: Length = 2 (0x2)
        IP6: Padding (4 bytes)
            IP6: Type = 1 (PadN)
                IP6: 00...... = Skip option if not recognized
                IP6: ..0..... = Option data does not change enroute
            IP6: Length = 2 (0x2)
            IP6: Data = 00 00
        IP6: Home Address Option
            IP6: Type = 201
                IP6: 11...... = Discard packet if not recognized, and send ICMP if not multicast
                IP6: ..0..... = Option data does not change enroute
            IP6: Length = 16 (0x10)
            IP6: Home Address = 3ffe:3103:0:8:201:2ff:fe2a:93b9
    IP6: Authentication Header
        IP6: Next Header = 60 (Destination Options Header)
        IP6: Length = 4 (0x4)
        IP6: Reserved
        IP6: SPI = 1034 (0x40A)
        IP6: Seq Num = 4 (0x4)
        IP6: Auth Data = 16 12 37 89 41 0B 1C 88 00 8A 81 DC
    IP6: Destination Options Header
        IP6: Next Header = 59 (None)
        IP6: Length = 1 (0x1)
        IP6: Binding Update Option
            IP6: Type = 198
                IP6: 11...... = Discard packet if not recognized, and send ICMP if not multicast
                IP6: ..0..... = Option data does not change enroute
            IP6: Length = 8 (0x8)
            IP6: Flags
                IP6: 1....... = Request a binding acknowledgement response.
                IP6: .1...... = Request host to act as home agent.
                IP6: ..1..... = Binding update includes a care-of address.
            IP6: Prefix Length = 0 (0x0)
            IP6: Sequence Number = 6 (0x6)
            IP6: Lifetime = 10000 (0x2710)
*           IP6: Care-of Address = 102:0:8100:a13b:fc24:1500:5e67:133d
```

* The last line (Care-of Address) is a wrong value because the sniffer just understands draft13.

(3) - Binding Acknowledge

```
ETHERNET: ETYPE = 0x86DD : Protocol = Unknown
IP6: Routing (1 left of 1); Auth SPI = 0x409; Dest Opts; Len = 0
    IP6: Version = 6 (0x6)
    IP6: Traffic Class = 0 (0x0)
    IP6: Flow Label = 0 (0x0)
    IP6: Payload Length = 64 (0x40)
    IP6: Next Header = 43 (Routing Header)
    IP6: Hop Limit = 255 (0xFF)
    IP6: Source Address = 3ffe:3103:0:8::3
    IP6: Destination Address = 3ffe:3103:0:9:201:2ff:fe2a:93b9
    IP6: Routing Header
        IP6: Next Header = 51 (Authentication Header)
        IP6: Length = 2 (0x2)
        IP6: Type = 0 (0x0)
        IP6: Segments Left = 1 (0x1)
        IP6: Reserved
        IP6: Route
            IP6: Address = 3ffe:3103:0:8:201:2ff:fe2a:93b9
    IP6: Authentication Header
        IP6: Next Header = 60 (Destination Options Header)
```

```
        IP6: Length = 4 (0x4)
        IP6: Reserved
        IP6: SPI = 1033 (0x409)
        IP6: Seq Num = 4 (0x4)
        IP6: Auth Data = A3 C5 76 C8 39 0C 5B 55 67 4D 94 D2
    IP6: Destination Options Header
        IP6: Next Header = 59 (None)
        IP6: Length = 1 (0x1)
        IP6: Padding (1 bytes)
            IP6: Type = 0 (Pad1)
                IP6: 00...... = Skip option if not recognized
                IP6: ..0..... = Option data does not change enroute
        IP6: Binding Acknowledgement Option
            IP6: Type = 7
                IP6: 00...... = Skip option if not recognized
                IP6: ..0..... = Option data does not change enroute
            IP6: Length = 11 (0xB)
            IP6: Status: 0 = Binding accepted.
            IP6: Sequence Number = 6 (0x6)
            IP6: Lifetime = 1000 (0x3E8)
            IP6: Refresh = 800 (0x320)
```

(5) - Echo Request - HA encapsulation

```
ETHERNET: ETYPE = 0x86DD : Protocol = Unknown
IP6: Proto = 0x29; Len = 104
    IP6: Version = 6 (0x6)
    IP6: Traffic Class = 0 (0x0)
    IP6: Flow Label = 0 (0x0)
    IP6: Payload Length = 104 (0x68)
    IP6: Next Header = 41
    IP6: Hop Limit = 255 (0xFF)
    IP6: Source Address = 3ffe:3103:0:8::3
    IP6: Destination Address = 3ffe:3103:0:9:201:2ff:fe2a:93b9
    IP6: Payload: Number of data bytes remaining = 104 (0x0068)
IP6: Proto = ICMP6; Len = 64
    IP6: Version = 6 (0x6)
    IP6: Traffic Class = 0 (0x0)
    IP6: Flow Label = 0 (0x0)
    IP6: Payload Length = 64 (0x40)
    IP6: Next Header = 58 (ICMP6)
    IP6: Hop Limit = 62 (0x3E)
    IP6: Source Address = 3ffe:3103:0:10:260:8ff:fe46:46e7
    IP6: Destination Address = 3ffe:3103:0:8:201:2ff:fe2a:93b9
    IP6: Payload: Number of data bytes remaining = 64 (0x0040)
ICMP6: Echo Request; ID = 64548, Seq = 5376
    ICMP6: Checksum = 0xA23B
    ICMP6: Type = 128 (Echo Request)
    ICMP6: Code = 0 (0x0)
    ICMP6: Identifier = 64548 (0xFC24)
    ICMP6: Sequence Number = 5376 (0x1500)
    ICMP6: Data: Number of data bytes remaining = 56 (0x0038)
```

(6) - Echo Reply + CN Binding Update

```
ETHERNET: ETYPE = 0x86DD : Protocol = Unknown
IP6: Dest Opts; Auth SPI = 0x409; Dest Opts; Proto = ICMP6; Len = 64
    IP6: Version = 6 (0x6)
    IP6: Traffic Class = 0 (0x0)
    IP6: Flow Label = 0 (0x0)
    IP6: Payload Length = 128 (0x80)
    IP6: Next Header = 60 (Destination Options Header)
    IP6: Hop Limit = 64 (0x40)
    IP6: Source Address = 3ffe:3103:0:9:201:2ff:fe2a:93b9
    IP6: Destination Address = 3ffe:3103:0:10:260:8ff:fe46:46e7
    IP6: Destination Options Header
        IP6: Next Header = 51 (Authentication Header)
        IP6: Length = 2 (0x2)
        IP6: Padding (4 bytes)
            IP6: Type = 1 (PadN)
                IP6: 00...... = Skip option if not recognized
                IP6: ..0..... = Option data does not change enroute
            IP6: Length = 2 (0x2)
            IP6: Data = 00 00
        IP6: Home Address Option
            IP6: Type = 201
                IP6: 11...... = Discard packet if not recognized, and send ICMP if not multicast
                IP6: ..0..... = Option data does not change enroute
            IP6: Length = 16 (0x10)
            IP6: Home Address = 3ffe:3103:0:8:201:2ff:fe2a:93b9
    IP6: Authentication Header
        IP6: Next Header = 60 (Destination Options Header)
        IP6: Length = 4 (0x4)
        IP6: Reserved
        IP6: SPI = 1033 (0x409)
        IP6: Seq Num = 1 (0x1)
        IP6: Auth Data = A6 EC A2 38 D7 C7 4A 62 35 65 ED E7
    IP6: Destination Options Header
        IP6: Next Header = 58 (ICMP6)
        IP6: Length = 1 (0x1)
        IP6: Binding Update Option
            IP6: Type = 198
                IP6: 11...... = Discard packet if not recognized, and send ICMP if not multicast
                IP6: ..0..... = Option data does not change enroute
            IP6: Length = 8 (0x8)
            IP6: Flags
                IP6: ..1..... = Binding update includes a care-of address.
            IP6: Prefix Length = 0 (0x0)
            IP6: Sequence Number = 9 (0x9)
            IP6: Lifetime = 1000 (0x3E8)
```

```
*           IP6: Care-of Address = 102:0:8100:a13b:fc24:1500:5e67:133d
ICMP6: Echo Reply; ID = 64548, Seq = 5376
    ICMP6: Checksum = 0xA13B (ERROR: should be 0xA13A)
    ICMP6: Type = 129 (Echo Reply)
    ICMP6: Code = 0 (0x0)
    ICMP6: Identifier = 64548 (0xFC24)
    ICMP6: Sequence Number = 5376 (0x1500)
    ICMP6: Data: Number of data bytes remaining = 56 (0x0038)
```

* The line Care-of Address is a wrong value because the sniffer just understands draft13.

(7) - Echo Request - directly

```
ETHERNET: ETYPE = 0x86DD : Protocol = Unknown
IP6: Routing (1 left of 1); Proto = ICMP6; Len = 64
    IP6: Version = 6 (0x6)
    IP6: Traffic Class = 0 (0x0)
    IP6: Flow Label = 0 (0x0)
    IP6: Payload Length = 88 (0x58)
    IP6: Next Header = 43 (Routing Header)
    IP6: Hop Limit = 63 (0x3F)
    IP6: Source Address = 3ffe:3103:0:10:260:8ff:fe46:46e7
    IP6: Destination Address = 3ffe:3103:0:9:201:2ff:fe2a:93b9
    IP6: Routing Header
        IP6: Next Header = 58 (ICMP6)
        IP6: Length = 2 (0x2)
        IP6: Type = 0 (0x0)
        IP6: Segments Left = 1 (0x1)
        IP6: Reserved
        IP6: Route
            IP6: Address = 3ffe:3103:0:8:201:2ff:fe2a:93b9
    IP6: Payload: Number of data bytes remaining = 64 (0x0040)
ICMP6: Echo Request; ID = 64548, Seq = 6144
    ICMP6: Checksum = 0x983B (ERROR: should be 0x983A)
    ICMP6: Type = 128 (Echo Request)
    ICMP6: Code = 0 (0x0)
    ICMP6: Identifier = 64548 (0xFC24)
    ICMP6: Sequence Number = 6144 (0x1800)
    ICMP6: Data: Number of data bytes remaining = 56 (0x0038)
```

(8) - Echo Reply - directly

```
ETHERNET: ETYPE = 0x86DD : Protocol = Unknown
IP6: Dest Opts; Proto = ICMP6; Len = 64
    IP6: Version = 6 (0x6)
    IP6: Traffic Class = 0 (0x0)
    IP6: Flow Label = 0 (0x0)
    IP6: Payload Length = 88 (0x58)
    IP6: Next Header = 60 (Destination Options Header)
    IP6: Hop Limit = 64 (0x40)
    IP6: Source Address = 3ffe:3103:0:9:201:2ff:fe2a:93b9
    IP6: Destination Address = 3ffe:3103:0:10:260:8ff:fe46:46e7
    IP6: Destination Options Header
        IP6: Next Header = 58 (ICMP6)
        IP6: Length = 2 (0x2)
        IP6: Padding (4 bytes)
            IP6: Type = 1 (PadN)
                IP6: 00...... = Skip option if not recognized
                IP6: ..0..... = Option data does not change enroute
            IP6: Length = 2 (0x2)
            IP6: Data = 00 00
        IP6: Home Address Option
            IP6: Type = 201
                IP6: 11...... = Discard packet if not recognized, and send ICMP if not multicast
                IP6: ..0..... = Option data does not change enroute
            IP6: Length = 16 (0x10)
            IP6: Home Address = 3ffe:3103:0:8:201:2ff:fe2a:93b9
    IP6: Payload: Number of data bytes remaining = 64 (0x0040)
ICMP6: Echo Reply; ID = 64548, Seq = 6144
    ICMP6: Checksum = 0x973B (ERROR: should be 0x973A)
    ICMP6: Type = 129 (Echo Reply)
    ICMP6: Code = 0 (0x0)
    ICMP6: Identifier = 64548 (0xFC24)
    ICMP6: Sequence Number = 6144 (0x1800)
    ICMP6: Data: Number of data bytes remaining = 56 (0x0038)
```

6.2.4.1.3 Scenario 3

In the scenario number 3, when the MN starts is located in its HN (1). Then, it moves to a FN (2) and, finally, it backs again to the HN (3). The Figure 6-7 shows graphically this scenario.

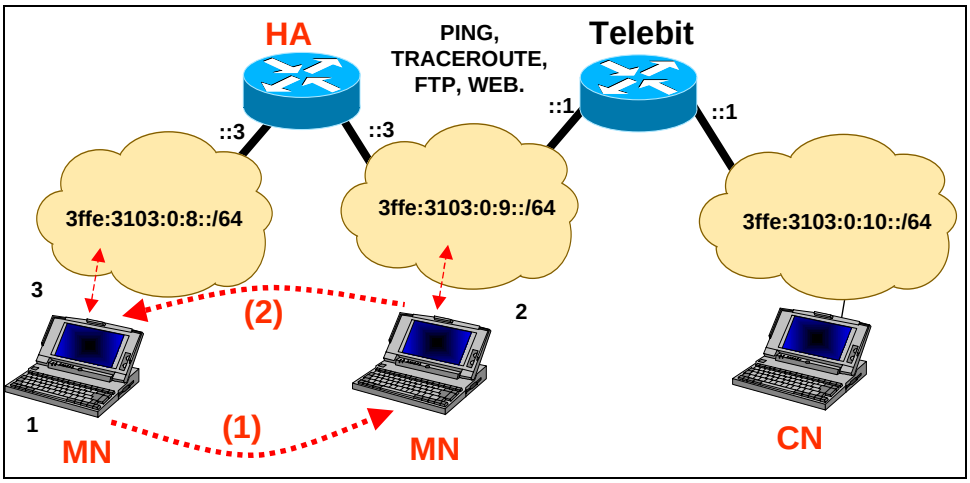


Figure 6-7: LAN: Scenario 3: The MN starts in its HN, goes to a FN and backs to the HN.

Characteristics of equipment involved:

Equipment Characteristics	HA	MN	CN	FN Router
Platform	*i386*	*i386*	*i386*	*Telebit*
Model	*"Blank" PC*	*"Blank" PC*	*"Blank" PC*	*AXI 462/3*
CPU	*Pentium II 400 MHz (cache 512 KB)*	*Pentium III 650 MHz (cache 256 KB)*	*Pentium II 266 MHz (cache 512 KB)*	*RISC (TBC 103)*
Memory	*64 RAM (swap 256 MB)*	*128 RAM (swap 256 MB)*	*64 RAM (swap 256 MB)*	*16 MB*
Network Interfaces Card	*2 Ethernet cards 3COM series 59x (driver 3c59x)*	*1 Ethernet card 3COM series 59x (driver 3c59x)*	*1 Ethernet card 3COM series 59x (driver 3c59x)*	*2 Ethernet cards (TCB 120) Motorola Power PC 750*
Operating System / Kernel	*Linux 7.2 (kernel 2.4.16)*	*Linux 7.2 (kernel 2.4.16)*	*Linux 7.2 (kernel 2.4.16)*	*Telebit OS swtbcpaxos 2.6a14*

MN location:

HN (3ffe:3103:0:8::/64) -> FN (3ffe:3103:0:9::/64) -> HN (3ffe:3103:0:8::/64) (via LAN).

CN location:

Correspondent Node Network (CNN) - 3ffe:3103:0:10::/64 (via LAN).

Applications:

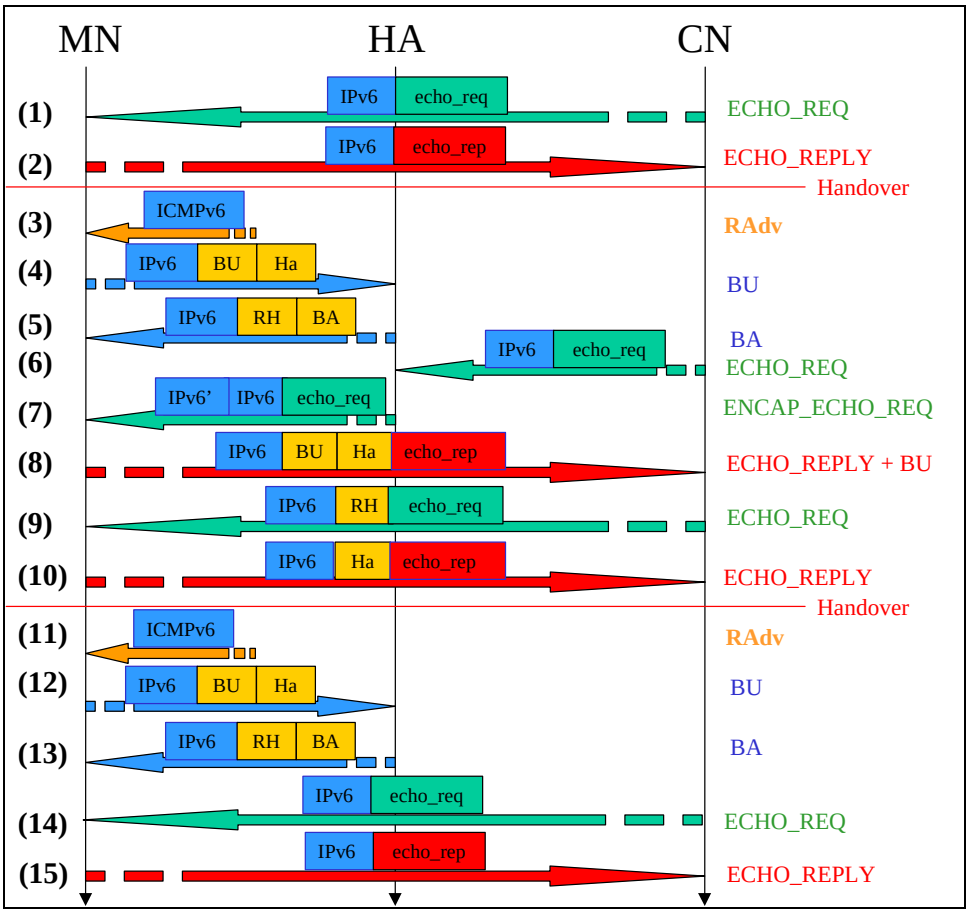
PINGs, FTPs and WWW.

External Behaviour:

When the MN starts in the HN it works as usually.
After, when it moves to the FN, the network goes down during a time random interval of about 2 - 5 seconds.
For the ping application, is clear that there is no answer between this interval of time. For the others application the download of the file/page stops during this time.
When the MN moves back to the HN, the network goes down again during a random period of about 2 - 5 seconds, starting the application without problems.
There are some situations, mainly using TCP (www and ftp), when the connection never restarts and its necessary restart the downloading/transference. In other cases, we must restart the mobility operation to make it works correctly again.
* Is important to notice that the time of 2 - 5 seconds, depends of how many time we spend changing the cable between the hubs. This time should not be considered as handover time.

Schematic DataFlow:

* Ping from the CN to MN.



Packet Analysis:

The packets "Echo Requests" are exchanged directly.

The packets "Echo Replies" are exchanged directly.

The Router Advertisements (RAs) are being sent periodically by the Telebit. The prefix announced is the 3ffe:3103:0:9::/64.

A Binding Update (BU) is sent from the MN to the HA, in order to register the CoA acquired, using a Ha Destination Option (DH).

A Binding Acknowledge (BA) from the HA confirms the BU. The HA use a Routing Header (RH) to forward the packet.

The first time that the MN sends an *Echo Request* to the MN, it is sent to the MN Home link.

The packet is there intercepted by the HA and, is encapsulated, using IPv6 over IPv6, and sent to the MN.

When the MN receives a packet encapsulated by his HA, it knows that there is a CN that doesn't know it's actual location. For this reason a BU is sent to the CN to actualize the CN Binding Cache (BC) - for this is used again the Ha DH.

The steps are completed and, the next packets from the CN to the MN are sent directly using a Routing Header (RH).

The packets from the MN to the CN are sent directly using a Home address (Ha) DH.

When the MN backs to its HN, it receives the RAs.

Then, it knows that is in its HN and sends a BU to the HA with a CoA equal to its Ha.

The HA confirms the MN and remove its entry in the BC.

And the communication starts to be directly between the CN and the MN, using ordinary IPv6 packets.

As well as for the MN - CN direction.

Sniffed packets:

* The particular packets here presented are from the PING application.

- Echo Request

```
ETHERNET: ETYPE = 0x86DD : Protocol = Unknown
IP6: Proto = ICMP6; Len = 40
    IP6: Version = 6 (0x6)
    IP6: Traffic Class = 0 (0x0)
    IP6: Flow Label = 0 (0x0)
    IP6: Payload Length = 40 (0x28)
    IP6: Next Header = 58 (ICMP6)
    IP6: Hop Limit = 126 (0x7E)
    IP6: Source Address = 3ffe:3103:0:10:260:8ff:fe46:46e7
    IP6: Destination Address = 3ffe:3103:0:8:201:2ff:fe2a:93b9
    IP6: Payload: Number of data bytes remaining = 40 (0x0028)
ICMP6: Echo Request; ID = 23, Seq = 60579
    ICMP6: Checksum = 0x8208
    ICMP6: Type = 128 (Echo Request)
    ICMP6: Code = 0 (0x0)
    ICMP6: Identifier = 23 (0x17)
    ICMP6: Sequence Number = 60579 (0xECA3)
    ICMP6: Data: Number of data bytes remaining = 32 (0x0020)
```

(2) - Echo Reply

```
ETHERNET: ETYPE = 0x86DD : Protocol = Unknown
IP6: Proto = ICMP6; Len = 40
    IP6: Version = 6 (0x6)
    IP6: Traffic Class = 0 (0x0)
    IP6: Flow Label = 0 (0x0)
    IP6: Payload Length = 40 (0x28)
    IP6: Next Header = 58 (ICMP6)
    IP6: Hop Limit = 64 (0x40)
    IP6: Source Address = 3ffe:3103:0:8:201:2ff:fe2a:93b9
    IP6: Destination Address = 3ffe:3103:0:10:260:8ff:fe46:46e7
    IP6: Payload: Number of data bytes remaining = 40 (0x0028)
ICMP6: Echo Reply; ID = 23, Seq = 60579
    ICMP6: Checksum = 0x8108
    ICMP6: Type = 129 (Echo Reply)
    ICMP6: Code = 0 (0x0)
    ICMP6: Identifier = 23 (0x17)
    ICMP6: Sequence Number = 60579 (0xECA3)
    ICMP6: Data: Number of data bytes remaining = 32 (0x0020)
```

(3) - Router Advertisement

```
ETHERNET: ETYPE = 0x86DD : Protocol = Unknown
IP6: Proto = ICMP6; Len = 56
    IP6: Version = 6 (0x6)
    IP6: Traffic Class = 0 (0x0)
    IP6: Flow Label = 0 (0x0)
    IP6: Payload Length = 56 (0x38)
    IP6: Next Header = 58 (ICMP6)
    IP6: Hop Limit = 255 (0xFF)
    IP6: Source Address = fe80::210:5aff:fe67:55bc
    IP6: Destination Address = ff02::1
    IP6: Payload: Number of data bytes remaining = 56 (0x0038)
ICMP6: Router Advertisement
    ICMP6: Checksum = 0x5AA8
    ICMP6: Type = 134 (Router Advertisement)
    ICMP6: Code = 0 (0x0)
    ICMP6: Current Hop Limit = 64 (0x40)
    ICMP6: 0....... = Not managed address config
    ICMP6: .0...... = Not other stateful config
    ICMP6: Router Lifetime = 600 (0x258)
    ICMP6: Reachable Time = 0 (0x0)
    ICMP6: Retransmission Timer = 0 (0x0)
    ICMP6: Prefix = 3ffe:3103:0:9::
        ICMP6: Type = 3 (0x3)
        ICMP6: Length = 4 (0x4)
        ICMP6: Prefix Length = 64 (0x40)
        ICMP6: Site Prefix Length = 0 (0x0)
        ICMP6: 1....... = On-link determination allowed
        ICMP6: .1...... = Autonomous address configuration
        ICMP6: ..0..... = No router address
        ICMP6: ...0.... = Not a site prefix
        ICMP6: .......0 = Not a route prefix
        ICMP6: Valid Lifetime = 2592000 (0x278D00)
        ICMP6: Preferred Lifetime = 604800 (0x93A80)
        ICMP6: Reserved
        ICMP6: Prefix = 3ffe:3103:0:9::
    ICMP6: Source Link-level Address = 00 10 5A 67 55 BC
```

```
        ICMP6: Type = 1 (0x1)
        ICMP6: Length = 1 (0x1)
        ICMP6: Source Link-level Address = 00 10 5A 67 55 BC
```

(4) - Binding Update

```
ETHERNET: ETYPE = 0x86DD : Protocol = Unknown
IP6: Dest Opts; Auth SPI = 0x40a; Dest Opts; Len = 0
    IP6: Version = 6 (0x6)
    IP6: Traffic Class = 0 (0x0)
    IP6: Flow Label = 0 (0x0)
    IP6: Payload Length = 64 (0x40)
    IP6: Next Header = 60 (Destination Options Header)
    IP6: Hop Limit = 255 (0xFF)
    IP6: Source Address = 3ffe:3103:0:9:201:2ff:fe2a:93b9
    IP6: Destination Address = 3ffe:3103:0:8::3
    IP6: Destination Options Header
        IP6: Next Header = 51 (Authentication Header)
        IP6: Length = 2 (0x2)
        IP6: Padding (4 bytes)
            IP6: Type = 1 (PadN)
                IP6: 00...... = Skip option if not recognized
                IP6: ..0..... = Option data does not change enroute
            IP6: Length = 2 (0x2)
            IP6: Data = 00 00
        IP6: Home Address Option
z            IP6: Type = 201
                IP6: 11...... = Discard packet if not recognized, and send ICMP if not multicast
                IP6: ..0..... = Option data does not change enroute
            IP6: Length = 16 (0x10)
            IP6: Home Address = 3ffe:3103:0:8:201:2ff:fe2a:93b9
    IP6: Authentication Header
        IP6: Next Header = 60 (Destination Options Header)
        IP6: Length = 4 (0x4)
        IP6: Reserved
        IP6: SPI = 1034 (0x40A)
        IP6: Seq Num = 4 (0x4)
        IP6: Auth Data = 16 12 37 89 41 0B 1C 88 00 8A 81 DC
    IP6: Destination Options Header
        IP6: Next Header = 59 (None)
        IP6: Length = 1 (0x1)
        IP6: Binding Update Option
            IP6: Type = 198
                IP6: 11...... = Discard packet if not recognized, and send ICMP if not multicast
                IP6: ..0..... = Option data does not change enroute
            IP6: Length = 8 (0x8)
            IP6: Flags
                IP6: 1....... = Request a binding acknowledgement response.
                IP6: .1...... = Request host to act as home agent.
                IP6: ..1..... = Binding update includes a care-of address.
            IP6: Prefix Length = 0 (0x0)
            IP6: Sequence Number = 6 (0x6)
            IP6: Lifetime = 10000 (0x2710)
*           IP6: Care-of Address = 102:0:88a6:8200::5e00:0
```

* The line Care-of Address is a wrong value because the sniffer just understands draft13.

(5) - Binding Acknowledge

```
ETHERNET: ETYPE = 0x86DD : Protocol = Unknown
IP6: Routing (1 left of 1); Auth SPI = 0x409; Dest Opts; Len = 0
    IP6: Version = 6 (0x6)
    IP6: Traffic Class = 0 (0x0)
    IP6: Flow Label = 0 (0x0)
    IP6: Payload Length = 64 (0x40)
    IP6: Next Header = 43 (Routing Header)
    IP6: Hop Limit = 255 (0xFF)
    IP6: Source Address = 3ffe:3103:0:8::3
    IP6: Destination Address = 3ffe:3103:0:9:201:2ff:fe2a:93b9
    IP6: Routing Header
        IP6: Next Header = 51 (Authentication Header)
        IP6: Length = 2 (0x2)
        IP6: Type = 0 (0x0)
        IP6: Segments Left = 1 (0x1)
        IP6: Reserved
        IP6: Route
            IP6: Address = 3ffe:3103:0:8:201:2ff:fe2a:93b9
    IP6: Authentication Header
        IP6: Next Header = 60 (Destination Options Header)
        IP6: Length = 4 (0x4)
        IP6: Reserved
        IP6: SPI = 1033 (0x409)
        IP6: Seq Num = 4 (0x4)
        IP6: Auth Data = A3 C5 76 C8 39 0C 5B 55 67 4D 94 D2
    IP6: Destination Options Header
        IP6: Next Header = 59 (None)
        IP6: Length = 1 (0x1)
        IP6: Padding (1 bytes)
            IP6: Type = 0 (Pad1)
                IP6: 00...... = Skip option if not recognized
                IP6: ..0..... = Option data does not change enroute
        IP6: Binding Acknowledgement Option
            IP6: Type = 7
                IP6: 00...... = Skip option if not recognized
                IP6: ..0..... = Option data does not change enroute
            IP6: Length = 11 (0xB)
            IP6: Status: 0 = Binding accepted.
            IP6: Sequence Number = 6 (0x6)
            IP6: Lifetime = 1000 (0x3E8)
            IP6: Refresh = 800 (0x320)
```

(7) - Echo Request - HA encapsulation

```
ETHERNET: ETYPE = 0x86DD : Protocol = Unknown
IP6: Proto = 0x29; Len = 104
    IP6: Version = 6 (0x6)
    IP6: Traffic Class = 0 (0x0)
    IP6: Flow Label = 0 (0x0)
    IP6: Payload Length = 104 (0x68)
    IP6: Next Header = 41
    IP6: Hop Limit = 255 (0xFF)
    IP6: Source Address = 3ffe:3103:0:8::3
    IP6: Destination Address = 3ffe:3103:0:9:201:2ff:fe2a:93b9
    IP6: Payload: Number of data bytes remaining = 104 (0x0068)
IP6: Proto = ICMP6; Len = 64
    IP6: Version = 6 (0x6)
    IP6: Traffic Class = 0 (0x0)
    IP6: Flow Label = 0 (0x0)
    IP6: Payload Length = 64 (0x40)
    IP6: Next Header = 58 (ICMP6)
    IP6: Hop Limit = 62 (0x3E)
    IP6: Source Address = 3ffe:3103:0:10:260:8ff:fe46:46e7
    IP6: Destination Address = 3ffe:3103:0:8:201:2ff:fe2a:93b9
    IP6: Payload: Number of data bytes remaining = 64 (0x0040)
ICMP6: Echo Request; ID = 64548, Seq = 5376
    ICMP6: Checksum = 0xA23B
    ICMP6: Type = 128 (Echo Request)
    ICMP6: Code = 0 (0x0)
    ICMP6: Identifier = 64548 (0xFC24)
    ICMP6: Sequence Number = 5376 (0x1500)
    ICMP6: Data: Number of data bytes remaining = 56 (0x0038)
```

(8) - Echo Reply + CN Binding Update

```
ETHERNET: ETYPE = 0x86DD : Protocol = Unknown
IP6: Dest Opts; Auth SPI = 0x409; Dest Opts; Proto = ICMP6; Len = 64
    IP6: Version = 6 (0x6)
    IP6: Traffic Class = 0 (0x0)
    IP6: Flow Label = 0 (0x0)
    IP6: Payload Length = 128 (0x80)
    IP6: Next Header = 60 (Destination Options Header)
    IP6: Hop Limit = 64 (0x40)
    IP6: Source Address = 3ffe:3103:0:9:201:2ff:fe2a:93b9
    IP6: Destination Address = 3ffe:3103:0:10:260:8ff:fe46:46e7
    IP6: Destination Options Header
        IP6: Next Header = 51 (Authentication Header)
        IP6: Length = 2 (0x2)
        IP6: Padding (4 bytes)
            IP6: Type = 1 (PadN)
                IP6: 00...... = Skip option if not recognized
                IP6: ..0..... = Option data does not change enroute
            IP6: Length = 2 (0x2)
            IP6: Data = 00 00
        IP6: Home Address Option
            IP6: Type = 201
                IP6: 11...... = Discard packet if not recognized, and send ICMP if not multicast
                IP6: ..0..... = Option data does not change enroute
            IP6: Length = 16 (0x10)
            IP6: Home Address = 3ffe:3103:0:8:201:2ff:fe2a:93b9
    IP6: Authentication Header
        IP6: Next Header = 60 (Destination Options Header)
        IP6: Length = 4 (0x4)
        IP6: Reserved
        IP6: SPI = 1033 (0x409)
        IP6: Seq Num = 1 (0x1)
        IP6: Auth Data = A6 EC A2 38 D7 C7 4A 62 35 65 ED E7
    IP6: Destination Options Header
        IP6: Next Header = 58 (ICMP6)
        IP6: Length = 1 (0x1)
        IP6: Binding Update Option
            IP6: Type = 198
                IP6: 11...... = Discard packet if not recognized, and send ICMP if not multicast
                IP6: ..0..... = Option data does not change enroute
            IP6: Length = 8 (0x8)
            IP6: Flags
                IP6: ..1..... = Binding update includes a care-of address.
            IP6: Prefix Length = 0 (0x0)
            IP6: Sequence Number = 9 (0x9)
            IP6: Lifetime = 1000 (0x3E8)
*           IP6: Care-of Address = 102:0:88a6:8200::5e00:0
ICMP6: Echo Reply; ID = 64548, Seq = 5376
    ICMP6: Checksum = 0xA13B (ERROR: should be 0xA13A)
    ICMP6: Type = 129 (Echo Reply)
    ICMP6: Code = 0 (0x0)
    ICMP6: Identifier = 64548 (0xFC24)
    ICMP6: Sequence Number = 5376 (0x1500)
    ICMP6: Data: Number of data bytes remaining = 56 (0x0038)
```

* The line Care-of Address is a wrong value because the sniffer just understands draft13.

(9) - Echo Request - directly

```
ETHERNET: ETYPE = 0x86DD : Protocol = Unknown
IP6: Routing (1 left of 1); Proto = ICMP6; Len = 64
    IP6: Version = 6 (0x6)
    IP6: Traffic Class = 0 (0x0)
    IP6: Flow Label = 0 (0x0)
    IP6: Payload Length = 88 (0x58)
    IP6: Next Header = 43 (Routing Header)
```

```
    IP6: Hop Limit = 63 (0x3F)
    IP6: Source Address = 3ffe:3103:0:10:260:8ff:fe46:46e7
    IP6: Destination Address = 3ffe:3103:0:9:201:2ff:fe2a:93b9
    IP6: Routing Header
        IP6: Next Header = 58 (ICMP6)
        IP6: Length = 2 (0x2)
        IP6: Type = 0 (0x0)
        IP6: Segments Left = 1 (0x1)
        IP6: Reserved
        IP6: Route
            IP6: Address = 3ffe:3103:0:8:201:2ff:fe2a:93b9
    IP6: Payload: Number of data bytes remaining = 64 (0x0040)
ICMP6: Echo Request; ID = 64548, Seq = 6144
    ICMP6: Checksum = 0x983B (ERROR: should be 0x983A)
    ICMP6: Type = 128 (Echo Request)
    ICMP6: Code = 0 (0x0)
    ICMP6: Identifier = 64548 (0xFC24)
    ICMP6: Sequence Number = 6144 (0x1800)
    ICMP6: Data: Number of data bytes remaining = 56 (0x0038)
```

(10) - Echo Reply

```
ETHERNET: ETYPE = 0x86DD : Protocol = Unknown
IP6: Dest Opts; Proto = ICMP6; Len = 64
    IP6: Version = 6 (0x6)
    IP6: Traffic Class = 0 (0x0)
    IP6: Flow Label = 0 (0x0)
    IP6: Payload Length = 88 (0x58)
    IP6: Next Header = 60 (Destination Options Header)
    IP6: Hop Limit = 64 (0x40)
    IP6: Source Address = 3ffe:3103:0:9:201:2ff:fe2a:93b9
    IP6: Destination Address = 3ffe:3103:0:10:260:8ff:fe46:46e7
    IP6: Destination Options Header
        IP6: Next Header = 58 (ICMP6)
        IP6: Length = 2 (0x2)
        IP6: Padding (4 bytes)
            IP6: Type = 1 (PadN)
                IP6: 00...... = Skip option if not recognized
                IP6: ..0..... = Option data does not change enroute
            IP6: Length = 2 (0x2)
            IP6: Data = 00 00
        IP6: Home Address Option
            IP6: Type = 201
                IP6: 11...... = Discard packet if not recognized, and send ICMP if not multicast
                IP6: ..0..... = Option data does not change enroute
            IP6: Length = 16 (0x10)
            IP6: Home Address = 3ffe:3103:0:8:201:2ff:fe2a:93b9
    IP6: Payload: Number of data bytes remaining = 64 (0x0040)
ICMP6: Echo Reply; ID = 64548, Seq = 6144
    ICMP6: Checksum = 0x973B (ERROR: should be 0x973A)
    ICMP6: Type = 129 (Echo Reply)
    ICMP6: Code = 0 (0x0)
    ICMP6: Identifier = 64548 (0xFC24)
    ICMP6: Sequence Number = 6144 (0x1800)
    ICMP6: Data: Number of data bytes remaining = 56 (0x0038)
```

(12) - Binding Update

```
ETHERNET: ETYPE = 0x86DD : Protocol = Unknown
IP6: Dest Opts; Auth SPI = 0x40a; Dest Opts; Len = 0
    IP6: Version = 6 (0x6)
    IP6: Traffic Class = 0 (0x0)
    IP6: Flow Label = 0 (0x0)
    IP6: Payload Length = 64 (0x40)
    IP6: Next Header = 60 (Destination Options Header)
    IP6: Hop Limit = 255 (0xFF)
    IP6: Source Address = 3ffe:3103:0:8:201:2ff:fe2a:93b9
    IP6: Destination Address = 3ffe:3103:0:8::3
    IP6: Destination Options Header
        IP6: Next Header = 51 (Authentication Header)
        IP6: Length = 2 (0x2)
        IP6: Padding (4 bytes)
            IP6: Type = 1 (PadN)
                IP6: 00...... = Skip option if not recognized
                IP6: ..0..... = Option data does not change enroute
            IP6: Length = 2 (0x2)
            IP6: Data = 00 00
        IP6: Home Address Option
            IP6: Type = 201
                IP6: 11...... = Discard packet if not recognized, and send ICMP if not multicast
                IP6: ..0..... = Option data does not change enroute
            IP6: Length = 16 (0x10)
            IP6: Home Address = 3ffe:3103:0:8:201:2ff:fe2a:93b9
    IP6: Authentication Header
        IP6: Next Header = 60 (Destination Options Header)
        IP6: Length = 4 (0x4)
        IP6: Reserved
        IP6: SPI = 1034 (0x40A)
        IP6: Seq Num = 4 (0x4)
        IP6: Auth Data = 16 12 37 89 41 0B 1C 88 00 8A 81 DC
    IP6: Destination Options Header
        IP6: Next Header = 59 (None)
        IP6: Length = 1 (0x1)
        IP6: Binding Update Option
            IP6: Type = 198
                IP6: 11...... = Discard packet if not recognized, and send ICMP if not multicast
                IP6: ..0..... = Option data does not change enroute
            IP6: Length = 8 (0x8)
            IP6: Flags
                IP6: 1....... = Request a binding acknowledgement response.
```

```
                IP6: .1...... = Request host to act as home agent.
                IP6: ..1..... = Binding update includes a care-of address.
            IP6: Prefix Length = 0 (0x0)
            IP6: Sequence Number = 6 (0x6)
            IP6: Lifetime = 10000 (0x2710)
*           IP6: Care-of Address = 102:0:88a6:8200::5e00:0
```

* The line Care-of Address is a wrong value because the sniffer just understands draft13.

(13) - Binding Acknowledge

```
ETHERNET: ETYPE = 0x86DD : Protocol = Unknown
IP6: Routing (1 left of 1); Auth SPI = 0x409; Dest Opts; Len = 0
    IP6: Version = 6 (0x6)
    IP6: Traffic Class = 0 (0x0)
    IP6: Flow Label = 0 (0x0)
    IP6: Payload Length = 64 (0x40)
    IP6: Next Header = 43 (Routing Header)
    IP6: Hop Limit = 255 (0xFF)
    IP6: Source Address = 3ffe:3103:0:8::3
    IP6: Destination Address = 3ffe:3103:0:8:201:2ff:fe2a:93b9
    IP6: Authentication Header
        IP6: Next Header = 60 (Destination Options Header)
        IP6: Length = 4 (0x4)
        IP6: Reserved
        IP6: SPI = 1033 (0x409)
        IP6: Seq Num = 4 (0x4)
        IP6: Auth Data = A3 C5 76 C8 39 0C 5B 55 67 4D 94 D2
    IP6: Destination Options Header
        IP6: Next Header = 59 (None)
        IP6: Length = 1 (0x1)
        IP6: Padding (1 bytes)
            IP6: Type = 0 (Pad1)
                IP6: 00...... = Skip option if not recognized
                IP6: ..0..... = Option data does not change enroute
        IP6: Binding Acknowledgement Option
            IP6: Type = 7
                IP6: 00...... = Skip option if not recognized
                IP6: ..0..... = Option data does not change enroute
            IP6: Length = 11 (0xB)
            IP6: Status: 0 = Binding accepted.
            IP6: Sequence Number = 6 (0x6)
            IP6: Lifetime = 1000 (0x3E8)
            IP6: Refresh = 800 (0x320)
```

(14) - Echo Request

```
ETHERNET: ETYPE = 0x86DD : Protocol = Unknown
IP6: Proto = ICMP6; Len = 40
    IP6: Version = 6 (0x6)
    IP6: Traffic Class = 0 (0x0)
    IP6: Flow Label = 0 (0x0)
    IP6: Payload Length = 40 (0x28)
    IP6: Next Header = 58 (ICMP6)
    IP6: Hop Limit = 126 (0x7E)
    IP6: Source Address = 3ffe:3103:0:10:260:8ff:fe46:46e7
    IP6: Destination Address = 3ffe:3103:0:8:201:2ff:fe2a:93b9
    IP6: Payload: Number of data bytes remaining = 40 (0x0028)
ICMP6: Echo Request; ID = 23, Seq = 60579
    ICMP6: Checksum = 0x8208
    ICMP6: Type = 128 (Echo Request)
    ICMP6: Code = 0 (0x0)
    ICMP6: Identifier = 23 (0x17)
    ICMP6: Sequence Number = 60579 (0xECA3)
    ICMP6: Data: Number of data bytes remaining = 32 (0x0020)
```

(15) - Echo Reply

```
ETHERNET: ETYPE = 0x86DD : Protocol = Unknown
IP6: Proto = ICMP6; Len = 40
    IP6: Version = 6 (0x6)
    IP6: Traffic Class = 0 (0x0)
    IP6: Flow Label = 0 (0x0)
    IP6: Payload Length = 40 (0x28)
    IP6: Next Header = 58 (ICMP6)
    IP6: Hop Limit = 64 (0x40)
    IP6: Source Address = 3ffe:3103:0:8:201:2ff:fe2a:93b9
    IP6: Destination Address = 3ffe:3103:0:10:260:8ff:fe46:46e7
    IP6: Payload: Number of data bytes remaining = 40 (0x0028)
ICMP6: Echo Reply; ID = 23, Seq = 60579
    ICMP6: Checksum = 0x8108
    ICMP6: Type = 129 (Echo Reply)
    ICMP6: Code = 0 (0x0)
    ICMP6: Identifier = 23 (0x17)
    ICMP6: Sequence Number = 60579 (0xECA3)
    ICMP6: Data: Number of data bytes remaining = 32 (0x0020)
```

6.2.4.1.4 Scenario 4

In the scenario number 4, we test the situation where there are two MNs on the network. One is in its HN and, the second, in a FN. The Figure 6-8 shows graphically this scenario.

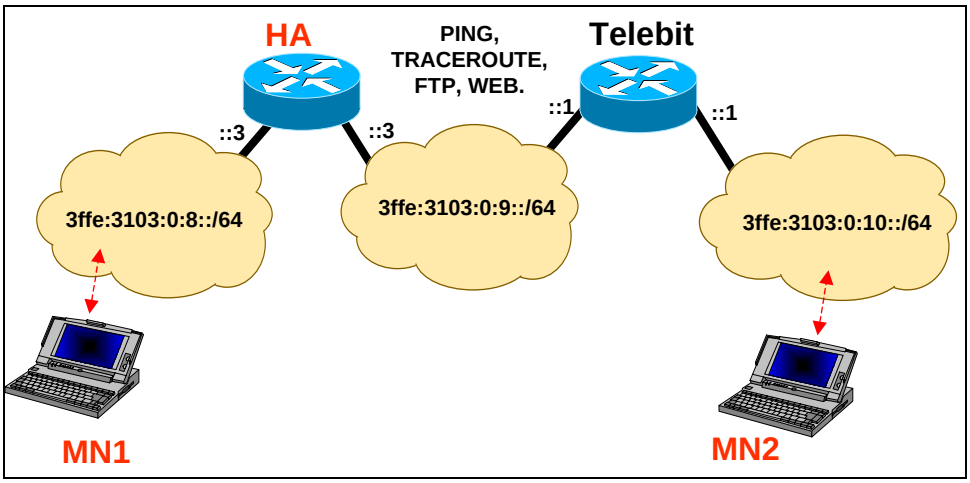


Figure 6-8: LAN: Scenario 4: The MN1 is on the HN and the MN2 is on a FN.

Characteristics of equipment involved:

Equipment Characteristics	HA	MN1	MN2	FN Router
Platform	*i386*	*i386*	*i386*	*Telebit*
Model	*"Blank" PC*	*"Blank" PC*	*"Blank" PC*	*AXI 462/3*
CPU	*Pentium II 400 MHz (cache 512 KB)*	*Pentium III 650 MHz (cache 256 KB)*	*Pentium II 266 MHz (cache 512 KB)*	*RISC (TBC 103)*
Memory	*64 RAM (swap 256 MB)*	*128 RAM (swap 256 MB)*	*64 RAM (swap 256 MB)*	*16 MB*
Network Interfaces Card	*2 Ethernet cards 3COM series 59x (driver 3c59x)*	*1 Ethernet card 3COM series 59x (driver 3c59x)*	*1 Ethernet card 3COM series 59x (driver 3c59x)*	*2 Ethernet cards (TCB 120) Motorola Power PC 750*
Operating System / Kernel	*Linux 7.2 (kernel 2.4.16)*	*Linux 7.2 (kernel 2.4.16)*	*Linux 7.2 (kernel 2.4.16)*	*Telebit OS swtbcpaxos 2.6a14*

MN1 location:

Home Network (HN) 3ffe:3103:0:8::/64 (via LAN).

MN2 location:

Foreign Network (FN) - 3ffe:3103:0:10::/64 (via LAN).

Applications:

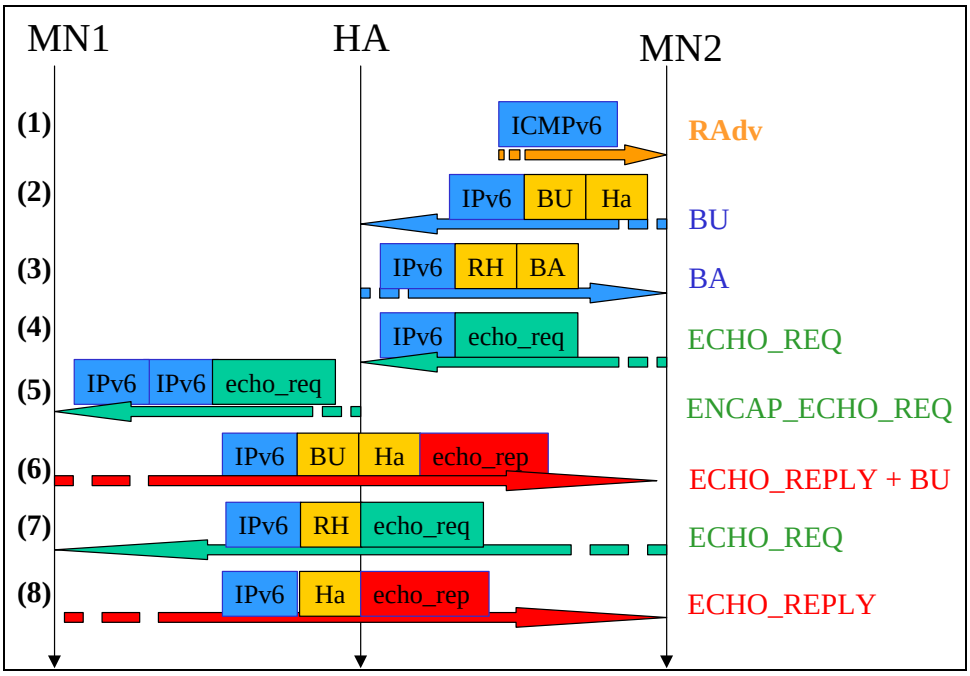
PINGs, FTPs and WWW.

External Behaviour:

Externally it works correctly with all the applications/services tested. After the movement to attach the nodes as specified in the Figure, when the MN is down about 2 - 5 seconds, it works properly.
Actually, in terms of functionality, the situation is exactly the same as if the MN1 was a CN.

Schematic DataFlow:

* Ping from the MN2 to MN1.



Packet Analysis:

The Router Advertisements (RAs) are being sent periodically by the Telebit. The prefix announced is the 3ffe:3103:0:10::/64.

A Binding Update (BU) is sent from the MN to the HA, in order to register the CoA acquired, using a Ha Destination Option (DH).

A Binding Acknowledge (BA) from the HA confirms the BU. The HA use a Routing Header (RH) to forward the packet.

The first time that the MN sends an *Echo Request* to the MN, it is sent to the MN Home link.

The packet is there intercepted by the HA and, is encapsulated, using IPv6 over IPv6, and sent to the MN.

When the MN receives a packet encapsulated by his HA, it knows that there is a CN that doesn't know it's actual location. For this reason a BU is sent to the CN to actualize the CN Binding Cache (BC) - for this is used again the Ha DH.

The steps are completed and, the next packets from the CN to the MN are sent directly using a Routing Header (RH).

The packets from the MN to the CN are sent directly using a Home address (Ha) DH.

Sniffed packets:

* The particular packets here presented are from the PING application.

(1) - Router Advertisement

```
ETHERNET: ETYPE = 0x86DD : Protocol = Unknown
IP6: Proto = ICMP6; Len = 56
    IP6: Version = 6 (0x6)
    IP6: Traffic Class = 0 (0x0)
    IP6: Flow Label = 0 (0x0)
    IP6: Payload Length = 56 (0x38)
    IP6: Next Header = 58 (ICMP6)
    IP6: Hop Limit = 255 (0xFF)
    IP6: Source Address = fe80::2c0:33ff:fe0c:a0
    IP6: Destination Address = ff02::1
    IP6: Payload: Number of data bytes remaining = 56 (0x0038)
ICMP6: Router Advertisement
    ICMP6: Checksum = 0x5AA8
    ICMP6: Type = 134 (Router Advertisement)
    ICMP6: Code = 0 (0x0)
    ICMP6: Current Hop Limit = 64 (0x40)
    ICMP6: 0....... = Not managed address config
```

```
    ICMP6: .0...... = Not other stateful config
    ICMP6: Router Lifetime = 600 (0x258)
    ICMP6: Reachable Time = 0 (0x0)
    ICMP6: Retransmission Timer = 0 (0x0)
    ICMP6: Prefix = 3ffe:3103:0:10::
        ICMP6: Type = 3 (0x3)
        ICMP6: Length = 4 (0x4)
        ICMP6: Prefix Length = 64 (0x40)
        ICMP6: Site Prefix Length = 0 (0x0)
        ICMP6: 1....... = On-link determination allowed
        ICMP6: .1...... = Autonomous address configuration
        ICMP6: ..0..... = No router address
        ICMP6: ...0.... = Not a site prefix
        ICMP6: .......0 = Not a route prefix
        ICMP6: Valid Lifetime = 2592000 (0x278D00)
        ICMP6: Preferred Lifetime = 604800 (0x93A80)
        ICMP6: Reserved
        ICMP6: Prefix = 3ffe:3103:0:10::
    ICMP6: Source Link-level Address = 00 C0 33 0C 00 10
        ICMP6: Type = 1 (0x1)
        ICMP6: Length = 1 (0x1)
        ICMP6: Source Link-level Address = 00 C0 33 0C 00 10
```

(2) - Binding Update

```
ETHERNET: ETYPE = 0x86DD : Protocol = Unknown
IP6: Dest Opts; Auth SPI = 0x40a; Dest Opts; Len = 0
    IP6: Version = 6 (0x6)
    IP6: Traffic Class = 0 (0x0)
    IP6: Flow Label = 0 (0x0)
    IP6: Payload Length = 64 (0x40)
    IP6: Next Header = 60 (Destination Options Header)
    IP6: Hop Limit = 255 (0xFF)
    IP6: Source Address = 3ffe:3103:0:10:260:8ff:fe46:46e7
    IP6: Destination Address = 3ffe:3103:0:8::3
    IP6: Destination Options Header
        IP6: Next Header = 51 (Authentication Header)
        IP6: Length = 2 (0x2)
        IP6: Padding (4 bytes)
            IP6: Type = 1 (PadN)
                IP6: 00...... = Skip option if not recognized
                IP6: ..0..... = Option data does not change enroute
            IP6: Length = 2 (0x2)
            IP6: Data = 00 00
        IP6: Home Address Option
            IP6: Type = 201
                IP6: 11...... = Discard packet if not recognized, and send ICMP if not multicast
                IP6: ..0..... = Option data does not change enroute
            IP6: Length = 16 (0x10)
            IP6: Home Address = 3ffe:3103:0:8:260:8ff:fe46:46e7
    IP6: Authentication Header
        IP6: Next Header = 60 (Destination Options Header)
        IP6: Length = 4 (0x4)
        IP6: Reserved
        IP6: SPI = 1034 (0x40A)
        IP6: Seq Num = 4 (0x4)
        IP6: Auth Data = 16 12 37 89 41 0B 1C 88 00 8A 81 DC
    IP6: Destination Options Header
        IP6: Next Header = 59 (None)
        IP6: Length = 1 (0x1)
        IP6: Binding Update Option
            IP6: Type = 198
                IP6: 11...... = Discard packet if not recognized, and send ICMP if not multicast
                IP6: ..0..... = Option data does not change enroute
            IP6: Length = 8 (0x8)
            IP6: Flags
                IP6: 1....... = Request a binding acknowledgement response.
                IP6: .1...... = Request host to act as home agent.
                IP6: ..1..... = Binding update includes a care-of address.
            IP6: Prefix Length = 0 (0x0)
            IP6: Sequence Number = 6 (0x6)
            IP6: Lifetime = 10000 (0x2710)
*           IP6: Care-of Address = 102:0:88a6:8200::5e00:0
```

* The line Care-of Address is a wrong value because the sniffer just understands draft13.

(3) - Binding Acknowledge

```
ETHERNET: ETYPE = 0x86DD : Protocol = Unknown
IP6: Routing (1 left of 1); Auth SPI = 0x409; Dest Opts; Len = 0
    IP6: Version = 6 (0x6)
    IP6: Traffic Class = 0 (0x0)
    IP6: Flow Label = 0 (0x0)
    IP6: Payload Length = 64 (0x40)
    IP6: Next Header = 43 (Routing Header)
    IP6: Hop Limit = 255 (0xFF)
    IP6: Source Address = 3ffe:3103:0:8::3
    IP6: Destination Address = 3ffe:3103:0:10:260:8ff:fe46:46e7
    IP6: Routing Header
        IP6: Next Header = 51 (Authentication Header)
        IP6: Length = 2 (0x2)
        IP6: Type = 0 (0x0)
        IP6: Segments Left = 1 (0x1)
        IP6: Reserved
        IP6: Route
            IP6: Address = 3ffe:3103:0:8:260:8ff:fe46:46e7
    IP6: Authentication Header
        IP6: Next Header = 60 (Destination Options Header)
        IP6: Length = 4 (0x4)
        IP6: Reserved
```

```
        IP6: SPI = 1033 (0x409)
        IP6: Seq Num = 4 (0x4)
        IP6: Auth Data = A3 C5 76 C8 39 0C 5B 55 67 4D 94 D2
    IP6: Destination Options Header
        IP6: Next Header = 59 (None)
        IP6: Length = 1 (0x1)
        IP6: Padding (1 bytes)
            IP6: Type = 0 (Pad1)
                IP6: 00...... = Skip option if not recognized
                IP6: ..0..... = Option data does not change enroute
        IP6: Binding Acknowledgement Option
            IP6: Type = 7
                IP6: 00...... = Skip option if not recognized
                IP6: ..0..... = Option data does not change enroute
            IP6: Length = 11 (0xB)
            IP6: Status: 0 = Binding accepted.
            IP6: Sequence Number = 6 (0x6)
            IP6: Lifetime = 1000 (0x3E8)
            IP6: Refresh = 800 (0x320)
```

(5) - Echo Request - HA encapsulation

```
ETHERNET: ETYPE = 0x86DD : Protocol = Unknown
IP6: Proto = 0x29; Len = 104
    IP6: Version = 6 (0x6)
    IP6: Traffic Class = 0 (0x0)
    IP6: Flow Label = 0 (0x0)
    IP6: Payload Length = 104 (0x68)
    IP6: Next Header = 41
    IP6: Hop Limit = 255 (0xFF)
    IP6: Source Address = 3ffe:3103:0:8::3
    IP6: Destination Address = 3ffe:3103:0:8:201:2ff:fe2a:93b9
    IP6: Payload: Number of data bytes remaining = 104 (0x0068)
IP6: Proto = ICMP6; Len = 64
    IP6: Version = 6 (0x6)
    IP6: Traffic Class = 0 (0x0)
    IP6: Flow Label = 0 (0x0)
    IP6: Payload Length = 64 (0x40)
    IP6: Next Header = 58 (ICMP6)
    IP6: Hop Limit = 62 (0x3E)
    IP6: Source Address = 3ffe:3103:0:10:260:8ff:fe46:46e7
    IP6: Destination Address = 3ffe:3103:0:8:201:2ff:fe2a:93b9
    IP6: Payload: Number of data bytes remaining = 64 (0x0040)
ICMP6: Echo Request; ID = 64548, Seq = 5376
    ICMP6: Checksum = 0xA23B
    ICMP6: Type = 128 (Echo Request)
    ICMP6: Code = 0 (0x0)
    ICMP6: Identifier = 64548 (0xFC24)
    ICMP6: Sequence Number = 5376 (0x1500)
    ICMP6: Data: Number of data bytes remaining = 56 (0x0038)
```

(6) - Echo Replay - MN2 Binding Update

```
ETHERNET: ETYPE = 0x86DD : Protocol = Unknown
IP6: Dest Opts; Auth SPI = 0x409; Dest Opts; Proto = ICMP6; Len = 64
    IP6: Version = 6 (0x6)
    IP6: Traffic Class = 0 (0x0)
    IP6: Flow Label = 0 (0x0)
    IP6: Payload Length = 128 (0x80)
    IP6: Next Header = 60 (Destination Options Header)
    IP6: Hop Limit = 64 (0x40)
    IP6: Source Address = 3ffe:3103:0:9:201:2ff:fe2a:93b9
    IP6: Destination Address = 3ffe:3103:0:10:260:8ff:fe46:46e7
    IP6: Destination Options Header
        IP6: Next Header = 51 (Authentication Header)
        IP6: Length = 2 (0x2)
        IP6: Padding (4 bytes)
            IP6: Type = 1 (PadN)
                IP6: 00...... = Skip option if not recognized
                IP6: ..0..... = Option data does not change enroute
            IP6: Length = 2 (0x2)
            IP6: Data = 00 00
        IP6: Home Address Option
            IP6: Type = 201
                IP6: 11...... = Discard packet if not recognized, and send ICMP if not multicast
                IP6: ..0..... = Option data does not change enroute
            IP6: Length = 16 (0x10)
            IP6: Home Address = 3ffe:3103:0:8:201:2ff:fe2a:93b9
    IP6: Authentication Header
        IP6: Next Header = 60 (Destination Options Header)
        IP6: Length = 4 (0x4)
        IP6: Reserved
        IP6: SPI = 1033 (0x409)
        IP6: Seq Num = 1 (0x1)
        IP6: Auth Data = A6 EC A2 38 D7 C7 4A 62 35 65 ED E7
    IP6: Destination Options Header
        IP6: Next Header = 58 (ICMP6)
        IP6: Length = 1 (0x1)
        IP6: Binding Update Option
            IP6: Type = 198
                IP6: 11...... = Discard packet if not recognized, and send ICMP if not multicast
                IP6: ..0..... = Option data does not change enroute
            IP6: Length = 8 (0x8)
            IP6: Flags
                IP6: ..1..... = Binding update includes a care-of address.
            IP6: Prefix Length = 0 (0x0)
            IP6: Sequence Number = 9 (0x9)
            IP6: Lifetime = 1000 (0x3E8)
*           IP6: Care-of Address = 102:0:88a6:8200::5e00:0
ICMP6: Echo Reply; ID = 64548, Seq = 5376
```

```
    ICMP6: Checksum = 0xA13B (ERROR: should be 0xA13A)
    ICMP6: Type = 129 (Echo Reply)
    ICMP6: Code = 0 (0x0)
    ICMP6: Identifier = 64548 (0xFC24)
    ICMP6: Sequence Number = 5376 (0x1500)
    ICMP6: Data: Number of data bytes remaining = 56 (0x0038)
```

* The line Care-of Address is a wrong value because the sniffer just understands draft13.

(7) - Echo Request

```
ETHERNET: ETYPE = 0x86DD : Protocol = Unknown
IP6: Dest Opts; Proto = ICMP6; Len = 64
    IP6: Version = 6 (0x6)
    IP6: Traffic Class = 0 (0x0)
    IP6: Flow Label = 0 (0x0)
    IP6: Payload Length = 88 (0x58)
    IP6: Next Header = 60 (Destination Options Header)
    IP6: Hop Limit = 64 (0x40)
    IP6: Source Address = 3ffe:3103:0:9:201:2ff:fe2a:93b9
    IP6: Destination Address = 3ffe:3103:0:10:260:8ff:fe46:46e7
    IP6: Destination Options Header
        IP6: Next Header = 58 (ICMP6)
        IP6: Length = 2 (0x2)
        IP6: Padding (4 bytes)
            IP6: Type = 1 (PadN)
                IP6: 00...... = Skip option if not recognized
                IP6: ..0..... = Option data does not change enroute
            IP6: Length = 2 (0x2)
            IP6: Data = 00 00
        IP6: Home Address Option
            IP6: Type = 201
                IP6: 11...... = Discard packet if not recognized, and send ICMP if not multicast
                IP6: ..0..... = Option data does not change enroute
            IP6: Length = 16 (0x10)
            IP6: Home Address = 3ffe:3103:0:8:201:2ff:fe2a:93b9
    IP6: Payload: Number of data bytes remaining = 64 (0x0040)
ICMP6: Echo Request; ID = 64548, Seq = 6144
    ICMP6: Checksum = 0x983B (ERROR: should be 0x983A)
    ICMP6: Type = 128 (Echo Request)
    ICMP6: Code = 0 (0x0)
    ICMP6: Identifier = 64548 (0xFC24)
    ICMP6: Sequence Number = 6144 (0x1800)
    ICMP6: Data: Number of data bytes remaining = 56 (0x0038)
```

(8) - Echo Replay

```
ETHERNET: ETYPE = 0x86DD : Protocol = Unknown
IP6: Routing (1 left of 1); Proto = ICMP6; Len = 64
    IP6: Version = 6 (0x6)
    IP6: Traffic Class = 0 (0x0)
    IP6: Flow Label = 0 (0x0)
    IP6: Payload Length = 88 (0x58)
    IP6: Next Header = 43 (Routing Header)
    IP6: Hop Limit = 63 (0x3F)
    IP6: Source Address = 3ffe:3103:0:10:260:8ff:fe46:46e7
    IP6: Destination Address = 3ffe:3103:0:9:201:2ff:fe2a:93b9
    IP6: Routing Header
        IP6: Next Header = 58 (ICMP6)
        IP6: Length = 2 (0x2)
        IP6: Type = 0 (0x0)
        IP6: Segments Left = 1 (0x1)
        IP6: Reserved
        IP6: Route
            IP6: Address = 3ffe:3103:0:8:201:2ff:fe2a:93b9
    IP6: Payload: Number of data bytes remaining = 64 (0x0040)
ICMP6: Echo Reply; ID = 64548, Seq = 6144
    ICMP6: Checksum = 0x973B (ERROR: should be 0x973A)
    ICMP6: Type = 129 (Echo Reply)
    ICMP6: Code = 0 (0x0)
    ICMP6: Identifier = 64548 (0xFC24)
    ICMP6: Sequence Number = 6144 (0x1800)
    ICMP6: Data: Number of data bytes remaining = 56 (0x0038)
```

6.2.4.1.5 Scenario 5

In the scenario number 5, we will test the situation when there are two MNs on the network. One is in a FN and, the second, in another FN; i.e. both are out of their Homes. The Figure 6-9 shows graphically this scenario.

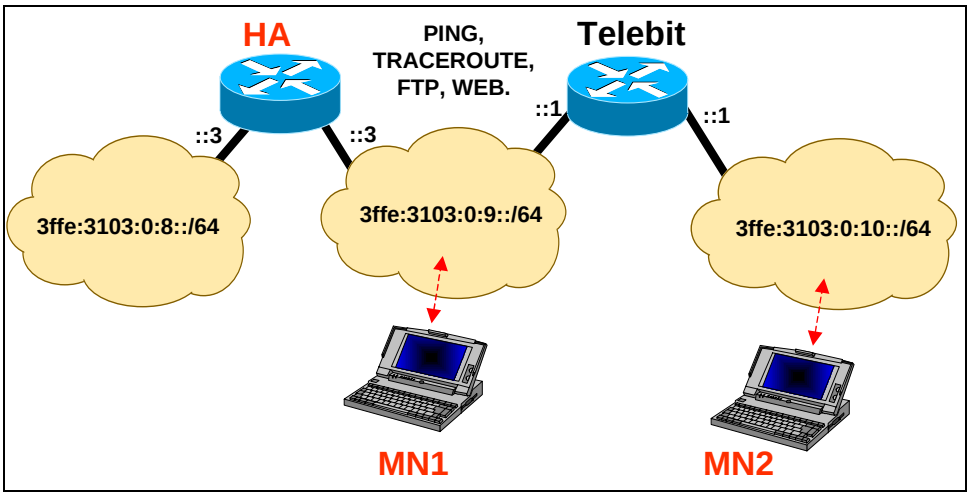


Figure 6-9: LAN: Scenario 5: The MN1 is on the FN1 and the MN2 is on a FN2.

Characteristics of equipment involved:

Equipment Characteristics	HA	MN1	MN2	FN Router
Platform	*i386*	*i386*	*i386*	*Telebit*
Model	*"Blank" PC*	*"Blank" PC*	*"Blank" PC*	*AXI 462/3*
CPU	*Pentium II 400 MHz (cache 512 KB)*	*Pentium III 650 MHz (cache 256 KB)*	*Pentium II 266 MHz (cache 512 KB)*	*RISC (TBC 103)*
Memory	*64 RAM (swap 256 MB)*	*128 RAM (swap 256 MB)*	*64 RAM (swap 256 MB)*	*16 MB*
Network Interfaces Card	*2 Ethernet cards 3COM series 59x (driver 3c59x)*	*1 Ethernet card 3COM series 59x (driver 3c59x)*	*1 Ethernet card 3COM series 59x (driver 3c59x)*	*2 Ethernet cards (TCB 120) Motorola Power PC 750*
Operating System / Kernel	*Linux 7.2 (kernel 2.4.16)*	*Linux 7.2 (kernel 2.4.16)*	*Linux 7.2 (kernel 2.4.16)*	*Telebit OS swtbcpaxos 2.6a14*

MN1 location:

Foreign Network 1 (FN) - 3ffe:3103:0:9::/64 (via LAN).

MN2 location:

Foreign Network 2 (FN) - 3ffe:3103:0:10::/64 (via LAN).

Applications:

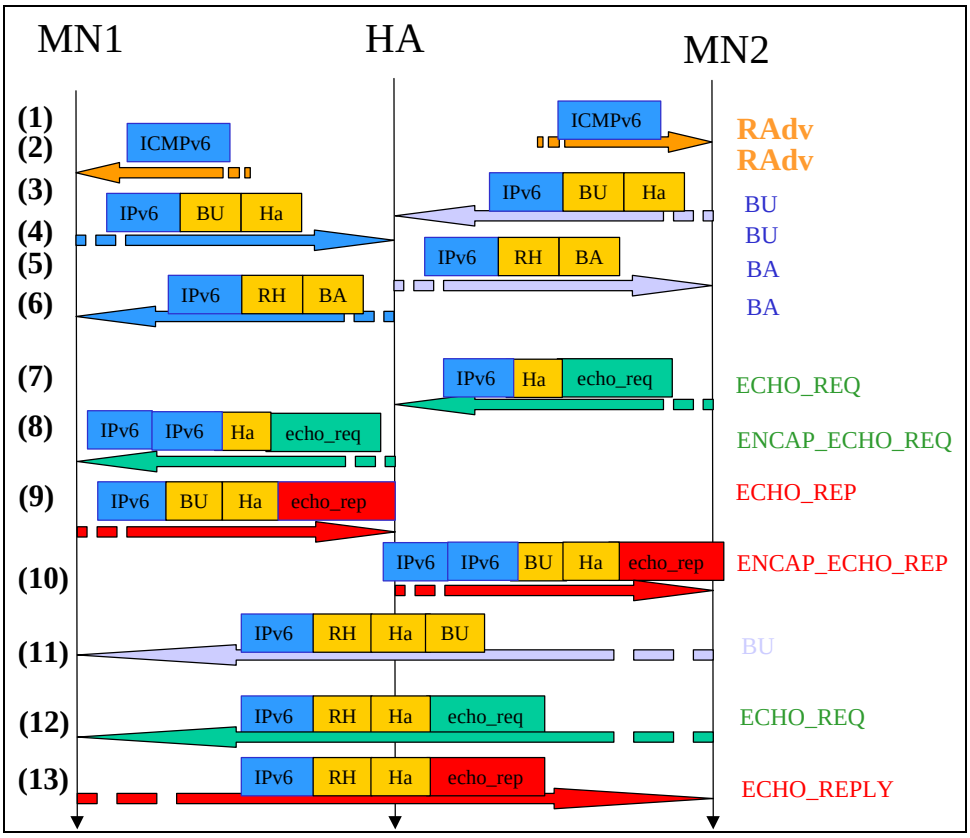
PINGs, FTPs and WWW.

External Behaviour:

Externally it works correctly with all the applications/services tested. After the movement to attach the nodes as specified in the Figure (2 - 5 seconds), it works properly.

Schematic DataFlow:

* Ping from the MN2 to MN1.



Packet Analysis:

The Router Advertisements (RAs) are being sent periodically by the Telebit. The prefix announced is the 3ffe:3103:0:10::/64.

The Router Advertisements (RAs) are being sent periodically by the Telebit. The prefix announced is the 3ffe:3103:0:9::/64.

A Binding Update (BU) is sent from the MN2 to the HA, in order to register the CoA acquired, using a Ha Destination Option (DH).

A Binding Update (BU) is sent from the MN1 to the HA, in order to register the CoA acquired, using a Ha Destination Option (DH).

A Binding Acknowledge (BA) from the HA confirms the BU of the MN2. The HA use a Routing Header (RH) to forward the packet.

A Binding Acknowledge (BA) from the HA confirms the BU of the MN1. The HA use a Routing Header (RH) to forward the packet.

The first time that the MN2 sends an *Echo Request* to the MN1, it is sent to the MN1 Home link. Note that the MN2 add a Ha header to the packet, because it knows that is located in a FN.

The packet is there intercepted by the HA and, is encapsulated, using IPv6 over IPv6, and sent to the MN1.

The first time that the MN1 sends an *Echo Reply* to the MN2, it is sent to the MN2 Home link. Note that the MN1 also add a Ha header to the packet, because it knows that is located in a FN.

The packet is there intercepted by the HA and, is encapsulated, using IPv6 over IPv6, and sent to the MN2.

The MN2 send a packet to add an entry in the MN1 BC. Notice that the packet doesn't have payload, but just because in this moment was not packet to be sent. Otherwise the piggybacking mechanism could be used.

Since the Binding Caches (BCs) of each MNs are updates, the communication starts to be made directly. From the MN2 to the MN1, are always sent a Ha to identify the MN2 Home address, and an RH to transport the Ha of the MN1.

Equally, the MN1 send a Ha to identify its Ha and a RH to identify the MN2 Ha.

Sniffed packets:

* The particular packets here presented are from the PING application.

(1) - Router Advertisement

```
ETHERNET: ETYPE = 0x86DD : Protocol = Unknown
IP6: Proto = ICMP6; Len = 56
    IP6: Version = 6 (0x6)
    IP6: Traffic Class = 0 (0x0)
    IP6: Flow Label = 0 (0x0)
    IP6: Payload Length = 56 (0x38)
    IP6: Next Header = 58 (ICMP6)
    IP6: Hop Limit = 255 (0xFF)
    IP6: Source Address = fe80::2c0:33ff:fe0c:a0
    IP6: Destination Address = ff02::1
    IP6: Payload: Number of data bytes remaining = 56 (0x0038)
ICMP6: Router Advertisement
    ICMP6: Checksum = 0x5AA8
    ICMP6: Type = 134 (Router Advertisement)
    ICMP6: Code = 0 (0x0)
    ICMP6: Current Hop Limit = 64 (0x40)
    ICMP6: 0....... = Not managed address config
    ICMP6: .0...... = Not other stateful config
    ICMP6: Router Lifetime = 600 (0x258)
    ICMP6: Reachable Time = 0 (0x0)
    ICMP6: Retransmission Timer = 0 (0x0)
    ICMP6: Prefix = 3ffe:3103:0:10::
        ICMP6: Type = 3 (0x3)
        ICMP6: Length = 4 (0x4)
        ICMP6: Prefix Length = 64 (0x40)
        ICMP6: Site Prefix Length = 0 (0x0)
        ICMP6: 1....... = On-link determination allowed
        ICMP6: .1...... = Autonomous address configuration
        ICMP6: ..0..... = No router address
        ICMP6: ...0.... = Not a site prefix
        ICMP6: .......0 = Not a route prefix
        ICMP6: Valid Lifetime = 2592000 (0x278D00)
        ICMP6: Preferred Lifetime = 604800 (0x93A80)
        ICMP6: Reserved
        ICMP6: Prefix = 3ffe:3103:0:10::
    ICMP6: Source Link-level Address = 00 C0 33 0C 00 A0
        ICMP6: Type = 1 (0x1)
        ICMP6: Length = 1 (0x1)
        ICMP6: Source Link-level Address = 00 C0 33 0C 00 A0
```

(2) - Router Advertisement

```
ETHERNET: ETYPE = 0x86DD : Protocol = Unknown
IP6: Proto = ICMP6; Len = 56
    IP6: Version = 6 (0x6)
    IP6: Traffic Class = 0 (0x0)
    IP6: Flow Label = 0 (0x0)
    IP6: Payload Length = 56 (0x38)
    IP6: Next Header = 58 (ICMP6)
    IP6: Hop Limit = 255 (0xFF)
    IP6: Source Address = fe80::210:5aff:fe67:55bc
    IP6: Destination Address = ff02::1
    IP6: Payload: Number of data bytes remaining = 56 (0x0038)
ICMP6: Router Advertisement
    ICMP6: Checksum = 0x5AA8
    ICMP6: Type = 134 (Router Advertisement)
    ICMP6: Code = 0 (0x0)
    ICMP6: Current Hop Limit = 64 (0x40)
    ICMP6: 0....... = Not managed address config
    ICMP6: .0...... = Not other stateful config
    ICMP6: Router Lifetime = 600 (0x258)
    ICMP6: Reachable Time = 0 (0x0)
    ICMP6: Retransmission Timer = 0 (0x0)
    ICMP6: Prefix = 3ffe:3103:0:9::
        ICMP6: Type = 3 (0x3)
        ICMP6: Length = 4 (0x4)
        ICMP6: Prefix Length = 64 (0x40)
        ICMP6: Site Prefix Length = 0 (0x0)
        ICMP6: 1....... = On-link determination allowed
        ICMP6: .1...... = Autonomous address configuration
        ICMP6: ..0..... = No router address
        ICMP6: ...0.... = Not a site prefix
        ICMP6: .......0 = Not a route prefix
        ICMP6: Valid Lifetime = 2592000 (0x278D00)
        ICMP6: Preferred Lifetime = 604800 (0x93A80)
        ICMP6: Reserved
        ICMP6: Prefix = 3ffe:3103:0:9::
    ICMP6: Source Link-level Address = 00 10 5A 67 55 BC
        ICMP6: Type = 1 (0x1)
        ICMP6: Length = 1 (0x1)
        ICMP6: Source Link-level Address = 00 10 5A 67 55 BC
```

(3) - MN2 Binding Update

```
ETHERNET: ETYPE = 0x86DD : Protocol = Unknown
IP6: Dest Opts; Auth SPI = 0x40a; Dest Opts; Len = 0
    IP6: Version = 6 (0x6)
    IP6: Traffic Class = 0 (0x0)
    IP6: Flow Label = 0 (0x0)
    IP6: Payload Length = 64 (0x40)
    IP6: Next Header = 60 (Destination Options Header)
    IP6: Hop Limit = 255 (0xFF)
    IP6: Source Address = 3ffe:3103:0:10:260:8ff:fe46:46e7
    IP6: Destination Address = 3ffe:3103:0:8::3
    IP6: Destination Options Header
        IP6: Next Header = 51 (Authentication Header)
        IP6: Length = 2 (0x2)
        IP6: Padding (4 bytes)
            IP6: Type = 1 (PadN)
                IP6: 00...... = Skip option if not recognized
                IP6: ..0..... = Option data does not change enroute
            IP6: Length = 2 (0x2)
            IP6: Data = 00 00
        IP6: Home Address Option
            IP6: Type = 201
                IP6: 11...... = Discard packet if not recognized, and send ICMP if not multicast
                IP6: ..0..... = Option data does not change enroute
            IP6: Length = 16 (0x10)
            IP6: Home Address = 3ffe:3103:0:8:260:8ff:fe46:46e7
    IP6: Authentication Header
        IP6: Next Header = 60 (Destination Options Header)
        IP6: Length = 4 (0x4)
        IP6: Reserved
        IP6: SPI = 1034 (0x40A)
        IP6: Seq Num = 4 (0x4)
        IP6: Auth Data = 16 12 37 89 41 0B 1C 88 00 8A 81 DC
    IP6: Destination Options Header
        IP6: Next Header = 59 (None)
        IP6: Length = 1 (0x1)
        IP6: Binding Update Option
            IP6: Type = 198
                IP6: 11...... = Discard packet if not recognized, and send ICMP if not multicast
                IP6: ..0..... = Option data does not change enroute
            IP6: Length = 8 (0x8)
            IP6: Flags
                IP6: 1....... = Request a binding acknowledgement response.
                IP6: .1...... = Request host to act as home agent.
                IP6: ..1..... = Binding update includes a care-of address.
            IP6: Prefix Length = 0 (0x0)
            IP6: Sequence Number = 6 (0x6)
            IP6: Lifetime = 10000 (0x2710)
*           IP6: Care-of Address = 102:0:15c1:5c00::a600:0
```

* The line Care-of Address is a wrong value because the sniffer just understands draft13.

(4) - MN1 Binding Update

```
ETHERNET: ETYPE = 0x86DD : Protocol = Unknown
IP6: Dest Opts; Auth SPI = 0x40a; Dest Opts; Len = 0
    IP6: Version = 6 (0x6)
    IP6: Traffic Class = 0 (0x0)
    IP6: Flow Label = 0 (0x0)
    IP6: Payload Length = 64 (0x40)
    IP6: Next Header = 60 (Destination Options Header)
    IP6: Hop Limit = 255 (0xFF)
    IP6: Source Address = 3ffe:3103:0:9:201:2ff:fe2a:93b9
    IP6: Destination Address = 3ffe:3103:0:8::3
    IP6: Destination Options Header
        IP6: Next Header = 51 (Authentication Header)
        IP6: Length = 2 (0x2)
        IP6: Padding (4 bytes)
            IP6: Type = 1 (PadN)
                IP6: 00...... = Skip option if not recognized
                IP6: ..0..... = Option data does not change enroute
            IP6: Length = 2 (0x2)
            IP6: Data = 00 00
        IP6: Home Address Option
            IP6: Type = 201
                IP6: 11...... = Discard packet if not recognized, and send ICMP if not multicast
                IP6: ..0..... = Option data does not change enroute
            IP6: Length = 16 (0x10)
            IP6: Home Address = 3ffe:3103:0:8:201:2ff:fe2a:93b9
    IP6: Authentication Header
        IP6: Next Header = 60 (Destination Options Header)
        IP6: Length = 4 (0x4)
        IP6: Reserved
        IP6: SPI = 1034 (0x40A)
        IP6: Seq Num = 4 (0x4)
        IP6: Auth Data = 16 12 37 89 41 0B 1C 88 00 8A 81 DC
    IP6: Destination Options Header
        IP6: Next Header = 59 (None)
        IP6: Length = 1 (0x1)
        IP6: Binding Update Option
            IP6: Type = 198
                IP6: 11...... = Discard packet if not recognized, and send ICMP if not multicast
                IP6: ..0..... = Option data does not change enroute
            IP6: Length = 8 (0x8)
            IP6: Flags
                IP6: 1....... = Request a binding acknowledgement response.
                IP6: .1...... = Request host to act as home agent.
                IP6: ..1..... = Binding update includes a care-of address.
            IP6: Prefix Length = 0 (0x0)
            IP6: Sequence Number = 6 (0x6)
```

```
            IP6: Lifetime = 10000 (0x2710)
*           IP6: Care-of Address = 102:0:b4c8:1f00::7600:0
```

* The line Care-of Address is a wrong value because the sniffer just understands draft13.

(5) - MN2 Binding Acknowledge

```
ETHERNET: ETYPE = 0x86DD : Protocol = Unknown
IP6: Routing (1 left of 1); Auth SPI = 0x409; Dest Opts; Len = 0
    IP6: Version = 6 (0x6)
    IP6: Traffic Class = 0 (0x0)
    IP6: Flow Label = 0 (0x0)
    IP6: Payload Length = 64 (0x40)
    IP6: Next Header = 43 (Routing Header)
    IP6: Hop Limit = 255 (0xFF)
    IP6: Source Address = 3ffe:3103:0:8::3
    IP6: Destination Address = 3ffe:3103:0:10:260:8ff:fe46:46e7
    IP6: Routing Header
        IP6: Next Header = 51 (Authentication Header)
        IP6: Length = 2 (0x2)
        IP6: Type = 0 (0x0)
        IP6: Segments Left = 1 (0x1)
        IP6: Reserved
        IP6: Route
            IP6: Address = 3ffe:3103:0:8:260:8ff:fe46:46e7
    IP6: Authentication Header
        IP6: Next Header = 60 (Destination Options Header)
        IP6: Length = 4 (0x4)
        IP6: Reserved
        IP6: SPI = 1033 (0x409)
        IP6: Seq Num = 4 (0x4)
        IP6: Auth Data = A3 C5 76 C8 39 0C 5B 55 67 4D 94 D2
    IP6: Destination Options Header
        IP6: Next Header = 59 (None)
        IP6: Length = 1 (0x1)
        IP6: Padding (1 bytes)
            IP6: Type = 0 (Pad1)
                IP6: 00...... = Skip option if not recognized
                IP6: ..0..... = Option data does not change enroute
        IP6: Binding Acknowledgement Option
            IP6: Type = 7
                IP6: 00...... = Skip option if not recognized
                IP6: ..0..... = Option data does not change enroute
            IP6: Length = 11 (0xB)
            IP6: Status: 0 = Binding accepted.
            IP6: Sequence Number = 6 (0x6)
            IP6: Lifetime = 1000 (0x3E8)
            IP6: Refresh = 800 (0x320)
```

(6) - MN1 Binding Acknowledge

```
ETHERNET: ETYPE = 0x86DD : Protocol = Unknown
IP6: Routing (1 left of 1); Auth SPI = 0x409; Dest Opts; Len = 0
    IP6: Version = 6 (0x6)
    IP6: Traffic Class = 0 (0x0)
    IP6: Flow Label = 0 (0x0)
    IP6: Payload Length = 64 (0x40)
    IP6: Next Header = 43 (Routing Header)
    IP6: Hop Limit = 255 (0xFF)
    IP6: Source Address = 3ffe:3103:0:8::3
    IP6: Destination Address = 3ffe:3103:0:9:201:2ff:fe2a:93b9
    IP6: Routing Header
        IP6: Next Header = 51 (Authentication Header)
        IP6: Length = 2 (0x2)
        IP6: Type = 0 (0x0)
        IP6: Segments Left = 1 (0x1)
        IP6: Reserved
        IP6: Route
            IP6: Address = 3ffe:3103:0:8:201:2ff:fe2a:93b9
    IP6: Authentication Header
        IP6: Next Header = 60 (Destination Options Header)
        IP6: Length = 4 (0x4)
        IP6: Reserved
        IP6: SPI = 1033 (0x409)
        IP6: Seq Num = 4 (0x4)
        IP6: Auth Data = A3 C5 76 C8 39 0C 5B 55 67 4D 94 D2
    IP6: Destination Options Header
        IP6: Next Header = 59 (None)
        IP6: Length = 1 (0x1)
        IP6: Padding (1 bytes)
            IP6: Type = 0 (Pad1)
                IP6: 00...... = Skip option if not recognized
                IP6: ..0..... = Option data does not change enroute
        IP6: Binding Acknowledgement Option
            IP6: Type = 7
                IP6: 00...... = Skip option if not recognized
                IP6: ..0..... = Option data does not change enroute
            IP6: Length = 11 (0xB)
            IP6: Status: 0 = Binding accepted.
            IP6: Sequence Number = 6 (0x6)
            IP6: Lifetime = 1000 (0x3E8)
            IP6: Refresh = 800 (0x320)
```

(7) - MN2 - Echo Request

```
ETHERNET: ETYPE = 0x86DD : Protocol = Unknown
IP6: Dest Opts; Proto = ICMP6; Len = 64
    IP6: Version = 6 (0x6)
    IP6: Traffic Class = 0 (0x0)
```

```
    IP6: Flow Label = 0 (0x0)
    IP6: Payload Length = 88 (0x58)
    IP6: Next Header = 60 (Destination Options Header)
    IP6: Hop Limit = 63 (0x3F)
    IP6: Source Address = 3ffe:3103:0:10:260:8ff:fe46:46e7
    IP6: Destination Address = 3ffe:3103:0:8:201:2ff:fe2a:93b9
    IP6: Destination Options Header
        IP6: Next Header = 58 (ICMP6)
        IP6: Length = 2 (0x2)
        IP6: Padding (4 bytes)
            IP6: Type = 1 (PadN)
                IP6: 00...... = Skip option if not recognized
                IP6: ..0..... = Option data does not change enroute
            IP6: Length = 2 (0x2)
            IP6: Data = 00 00
        IP6: Home Address Option
            IP6: Type = 201
                IP6: 11...... = Discard packet if not recognized, and send ICMP if not multicast
                IP6: ..0..... = Option data does not change enroute
            IP6: Length = 16 (0x10)
            IP6: Home Address = 3ffe:3103:0:8:260:8ff:fe46:46e7
    IP6: Payload: Number of data bytes remaining = 64 (0x0040)
ICMP6: Echo Request; ID = 31497, Seq = 0
    ICMP6: Checksum = 0x5501 (ERROR: should be 0x54F9)
    ICMP6: Type = 128 (Echo Request)
    ICMP6: Code = 0 (0x0)
    ICMP6: Identifier = 31497 (0x7B09)
    ICMP6: Sequence Number = 0 (0x0)
    ICMP6: Data: Number of data bytes remaining = 56 (0x0038)
```

(8) - MN2 Echo Request - HA encapsulation

```
ETHERNET: ETYPE = 0x86DD : Protocol = Unknown
IP6: Proto = 0x29; Len = 128
    IP6: Version = 6 (0x6)
    IP6: Traffic Class = 0 (0x0)
    IP6: Flow Label = 0 (0x0)
    IP6: Payload Length = 128 (0x80)
    IP6: Next Header = 41
    IP6: Hop Limit = 255 (0xFF)
    IP6: Source Address = 3ffe:3103:0:8::3
    IP6: Destination Address = 3ffe:3103:0:9:201:2ff:fe2a:93b9
    IP6: Payload: Number of data bytes remaining = 128 (0x0080)
IP6: Dest Opts; Proto = ICMP6; Len = 64
    IP6: Version = 6 (0x6)
    IP6: Traffic Class = 0 (0x0)
    IP6: Flow Label = 0 (0x0)
    IP6: Payload Length = 88 (0x58)
    IP6: Next Header = 60 (Destination Options Header)
    IP6: Hop Limit = 62 (0x3E)
    IP6: Source Address = 3ffe:3103:0:10:260:8ff:fe46:46e7
    IP6: Destination Address = 3ffe:3103:0:8:201:2ff:fe2a:93b9
    IP6: Destination Options Header
        IP6: Next Header = 58 (ICMP6)
        IP6: Length = 2 (0x2)
        IP6: Padding (4 bytes)
            IP6: Type = 1 (PadN)
                IP6: 00...... = Skip option if not recognized
                IP6: ..0..... = Option data does not change enroute
            IP6: Length = 2 (0x2)
            IP6: Data = 00 00
        IP6: Home Address Option
            IP6: Type = 201
                IP6: 11...... = Discard packet if not recognized, and send ICMP if not multicast
                IP6: ..0..... = Option data does not change enroute
            IP6: Length = 16 (0x10)
            IP6: Home Address = 3ffe:3103:0:8:260:8ff:fe46:46e7
    IP6: Payload: Number of data bytes remaining = 64 (0x0040)
ICMP6: Echo Request; ID = 31497, Seq = 0
    ICMP6: Checksum = 0x5501 (ERROR: should be 0x54F9)
    ICMP6: Type = 128 (Echo Request)
    ICMP6: Code = 0 (0x0)
    ICMP6: Identifier = 31497 (0x7B09)
    ICMP6: Sequence Number = 0 (0x0)
    ICMP6: Data: Number of data bytes remaining = 56 (0x0038)
```

(9) - MN1 Echo Reply - BU to MN2

```
ETHERNET: ETYPE = 0x86DD : Protocol = Unknown
IP6: Dest Opts; Auth SPI = 0x409; Dest Opts; Proto = ICMP6; Len = 64
    IP6: Version = 6 (0x6)
    IP6: Traffic Class = 0 (0x0)
    IP6: Flow Label = 0 (0x0)
    IP6: Payload Length = 128 (0x80)
    IP6: Next Header = 60 (Destination Options Header)
    IP6: Hop Limit = 64 (0x40)
    IP6: Source Address = 3ffe:3103:0:9:201:2ff:fe2a:93b9
    IP6: Destination Address = 3ffe:3103:0:8:260:8ff:fe46:46e7
    IP6: Destination Options Header
        IP6: Next Header = 51 (Authentication Header)
        IP6: Length = 2 (0x2)
        IP6: Padding (4 bytes)
            IP6: Type = 1 (PadN)
                IP6: 00...... = Skip option if not recognized
                IP6: ..0..... = Option data does not change enroute
            IP6: Length = 2 (0x2)
            IP6: Data = 00 00
        IP6: Home Address Option
            IP6: Type = 201
                IP6: 11...... = Discard packet if not recognized, and send ICMP if not multicast
```

```
                IP6: ..0..... = Option data does not change enroute
            IP6: Length = 16 (0x10)
            IP6: Home Address = 3ffe:3103:0:8:201:2ff:fe2a:93b9
    IP6: Authentication Header
        IP6: Next Header = 60 (Destination Options Header)
        IP6: Length = 4 (0x4)
        IP6: Reserved
        IP6: SPI = 1033 (0x409)
        IP6: Seq Num = 1 (0x1)
        IP6: Auth Data = 23 4A 57 96 67 1C EE F5 D4 3F 74 14
    IP6: Destination Options Header
        IP6: Next Header = 58 (ICMP6)
        IP6: Length = 1 (0x1)
        IP6: Binding Update Option
            IP6: Type = 198
                IP6: 11...... = Discard packet if not recognized, and send ICMP if not multicast
                IP6: ..0..... = Option data does not change enroute
            IP6: Length = 8 (0x8)
            IP6: Flags
                IP6: ..1..... = Binding update includes a care-of address.
            IP6: Prefix Length = 0 (0x0)
            IP6: Sequence Number = 3 (0x3)
            IP6: Lifetime = 1000 (0x3E8)
*           IP6: Care-of Address = 102:0:b4c8:1f00::7600:0
ICMP6: Echo Reply; ID = 31497, Seq = 0
    ICMP6: Checksum = 0x5401 (ERROR: should be 0x5400)
    ICMP6: Type = 129 (Echo Reply)
    ICMP6: Code = 0 (0x0)
    ICMP6: Identifier = 31497 (0x7B09)
    ICMP6: Sequence Number = 0 (0x0)
    ICMP6: Data: Number of data bytes remaining = 56 (0x0038)
```

* The line Care-of Address is a wrong value because the sniffer just understands draft13.

(10) - MN1 Echo Reply - BU to MN2 - HA encapsulation

```
ETHERNET: ETYPE = 0x86DD : Protocol = Unknown
IP6: Proto = 0x29; Len = 168
    IP6: Version = 6 (0x6)
    IP6: Traffic Class = 0 (0x0)
    IP6: Flow Label = 0 (0x0)
    IP6: Payload Length = 168 (0xA8)
    IP6: Next Header = 41
    IP6: Hop Limit = 255 (0xFF)
    IP6: Source Address = 3ffe:3103:0:8::3
    IP6: Destination Address = 3ffe:3103:0:10:260:8ff:fe46:46e7
    IP6: Payload: Number of data bytes remaining = 168 (0x00A8)
IP6: Dest Opts; Auth SPI = 0x409; Dest Opts; Proto = ICMP6; Len = 64
    IP6: Version = 6 (0x6)
    IP6: Traffic Class = 0 (0x0)
    IP6: Flow Label = 0 (0x0)
    IP6: Payload Length = 128 (0x80)
    IP6: Next Header = 60 (Destination Options Header)
    IP6: Hop Limit = 62 (0x3E)
    IP6: Source Address = 3ffe:3103:0:9:201:2ff:fe2a:93b9
    IP6: Destination Address = 3ffe:3103:0:8:260:8ff:fe46:46e7
    IP6: Destination Options Header
        IP6: Next Header = 51 (Authentication Header)
        IP6: Length = 2 (0x2)
        IP6: Padding (4 bytes)
            IP6: Type = 1 (PadN)
                IP6: 00...... = Skip option if not recognized
                IP6: ..0..... = Option data does not change enroute
            IP6: Length = 2 (0x2)
            IP6: Data = 00 00
        IP6: Home Address Option
            IP6: Type = 201
                IP6: 11...... = Discard packet if not recognized, and send ICMP if not multicast
                IP6: ..0..... = Option data does not change enroute
            IP6: Length = 16 (0x10)
            IP6: Home Address = 3ffe:3103:0:8:201:2ff:fe2a:93b9
    IP6: Authentication Header
        IP6: Next Header = 60 (Destination Options Header)
        IP6: Length = 4 (0x4)
        IP6: Reserved
        IP6: SPI = 1033 (0x409)
        IP6: Seq Num = 1 (0x1)
        IP6: Auth Data = 23 4A 57 96 67 1C EE F5 D4 3F 74 14
    IP6: Destination Options Header
        IP6: Next Header = 58 (ICMP6)
        IP6: Length = 1 (0x1)
        IP6: Binding Update Option
            IP6: Type = 198
                IP6: 11...... = Discard packet if not recognized, and send ICMP if not multicast
                IP6: ..0..... = Option data does not change enroute
            IP6: Length = 8 (0x8)
            IP6: Flags
                IP6: ..1..... = Binding update includes a care-of address.
            IP6: Prefix Length = 0 (0x0)
            IP6: Sequence Number = 3 (0x3)
            IP6: Lifetime = 1000 (0x3E8)
*           IP6: Care-of Address = 102:0:b4c8:1f00::7600:0
ICMP6: Echo Reply; ID = 31497, Seq = 0
    ICMP6: Checksum = 0x5401 (ERROR: should be 0x5400)
    ICMP6: Type = 129 (Echo Reply)
    ICMP6: Code = 0 (0x0)
    ICMP6: Identifier = 31497 (0x7B09)
    ICMP6: Sequence Number = 0 (0x0)
    ICMP6: Data: Number of data bytes remaining = 56 (0x0038)
```

* The line Care-of Address is a wrong value because the sniffer just understands draft13.

(11) - MN2 BU to MN1

```
ETHERNET: ETYPE = 0x86DD : Protocol = Unknown
IP6: Dest Opts; Routing (1 left of 1); Auth SPI = 0x40a; Dest Opts; Len = 0
    IP6: Version = 6 (0x6)
    IP6: Traffic Class = 0 (0x0)
    IP6: Flow Label = 0 (0x0)
    IP6: Payload Length = 88 (0x58)
    IP6: Next Header = 60 (Destination Options Header)
    IP6: Hop Limit = 254 (0xFE)
    IP6: Source Address = 3ffe:3103:0:10:260:8ff:fe46:46e7
    IP6: Destination Address = 3ffe:3103:0:9:201:2ff:fe2a:93b9
    IP6: Destination Options Header
        IP6: Next Header = 43 (Routing Header)
        IP6: Length = 2 (0x2)
        IP6: Padding (4 bytes)
            IP6: Type = 1 (PadN)
                IP6: 00...... = Skip option if not recognized
                IP6: ..0..... = Option data does not change enroute
            IP6: Length = 2 (0x2)
            IP6: Data = 00 00
        IP6: Home Address Option
            IP6: Type = 201
                IP6: 11...... = Discard packet if not recognized, and send ICMP if not multicast
                IP6: ..0..... = Option data does not change enroute
            IP6: Length = 16 (0x10)
            IP6: Home Address = 3ffe:3103:0:8:260:8ff:fe46:46e7
    IP6: Routing Header
        IP6: Next Header = 51 (Authentication Header)
        IP6: Length = 2 (0x2)
        IP6: Type = 0 (0x0)
        IP6: Segments Left = 1 (0x1)
        IP6: Reserved
        IP6: Route
            IP6: Address = 3ffe:3103:0:8:201:2ff:fe2a:93b9
    IP6: Authentication Header
        IP6: Next Header = 60 (Destination Options Header)
        IP6: Length = 4 (0x4)
        IP6: Reserved
        IP6: SPI = 1034 (0x40A)
        IP6: Seq Num = 3 (0x3)
        IP6: Auth Data = F1 F8 AF 6A 56 79 5D A7 4F 92 94 68
    IP6: Destination Options Header
        IP6: Next Header = 59 (None)
        IP6: Length = 1 (0x1)
        IP6: Binding Update Option
            IP6: Type = 198
                IP6: 11...... = Discard packet if not recognized, and send ICMP if not multicast
                IP6: ..0..... = Option data does not change enroute
            IP6: Length = 8 (0x8)
            IP6: Flags
                IP6: ..1..... = Binding update includes a care-of address.
            IP6: Prefix Length = 0 (0x0)
            IP6: Sequence Number = 7 (0x7)
            IP6: Lifetime = 1000 (0x3E8)
*           IP6: Care-of Address = 102:0:15c1:5c00::a600:0
```

* The line Care-of Address is a wrong value because the sniffer just understands draft13.

(12) - Echo Request - MN2 to MN1 directly

```
ETHERNET: ETYPE = 0x86DD : Protocol = Unknown
IP6: Proto = 0x29; Len = 128
    IP6: Version = 6 (0x6)
    IP6: Traffic Class = 0 (0x0)
    IP6: Flow Label = 0 (0x0)
    IP6: Payload Length = 128 (0x80)
    IP6: Next Header = 41
    IP6: Hop Limit = 255 (0xFF)
    IP6: Source Address = 3ffe:3103:0:8::3
    IP6: Destination Address = 3ffe:3103:0:9:201:2ff:fe2a:93b9
    IP6: Payload: Number of data bytes remaining = 128 (0x0080)
IP6: Dest Opts; Proto = ICMP6; Len = 64
    IP6: Version = 6 (0x6)
    IP6: Traffic Class = 0 (0x0)
    IP6: Flow Label = 0 (0x0)
    IP6: Payload Length = 88 (0x58)
    IP6: Next Header = 60 (Destination Options Header)
    IP6: Hop Limit = 62 (0x3E)
    IP6: Source Address = 3ffe:3103:0:10:260:8ff:fe46:46e7
    IP6: Destination Address = 3ffe:3103:0:8:201:2ff:fe2a:93b9
    IP6: Destination Options Header
        IP6: Next Header = 58 (ICMP6)
        IP6: Length = 2 (0x2)
        IP6: Padding (4 bytes)
            IP6: Type = 1 (PadN)
                IP6: 00...... = Skip option if not recognized
                IP6: ..0..... = Option data does not change enroute
            IP6: Length = 2 (0x2)
            IP6: Data = 00 00
        IP6: Home Address Option
            IP6: Type = 201
                IP6: 11...... = Discard packet if not recognized, and send ICMP if not multicast
                IP6: ..0..... = Option data does not change enroute
            IP6: Length = 16 (0x10)
            IP6: Home Address = 3ffe:3103:0:8:260:8ff:fe46:46e7
    IP6: Payload: Number of data bytes remaining = 64 (0x0040)
ICMP6: Echo Request; ID = 31497, Seq = 256
    ICMP6: Checksum = 0x860A (ERROR: should be 0x8602)
```

```
    ICMP6: Type = 128 (Echo Request)
    ICMP6: Code = 0 (0x0)
    ICMP6: Identifier = 31497 (0x7B09)
    ICMP6: Sequence Number = 256 (0x100)
    ICMP6: Data: Number of data bytes remaining = 56 (0x0038)
```

(13) - Echo Reply - MN1 to MN2 directly

```
ETHERNET: ETYPE = 0x86DD : Protocol = Unknown
IP6: Dest Opts; Routing (1 left of 1); Proto = ICMP6; Len = 64
    IP6: Version = 6 (0x6)
    IP6: Traffic Class = 0 (0x0)
    IP6: Flow Label = 0 (0x0)
    IP6: Payload Length = 112 (0x70)
    IP6: Next Header = 60 (Destination Options Header)
    IP6: Hop Limit = 64 (0x40)
    IP6: Source Address = 3ffe:3103:0:9:201:2ff:fe2a:93b9
    IP6: Destination Address = 3ffe:3103:0:10:260:8ff:fe46:46e7
    IP6: Destination Options Header
        IP6: Next Header = 43 (Routing Header)
        IP6: Length = 2 (0x2)
        IP6: Padding (4 bytes)
            IP6: Type = 1 (PadN)
                IP6: 00...... = Skip option if not recognized
                IP6: ..0..... = Option data does not change enroute
            IP6: Length = 2 (0x2)
            IP6: Data = 00 00
        IP6: Home Address Option
            IP6: Type = 201
                IP6: 11...... = Discard packet if not recognized, and send ICMP if not multicast
                IP6: ..0..... = Option data does not change enroute
            IP6: Length = 16 (0x10)
            IP6: Home Address = 3ffe:3103:0:8:201:2ff:fe2a:93b9
                                                                                                    IP6: Routing Header
        IP6: Next Header = 58 (ICMP6)
        IP6: Length = 2 (0x2)
        IP6: Type = 0 (0x0)
        IP6: Segments Left = 1 (0x1)
        IP6: Reserved
        IP6: Route
            IP6: Address = 3ffe:3103:0:8:260:8ff:fe46:46e7
    IP6: Payload: Number of data bytes remaining = 64 (0x0040)
ICMP6: Echo Reply; ID = 31497, Seq = 768
    ICMP6: Checksum = 0x810A (ERROR: should be 0x8101)
    ICMP6: Type = 129 (Echo Reply)
    ICMP6: Code = 0 (0x0)
    ICMP6: Identifier = 31497 (0x7B09)
    ICMP6: Sequence Number = 768 (0x300)
    ICMP6: Data: Number of data bytes remaining = 56 (0x0038)
```

6.2.4.2 WLANs testing

In these tests, we use both, the wired LAN technology and the wireless WLAN technology in order to attach the nodes to the network. The WLAN technology will be always used to perform the handovers in the MN, in order to establish a comparison between the previous tests.

The way the handover is made is using the command

iwconfig ethX essid NAMEWLAN key XXXX-XXXX-XX ap YY:YY:YY:YY:YY

where:

- *ethX* is the wireless interface;
- *essid* is the network name identification (SSID);
- *key* is the WepKey encrypt the data to be sent; and
- *ap* is the Access Point MAC address of the AP.

When the handovers are performed, the MN is covered by the radio signal of both AP (Access Points) of each links. So, the command will attach and detach to one or another network depending on the parameters used.

6.2.4.2.1 Scenario 1

In the first scenario, the MN is located in its HN and the CN is located in another network (Correspondent Node Network - CNN). The Figure 6-10 shows graphically this scenario.

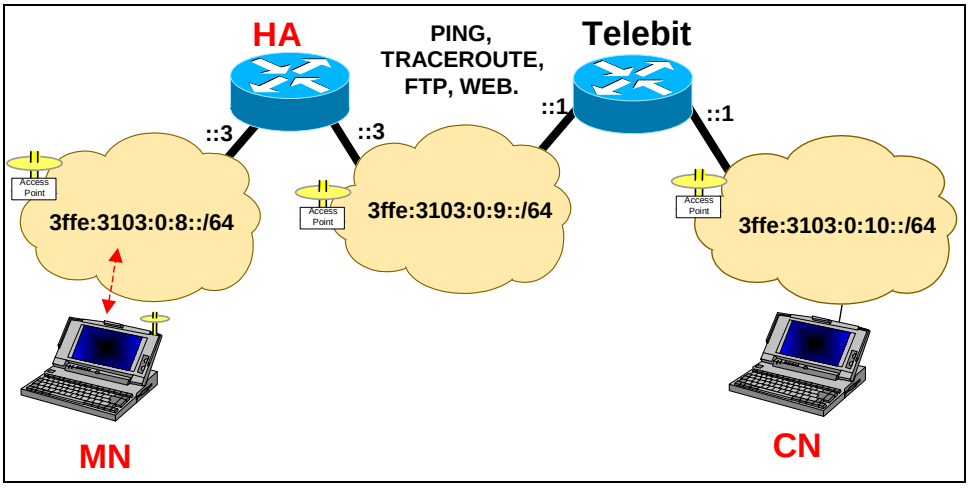


Figure 6-10: WLAN: Scenario 1: MN in the HN. CN is located in other network.

Characteristics of equipment involved:

Equipment Characteristics	HA	MN	CN	FN Router
Platform	*i386*	*i386*	*i386*	*Telebit*
Model	*"Blank" PC*	*"Blank" PC*	*"Blank" PC*	*AXI 462/3*
CPU	*Pentium II 400 MHz (cache 512 KB)*	*Pentium III 650 MHz (cache 256 KB)*	*Pentium II 266 MHz (cache 512 KB)*	*RISC (TBC 103)*
Memory	*64 RAM (swap 256 MB)*	*128 RAM (swap 256 MB)*	*64 RAM (swap 256 MB)*	*16 MB*
Network Interfaces Card	*2 Ethernet cards 3COM series 59x (driver 3c59x)*	*1 Wireless card PCI (802.11) 350 series AIR PCI 352*	*1 Ethernet card 3COM series 59x (driver 3c59x)*	*2 Ethernet cards (TCB 120) Motorola Power PC 750*
Operating System / Kernel	*Linux 7.2 (kernel 2.4.16)*	*Linux 7.2 (kernel 2.4.16)*	*Linux 7.2 (kernel 2.4.16)*	*Telebit OS swtbcpaxos 2.6a14*

** Wireless Access point used: Cisco Access Point 350 SERIES Aironet AIR AP352.*

MN location:

Home Network (HN) - 3ffe:3103:0:9::/64 (via WLAN).

CN location:

Correspondent Node Network (FN) - 3ffe:3103:0:10::/64 (via LAN).

Applications:

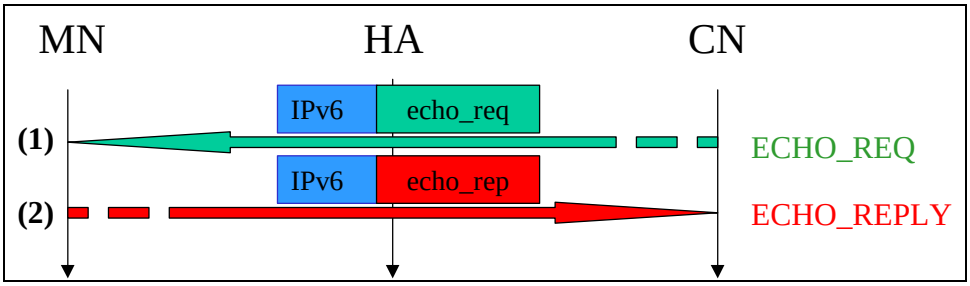
PINGs, FTPs and WWW.

External Behaviour:

In this scenario all packets are processes as an IPv6 standard way. For this reason, all the applications/services works as usually.

Schematic DataFlow:

* Ping from the CN to MN.



Packet Analysis:

A normal IPv6 Echo Request. Equal as if the mobility has not been enabled.

A normal IPv6 Echo Reply.

Sniffed packets:

* The particular packets here presented are from the PING application.

- Echo Request

```
ETHERNET: ETYPE = 0x86DD : Protocol = Unknown
IP6: Proto = ICMP6; Len = 40
    IP6: Version = 6 (0x6)
    IP6: Traffic Class = 0 (0x0)
    IP6: Flow Label = 0 (0x0)
    IP6: Payload Length = 40 (0x28)
    IP6: Next Header = 58 (ICMP6)
    IP6: Hop Limit = 126 (0x7E)
    IP6: Source Address = 3ffe:3103:0:10:260:8ff:fe46:46e7
    IP6: Destination Address = 3ffe:3103:0:8:207:eff:feb9:69fc
    IP6: Payload: Number of data bytes remaining = 40 (0x0028)
ICMP6: Echo Request; ID = 23, Seq = 60579
    ICMP6: Checksum = 0x8208
    ICMP6: Type = 128 (Echo Request)
    ICMP6: Code = 0 (0x0)
    ICMP6: Identifier = 23 (0x17)
    ICMP6: Sequence Number = 60579 (0xECA3)
    ICMP6: Data: Number of data bytes remaining = 32 (0x0020)
```

(2) - Echo Reply

```
ETHERNET: ETYPE = 0x86DD : Protocol = Unknown
IP6: Proto = ICMP6; Len = 40
    IP6: Version = 6 (0x6)
    IP6: Traffic Class = 0 (0x0)
    IP6: Flow Label = 0 (0x0)
    IP6: Payload Length = 40 (0x28)
    IP6: Next Header = 58 (ICMP6)
    IP6: Hop Limit = 64 (0x40)
    IP6: Source Address = 3ffe:3103:0:8:207:eff:feb9:69fc
    IP6: Destination Address = 3ffe:3103:0:10:260:8ff:fe46:46e7
    IP6: Payload: Number of data bytes remaining = 40 (0x0028)
ICMP6: Echo Reply; ID = 23, Seq = 60579
    ICMP6: Checksum = 0x8108
    ICMP6: Type = 129 (Echo Reply)
    ICMP6: Code = 0 (0x0)
    ICMP6: Identifier = 23 (0x17)
    ICMP6: Sequence Number = 60579 (0xECA3)
    ICMP6: Data: Number of data bytes remaining = 32 (0x0020)
```

6.2.4.2.2 Scenario 2

In the second scenario, the MN starts working in a visited network (FN). The CN is located in another network. The Figure 6-11 shows graphically this scenario.

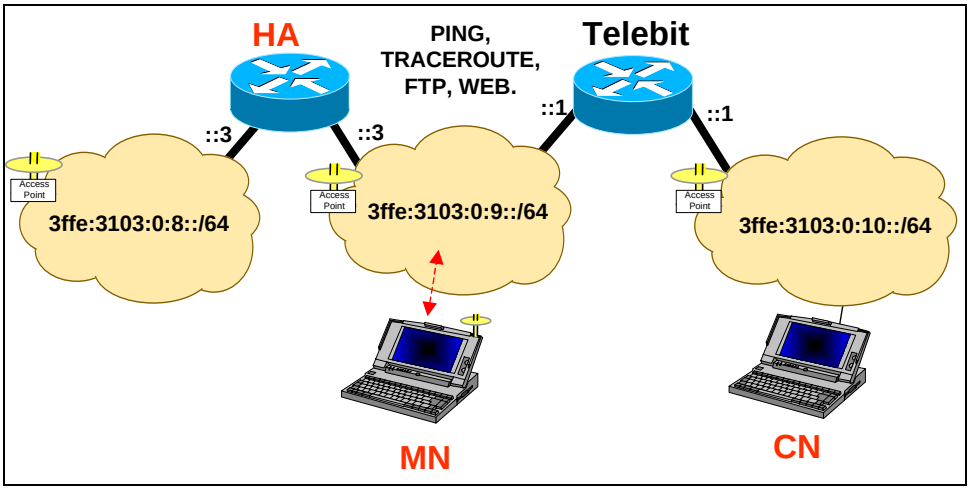


Figure 6-11: WLAN: Scenario 2: MN starting in a FN.

MN location:

Foreign Network (FN) - 3ffe:3103:0:9::/64 (via WLAN).

CN location:

Correspondent Node Network (CNN) - 3ffe:3103:0:10::/64 (via LAN).

Applications:

PINGs, FTPs and WWW.

Characteristics of equipment involved:

Equipment Characteristics	HA	MN	CN	FN Router
Platform	*i386*	*i386*	*i386*	*Telebit*
Model	*"Blank" PC*	*"Blank" PC*	*"Blank" PC*	*AXI 462/3*
CPU	*Pentium II 400 MHz (cache 512 KB)*	*Pentium III 650 MHz (cache 256 KB)*	*Pentium II 266 MHz (cache 512 KB)*	*RISC (TBC 103)*
Memory	*64 RAM (swap 256 MB)*	*128 RAM (swap 256 MB)*	*64 RAM (swap 256 MB)*	*16 MB*
Network Interfaces Card	*2 Ethernet cards 3COM series 59x (driver 3c59x)*	*1 Wireless card PCI (802.11) 350 series AIR PCI 352*	*1 Ethernet card 3COM series 59x (driver 3c59x)*	*2 Ethernet cards (TCB 120) Motorola Power PC 750*
Operating System / Kernel	*Linux 7.2 (kernel 2.4.16)*	*Linux 7.2 (kernel 2.4.16)*	*Linux 7.2 (kernel 2.4.16)*	*Telebit OS Swtbcpaxos 2.6a14*

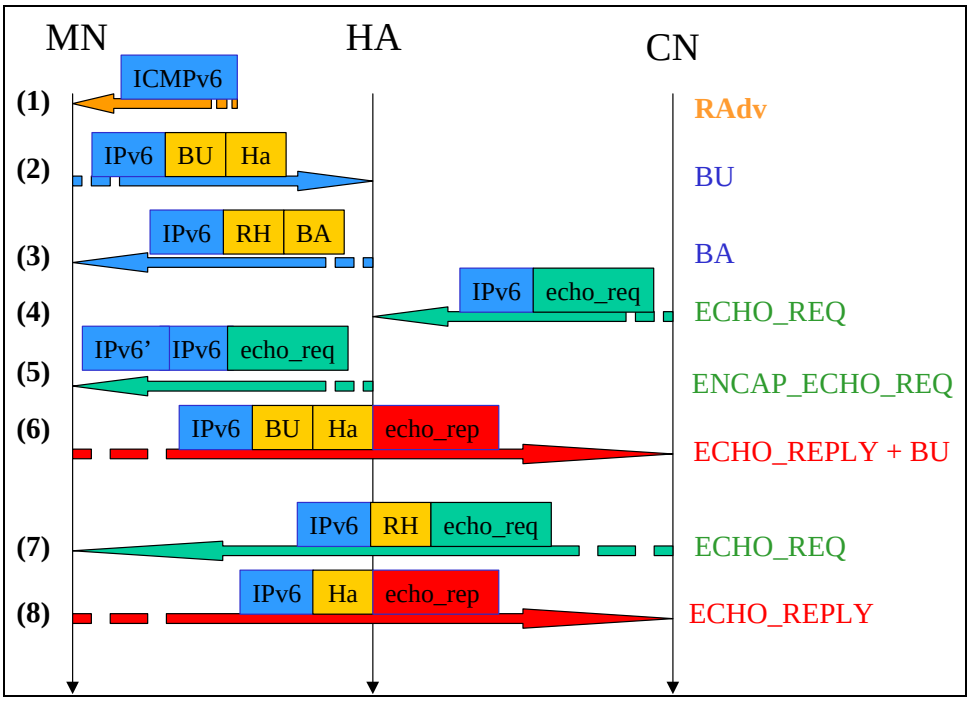
** Wireless Access point used: Cisco Access Point 350 SERIES Aironet AIR AP352.*

External Behaviour:

The MN works with all applications/services transparently as if it was in its HN. When the *iwconfig* command is run to change the nodes as specified in the Figure, the MN is down a time randomly about 0.5 - 2 seconds. After, the applications services work properly.
No any relevant difference is noticed in comparison to the previous scenario, except that sometimes is noticeable an increase of the round trip times in the ping.
In comparison to the wired version the difference is noticeable. Sometimes, no ICMP packets of the ping are lost and, when are lost just one or two.
Using the other TCP application the difference is already big and, sometimes the handover seems to be seamless.

Schematic DataFlow:

* Ping from the CN to MN.



Packet Analysis:

The Router Advertisements (RAs) are being sent periodically by the Telebit. The prefix announced is the 3ffe:3103:0:9::/64.

A Binding Update (BU) is sent from the MN to the HA, in order to register the CoA acquired, using a Ha Destination Option (DH).

A Binding Acknowledge (BA) from the HA confirms the BU. The HA use a Routing Header (RH) to forward the packet.

The first time that the MN sends an *Echo Request* to the MN, it is sent to the MN Home link.

The packet is there intercepted by the HA and, is encapsulated, using IPv6 over IPv6, and sent to the MN.

When the MN receives a packet encapsulated by his HA, it knows that there is a CN that doesn't know it's actual location. For this reason a BU is sent to the CN to actualize the CN Binding Cache (BC) - for this is used again the Ha DH.

The steps are completed and, the next packets from the CN to the MN are sent directly using a Routing Header (RH).

The packets from the MN to the CN are sent directly using a Home address (Ha) DH.

Sniffed packets:

* The particular packets here presented are from the PING application.

(1) - Router Advertisement

```
ETHERNET: ETYPE = 0x86DD : Protocol = Unknown
IP6: Proto = ICMP6; Len = 56
    IP6: Version = 6 (0x6)
    IP6: Traffic Class = 0 (0x0)
    IP6: Flow Label = 0 (0x0)
    IP6: Payload Length = 56 (0x38)
    IP6: Next Header = 58 (ICMP6)
    IP6: Hop Limit = 255 (0xFF)
    IP6: Source Address = fe80::210:5aff:fe67:55bc
    IP6: Destination Address = ff02::1
    IP6: Payload: Number of data bytes remaining = 56 (0x0038)
ICMP6: Router Advertisement
    ICMP6: Checksum = 0x5AA8
    ICMP6: Type = 134 (Router Advertisement)
    ICMP6: Code = 0 (0x0)
```

```
    ICMP6: Current Hop Limit = 64 (0x40)
    ICMP6: 0....... = Not managed address config
    ICMP6: .0...... = Not other stateful config
    ICMP6: Router Lifetime = 600 (0x258)
    ICMP6: Reachable Time = 0 (0x0)
    ICMP6: Retransmission Timer = 0 (0x0)
    ICMP6: Prefix = 3ffe:3103:0:9::
        ICMP6: Type = 3 (0x3)
        ICMP6: Length = 4 (0x4)
        ICMP6: Prefix Length = 64 (0x40)
        ICMP6: Site Prefix Length = 0 (0x0)
        ICMP6: 1....... = On-link determination allowed
        ICMP6: .1...... = Autonomous address configuration
        ICMP6: ..0..... = No router address
        ICMP6: ...0.... = Not a site prefix
        ICMP6: .......0 = Not a route prefix
        ICMP6: Valid Lifetime = 2592000 (0x278D00)
        ICMP6: Preferred Lifetime = 604800 (0x93A80)
        ICMP6: Reserved
        ICMP6: Prefix = 3ffe:3103:0:9::
    ICMP6: Source Link-level Address = 00 10 5A 67 55 BC
        ICMP6: Type = 1 (0x1)
        ICMP6: Length = 1 (0x1)
        ICMP6: Source Link-level Address = 00 10 5A 67 55 BC
```

(2) - Binding Update

```
ETHERNET: ETYPE = 0x86DD : Protocol = Unknown
IP6: Dest Opts; Auth SPI = 0x40a; Dest Opts; Len = 0
    IP6: Version = 6 (0x6)
    IP6: Traffic Class = 0 (0x0)
    IP6: Flow Label = 0 (0x0)
    IP6: Payload Length = 64 (0x40)
    IP6: Next Header = 60 (Destination Options Header)
    IP6: Hop Limit = 255 (0xFF)
    IP6: Source Address = 3ffe:3103:0:9:207:eff:feb9:69fc
    IP6: Destination Address = 3ffe:3103:0:8::3
    IP6: Destination Options Header
        IP6: Next Header = 51 (Authentication Header)
        IP6: Length = 2 (0x2)
        IP6: Padding (4 bytes)
            IP6: Type = 1 (PadN)
                IP6: 00...... = Skip option if not recognized
                IP6: ..0..... = Option data does not change enroute
            IP6: Length = 2 (0x2)
            IP6: Data = 00 00
        IP6: Home Address Option
            IP6: Type = 201
                IP6: 11...... = Discard packet if not recognized, and send ICMP if not multicast
                IP6: ..0..... = Option data does not change enroute
            IP6: Length = 16 (0x10)
            IP6: Home Address = 3ffe:3103:0:8:207:eff:feb9:69fc
    IP6: Authentication Header
        IP6: Next Header = 60 (Destination Options Header)
        IP6: Length = 4 (0x4)
        IP6: Reserved
        IP6: SPI = 1034 (0x40A)
        IP6: Seq Num = 4 (0x4)
        IP6: Auth Data = 16 12 37 89 41 0B 1C 88 00 8A 81 DC
    IP6: Destination Options Header
        IP6: Next Header = 59 (None)
        IP6: Length = 1 (0x1)
        IP6: Binding Update Option
            IP6: Type = 198
                IP6: 11...... = Discard packet if not recognized, and send ICMP if not multicast
                IP6: ..0..... = Option data does not change enroute
            IP6: Length = 8 (0x8)
            IP6: Flags
                IP6: 1....... = Request a binding acknowledgement response.
                IP6: .1...... = Request host to act as home agent.
                IP6: ..1..... = Binding update includes a care-of address.
            IP6: Prefix Length = 0 (0x0)
            IP6: Sequence Number = 6 (0x6)
            IP6: Lifetime = 10000 (0x2710)
*           IP6: Care-of Address = 102:0:8100:a13b:fc24:1500:5e67:133d
```

* The line Care-of Address is a wrong value because the sniffer just understands draft13.

(3) - Binding Acknowledge

```
ETHERNET: ETYPE = 0x86DD : Protocol = Unknown
IP6: Routing (1 left of 1); Auth SPI = 0x409; Dest Opts; Len = 0
    IP6: Version = 6 (0x6)
    IP6: Traffic Class = 0 (0x0)
    IP6: Flow Label = 0 (0x0)
    IP6: Payload Length = 64 (0x40)
    IP6: Next Header = 43 (Routing Header)
    IP6: Hop Limit = 255 (0xFF)
    IP6: Source Address = 3ffe:3103:0:8::3
    IP6: Destination Address = 3ffe:3103:0:9:207:eff:feb9:69fc
    IP6: Routing Header
        IP6: Next Header = 51 (Authentication Header)
        IP6: Length = 2 (0x2)
        IP6: Type = 0 (0x0)
        IP6: Segments Left = 1 (0x1)
        IP6: Reserved
        IP6: Route
            IP6: Address = 3ffe:3103:0:8:207:eff:feb9:69fc
    IP6: Authentication Header
        IP6: Next Header = 60 (Destination Options Header)
```

```
        IP6: Length = 4 (0x4)
        IP6: Reserved
        IP6: SPI = 1033 (0x409)
        IP6: Seq Num = 4 (0x4)
        IP6: Auth Data = A3 C5 76 C8 39 0C 5B 55 67 4D 94 D2
    IP6: Destination Options Header
        IP6: Next Header = 59 (None)
        IP6: Length = 1 (0x1)
        IP6: Padding (1 bytes)
            IP6: Type = 0 (Pad1)
                IP6: 00...... = Skip option if not recognized
                IP6: ..0..... = Option data does not change enroute
        IP6: Binding Acknowledgement Option
            IP6: Type = 7
                IP6: 00...... = Skip option if not recognized
                IP6: ..0..... = Option data does not change enroute
            IP6: Length = 11 (0xB)
            IP6: Status: 0 = Binding accepted.
            IP6: Sequence Number = 6 (0x6)
            IP6: Lifetime = 1000 (0x3E8)
            IP6: Refresh = 800 (0x320)
```

(5) - Echo Request - HA encapsulation

```
ETHERNET: ETYPE = 0x86DD : Protocol = Unknown
IP6: Proto = 0x29; Len = 104
    IP6: Version = 6 (0x6)
    IP6: Traffic Class = 0 (0x0)
    IP6: Flow Label = 0 (0x0)
    IP6: Payload Length = 104 (0x68)
    IP6: Next Header = 41
    IP6: Hop Limit = 255 (0xFF)
    IP6: Source Address = 3ffe:3103:0:8::3
    IP6: Destination Address = 3ffe:3103:0:9:207:eff:feb9:69fc
    IP6: Payload: Number of data bytes remaining = 104 (0x0068)
IP6: Proto = ICMP6; Len = 64
    IP6: Version = 6 (0x6)
    IP6: Traffic Class = 0 (0x0)
    IP6: Flow Label = 0 (0x0)
    IP6: Payload Length = 64 (0x40)
    IP6: Next Header = 58 (ICMP6)
    IP6: Hop Limit = 62 (0x3E)
    IP6: Source Address = 3ffe:3103:0:10:260:8ff:fe46:46e7
    IP6: Destination Address = 3ffe:3103:0:8:207:eff:feb9:69fc
    IP6: Payload: Number of data bytes remaining = 64 (0x0040)
ICMP6: Echo Request; ID = 64548, Seq = 5376
    ICMP6: Checksum = 0xA23B
    ICMP6: Type = 128 (Echo Request)
    ICMP6: Code = 0 (0x0)
    ICMP6: Identifier = 64548 (0xFC24)
    ICMP6: Sequence Number = 5376 (0x1500)
    ICMP6: Data: Number of data bytes remaining = 56 (0x0038)
```

(6) - Echo Reply + CN Binding Update

```
ETHERNET: ETYPE = 0x86DD : Protocol = Unknown
IP6: Dest Opts; Auth SPI = 0x409; Dest Opts; Proto = ICMP6; Len = 64
    IP6: Version = 6 (0x6)
    IP6: Traffic Class = 0 (0x0)
    IP6: Flow Label = 0 (0x0)
    IP6: Payload Length = 128 (0x80)
    IP6: Next Header = 60 (Destination Options Header)
    IP6: Hop Limit = 64 (0x40)
    IP6: Source Address = 3ffe:3103:0:9:207:eff:feb9:69fc
    IP6: Destination Address = 3ffe:3103:0:10:260:8ff:fe46:46e7
    IP6: Destination Options Header
        IP6: Next Header = 51 (Authentication Header)
        IP6: Length = 2 (0x2)
        IP6: Padding (4 bytes)
            IP6: Type = 1 (PadN)
                IP6: 00...... = Skip option if not recognized
                IP6: ..0..... = Option data does not change enroute
            IP6: Length = 2 (0x2)
            IP6: Data = 00 00
        IP6: Home Address Option
            IP6: Type = 201
                IP6: 11...... = Discard packet if not recognized, and send ICMP if not multicast
                IP6: ..0..... = Option data does not change enroute
            IP6: Length = 16 (0x10)
            IP6: Home Address = 3ffe:3103:0:8:207:eff:feb9:69fc
    IP6: Authentication Header
        IP6: Next Header = 60 (Destination Options Header)
        IP6: Length = 4 (0x4)
        IP6: Reserved
        IP6: SPI = 1033 (0x409)
        IP6: Seq Num = 1 (0x1)
        IP6: Auth Data = A6 EC A2 38 D7 C7 4A 62 35 65 ED E7
    IP6: Destination Options Header
        IP6: Next Header = 58 (ICMP6)
        IP6: Length = 1 (0x1)
        IP6: Binding Update Option
            IP6: Type = 198
                IP6: 11...... = Discard packet if not recognized, and send ICMP if not multicast
                IP6: ..0..... = Option data does not change enroute
            IP6: Length = 8 (0x8)
            IP6: Flags
                IP6: ..1..... = Binding update includes a care-of address.
            IP6: Prefix Length = 0 (0x0)
            IP6: Sequence Number = 9 (0x9)
            IP6: Lifetime = 1000 (0x3E8)
```

```
*           IP6: Care-of Address = 102:0:8100:a13b:fc24:1500:5e67:133d
ICMP6: Echo Reply; ID = 64548, Seq = 5376
    ICMP6: Checksum = 0xA13B (ERROR: should be 0xA13A)
    ICMP6: Type = 129 (Echo Reply)
    ICMP6: Code = 0 (0x0)
    ICMP6: Identifier = 64548 (0xFC24)
    ICMP6: Sequence Number = 5376 (0x1500)
    ICMP6: Data: Number of data bytes remaining = 56 (0x0038)
```

* The line Care-of Address is a wrong value because the sniffer just understands draft13.

(7) - Echo Request - directly

```
ETHERNET: ETYPE = 0x86DD : Protocol = Unknown
IP6: Routing (1 left of 1); Proto = ICMP6; Len = 64
    IP6: Version = 6 (0x6)
    IP6: Traffic Class = 0 (0x0)
    IP6: Flow Label = 0 (0x0)
    IP6: Payload Length = 88 (0x58)
    IP6: Next Header = 43 (Routing Header)
    IP6: Hop Limit = 63 (0x3F)
    IP6: Source Address = 3ffe:3103:0:10:260:8ff:fe46:46e7
    IP6: Destination Address = 3ffe:3103:0:9:207:eff:feb9:69fc
    IP6: Routing Header
        IP6: Next Header = 58 (ICMP6)
        IP6: Length = 2 (0x2)
        IP6: Type = 0 (0x0)
        IP6: Segments Left = 1 (0x1)
        IP6: Reserved
        IP6: Route
            IP6: Address = 3ffe:3103:0:8:207:eff:feb9:69fc
    IP6: Payload: Number of data bytes remaining = 64 (0x0040)
ICMP6: Echo Request; ID = 64548, Seq = 6144
    ICMP6: Checksum = 0x983B (ERROR: should be 0x983A)
    ICMP6: Type = 128 (Echo Request)
    ICMP6: Code = 0 (0x0)
    ICMP6: Identifier = 64548 (0xFC24)
    ICMP6: Sequence Number = 6144 (0x1800)
    ICMP6: Data: Number of data bytes remaining = 56 (0x0038)
```

(8) - Echo Reply - directly

```
ETHERNET: ETYPE = 0x86DD : Protocol = Unknown
IP6: Dest Opts; Proto = ICMP6; Len = 64
    IP6: Version = 6 (0x6)
    IP6: Traffic Class = 0 (0x0)
    IP6: Flow Label = 0 (0x0)
    IP6: Payload Length = 88 (0x58)
    IP6: Next Header = 60 (Destination Options Header)
    IP6: Hop Limit = 64 (0x40)
    IP6: Source Address = 3ffe:3103:0:9:207:eff:feb9:69fc
    IP6: Destination Address = 3ffe:3103:0:10:260:8ff:fe46:46e7
    IP6: Destination Options Header
        IP6: Next Header = 58 (ICMP6)
        IP6: Length = 2 (0x2)
        IP6: Padding (4 bytes)
            IP6: Type = 1 (PadN)
                IP6: 00...... = Skip option if not recognized
                IP6: ..0..... = Option data does not change enroute
            IP6: Length = 2 (0x2)
            IP6: Data = 00 00
        IP6: Home Address Option
            IP6: Type = 201
                IP6: 11...... = Discard packet if not recognized, and send ICMP if not multicast
                IP6: ..0..... = Option data does not change enroute
            IP6: Length = 16 (0x10)
            IP6: Home Address = 3ffe:3103:0:8:207:eff:feb9:69fc
    IP6: Payload: Number of data bytes remaining = 64 (0x0040)
ICMP6: Echo Reply; ID = 64548, Seq = 6144
    ICMP6: Checksum = 0x973B (ERROR: should be 0x973A)
    ICMP6: Type = 129 (Echo Reply)
    ICMP6: Code = 0 (0x0)
    ICMP6: Identifier = 64548 (0xFC24)
    ICMP6: Sequence Number = 6144 (0x1800)
    ICMP6: Data: Number of data bytes remaining = 56 (0x0038)
```

6.2.4.2.3 Scenario 3

In the scenario number 3, when the MN starts is located in its HN (1). Then, it moves to a FN (2) and, finally, it backs again to the HN (3). The Figure 6-12 shows graphically this scenario.

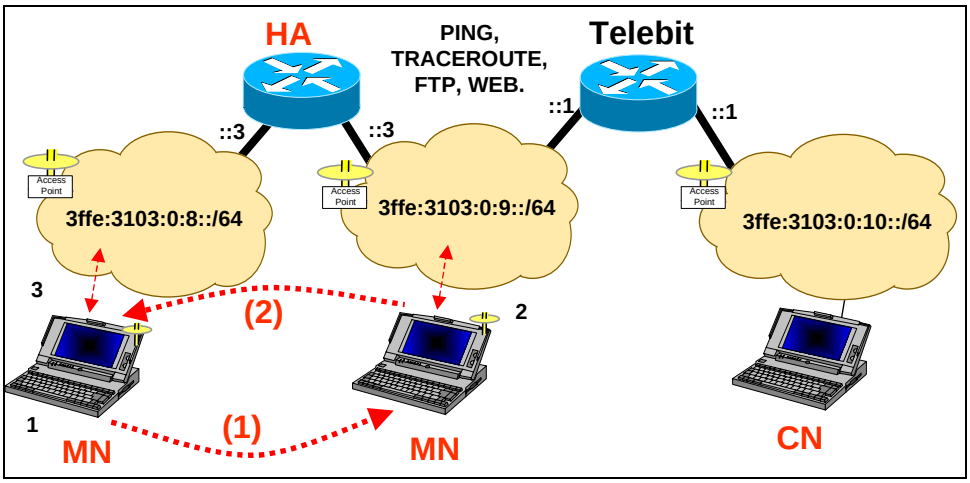


Figure 6-12: WLAN: Scenario 3: MN starts in its HN, goes to a FN and backs to the HN.

Characteristics of equipment involved:

Equipment Characteristics	HA	MN	CN	FN Router
Platform	*i386*	*i386*	*i386*	*Telebit*
Model	*"Blank" PC*	*"Blank" PC*	*"Blank" PC*	*AXI 462/3*
CPU	*Pentium II 400 MHz (cache 512 KB)*	*Pentium III 650 MHz (cache 256 KB)*	*Pentium II 266 MHz (cache 512 KB)*	*RISC (TBC 103)*
Memory	*64 RAM (swap 256 MB)*	*128 RAM (swap 256 MB)*	*64 RAM (swap 256 MB)*	*16 MB*
Network Interfaces Card	*2 Ethernet cards 3COM series 59x (driver 3c59x)*	*1 Wireless card PCI (802.11) 350 series AIR PCI 352*	*1 Ethernet card 3COM series 59x (driver 3c59x)*	*2 Ethernet cards (TCB 120) Motorola Power PC 750*
Operating System / Kernel	*Linux 7.2 (kernel 2.4.16)*	*Linux 7.2 (kernel 2.4.16)*	*Linux 7.2 (kernel 2.4.16)*	*Telebit OS swtbcpaxos 2.6a14*

** Wireless Access point used: Cisco Access Point 350 SERIES Aironet AIR AP352.*

MN location:

HN (3ffe:3103:0:8::/64) -> FN (3ffe:3103:0:9::/64) -> HN (3ffe:3103:0:8::/64) (via WLAN).

CN location:

Correspondent Node Network (CNN) - 3ffe:3103:0:10::/64 (via LAN).

Applications:

PINGs, FTPs and WWW.

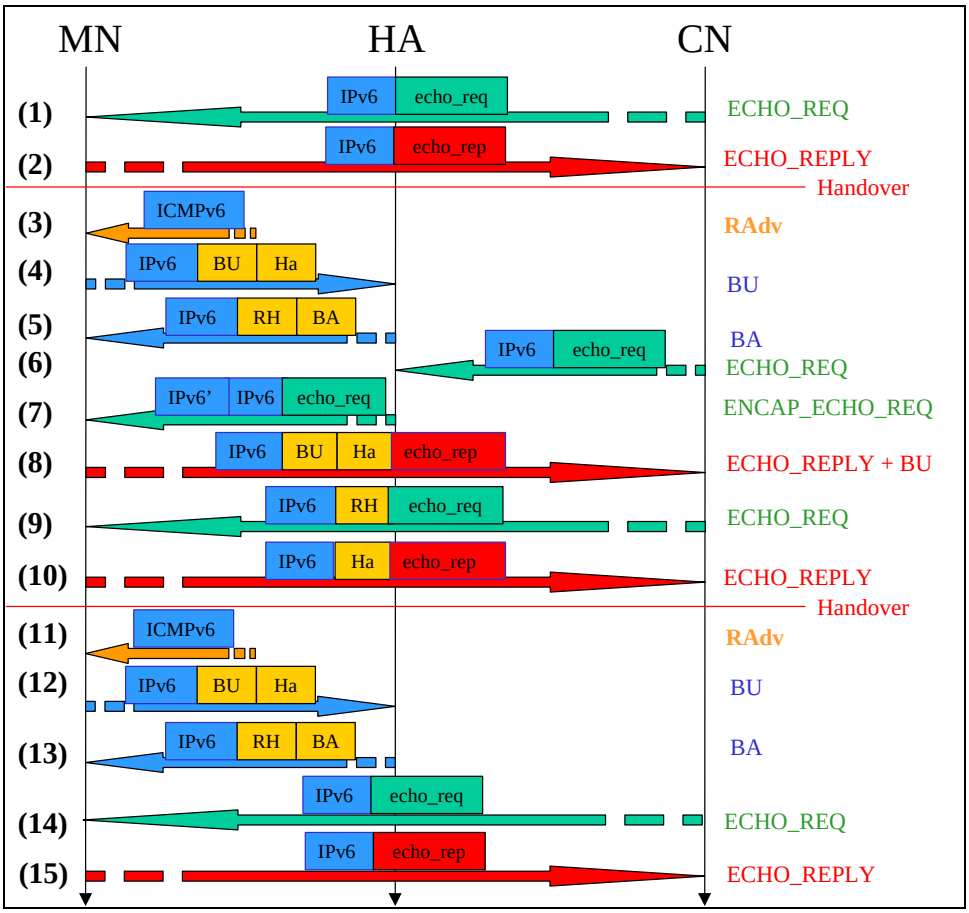
External Behaviour:

When the MN starts in the HN it works as usually.
After, when it moves to the FN, the network goes down during a time random interval of about 0.5 - 2 seconds.
For the ping application, the time of handover is so small that sometime no packets are lost, and even the packet delay doesn't suffer any peak. Other times one or two packets are lost during the handover time.
When the MN moves back to the HN, the network goes down again during a random period of about 0.5-2 seconds, starting the application without problems. The times of the two handovers are very similar.
For TCP applications (www and ftp), some time the handover is seamless. Other times, is clear that the network goes down during a couple of seconds.

There are some situations, mainly using TCP (www and ftp), when the connection never restarts and its necessary restart the downloading/transference. In other cases, we must restart the mobility operation to make it works correctly again.

Schematic DataFlow:

* Ping from the CN to MN.



Packet Analysis:

The packets "Echo Requests" are exchanged directly.

The packets "Echo Replys" are exchanged directly.

The Router Advertisements (RAs) are being sent periodically by the Telebit. The prefix announced is the 3ffe:3103:0:9::/64.

A Binding Update (BU) is sent from the MN to the HA, in order to register the CoA acquired, using a Ha Destination Option (DH).

A Binding Acknowledge (BA) from the HA confirms the BU. The HA use a Routing Header (RH) to forward the packet.

The first time that the MN sends an *Echo Request* to the MN, it is sent to the MN Home link.

The packet is there intercepted by the HA and, is encapsulated, using IPv6 over IPv6, and sent to the MN.

When the MN receives a packet encapsulated by his HA, it knows that there is a CN that doesn't know it's actual location. For this reason a BU is sent to the CN to actualize the CN Binding Cache (BC) - for this is used again the Ha DH.

The steps are completed and, the next packets from the CN to the MN are sent directly using a Routing Header (RH).

(10) The packets from the MN to the CN are sent directly using a Home address (Ha) DH.
(11) When the MN backs to its HN, it receives the RAs.
(12) Then, it knows that is in its HN and sends a BU to the HA with a CoA equal to its Ha.
(13) The HA confirms the MN and remove its entry in the BC.
(14) And the communication starts to be directly between the CN and the MN, using ordinary IPv6 packets.
(15) As well as for the MN - CN direction.

<u>Sniffed packets</u>:

* The particular packets here presented are from the PING application.

- Echo Request

```
ETHERNET: ETYPE = 0x86DD : Protocol = Unknown
IP6: Proto = ICMP6; Len = 40
    IP6: Version = 6 (0x6)
    IP6: Traffic Class = 0 (0x0)
    IP6: Flow Label = 0 (0x0)
    IP6: Payload Length = 40 (0x28)
    IP6: Next Header = 58 (ICMP6)
    IP6: Hop Limit = 126 (0x7E)
    IP6: Source Address = 3ffe:3103:0:10:260:8ff:fe46:46e7
    IP6: Destination Address = 3ffe:3103:0:8:207:eff:feb9:69fc
    IP6: Payload: Number of data bytes remaining = 40 (0x0028)
ICMP6: Echo Request; ID = 23, Seq = 60579
    ICMP6: Checksum = 0x8208
    ICMP6: Type = 128 (Echo Request)
    ICMP6: Code = 0 (0x0)
    ICMP6: Identifier = 23 (0x17)
    ICMP6: Sequence Number = 60579 (0xECA3)
    ICMP6: Data: Number of data bytes remaining = 32 (0x0020)
```

(2) - Echo Reply

```
ETHERNET: ETYPE = 0x86DD : Protocol = Unknown
IP6: Proto = ICMP6; Len = 40
    IP6: Version = 6 (0x6)
    IP6: Traffic Class = 0 (0x0)
    IP6: Flow Label = 0 (0x0)
    IP6: Payload Length = 40 (0x28)
    IP6: Next Header = 58 (ICMP6)
    IP6: Hop Limit = 64 (0x40)
    IP6: Source Address = 3ffe:3103:0:8:207:eff:feb9:69fc
    IP6: Destination Address = 3ffe:3103:0:10:260:8ff:fe46:46e7
    IP6: Payload: Number of data bytes remaining = 40 (0x0028)
ICMP6: Echo Reply; ID = 23, Seq = 60579
    ICMP6: Checksum = 0x8108
    ICMP6: Type = 129 (Echo Reply)
    ICMP6: Code = 0 (0x0)
    ICMP6: Identifier = 23 (0x17)
    ICMP6: Sequence Number = 60579 (0xECA3)
    ICMP6: Data: Number of data bytes remaining = 32 (0x0020)
```

(3) - Router Advertisement

```
ETHERNET: ETYPE = 0x86DD : Protocol = Unknown
IP6: Proto = ICMP6; Len = 56
    IP6: Version = 6 (0x6)
    IP6: Traffic Class = 0 (0x0)
    IP6: Flow Label = 0 (0x0)
    IP6: Payload Length = 56 (0x38)
    IP6: Next Header = 58 (ICMP6)
    IP6: Hop Limit = 255 (0xFF)
    IP6: Source Address = fe80::210:5aff:fe67:55bc
    IP6: Destination Address = ff02::1
    IP6: Payload: Number of data bytes remaining = 56 (0x0038)
ICMP6: Router Advertisement
    ICMP6: Checksum = 0x5AA8
    ICMP6: Type = 134 (Router Advertisement)
    ICMP6: Code = 0 (0x0)
    ICMP6: Current Hop Limit = 64 (0x40)
    ICMP6: 0....... = Not managed address config
    ICMP6: .0...... = Not other stateful config
    ICMP6: Router Lifetime = 600 (0x258)
    ICMP6: Reachable Time = 0 (0x0)
    ICMP6: Retransmission Timer = 0 (0x0)
    ICMP6: Prefix = 3ffe:3103:0:9::
        ICMP6: Type = 3 (0x3)
        ICMP6: Length = 4 (0x4)
        ICMP6: Prefix Length = 64 (0x40)
```

```
        ICMP6: Site Prefix Length = 0 (0x0)
        ICMP6: 1....... = On-link determination allowed
        ICMP6: .1...... = Autonomous address configuration
        ICMP6: ..0..... = No router address
        ICMP6: ...0.... = Not a site prefix
        ICMP6: .......0 = Not a route prefix
        ICMP6: Valid Lifetime = 2592000 (0x278D00)
        ICMP6: Preferred Lifetime = 604800 (0x93A80)
        ICMP6: Reserved
        ICMP6: Prefix = 3ffe:3103:0:9::
    ICMP6: Source Link-level Address = 00 10 5A 67 55 BC
        ICMP6: Type = 1 (0x1)
        ICMP6: Length = 1 (0x1)
        ICMP6: Source Link-level Address = 00 10 5A 67 55 BC
```

(4) - Binding Update

```
ETHERNET: ETYPE = 0x86DD : Protocol = Unknown
IP6: Dest Opts; Auth SPI = 0x40a; Dest Opts; Len = 0
    IP6: Version = 6 (0x6)
    IP6: Traffic Class = 0 (0x0)
    IP6: Flow Label = 0 (0x0)
    IP6: Payload Length = 64 (0x40)
    IP6: Next Header = 60 (Destination Options Header)
    IP6: Hop Limit = 255 (0xFF)
    IP6: Source Address = 3ffe:3103:0:9:207:eff:feb9:69fc
    IP6: Destination Address = 3ffe:3103:0:8::3
    IP6: Destination Options Header
        IP6: Next Header = 51 (Authentication Header)
        IP6: Length = 2 (0x2)
        IP6: Padding (4 bytes)
            IP6: Type = 1 (PadN)
                IP6: 00...... = Skip option if not recognized
                IP6: ..0..... = Option data does not change enroute
            IP6: Length = 2 (0x2)
            IP6: Data = 00 00
        IP6: Home Address Option
            IP6: Type = 201
                IP6: 11...... = Discard packet if not recognized, and send ICMP if not multicast
                IP6: ..0..... = Option data does not change enroute
            IP6: Length = 16 (0x10)
            IP6: Home Address = 3ffe:3103:0:8:207:eff:feb9:69fc
    IP6: Authentication Header
        IP6: Next Header = 60 (Destination Options Header)
        IP6: Length = 4 (0x4)
        IP6: Reserved
        IP6: SPI = 1034 (0x40A)
        IP6: Seq Num = 4 (0x4)
        IP6: Auth Data = 16 12 37 89 41 0B 1C 88 00 8A 81 DC
    IP6: Destination Options Header
        IP6: Next Header = 59 (None)
        IP6: Length = 1 (0x1)
        IP6: Binding Update Option
            IP6: Type = 198
                IP6: 11...... = Discard packet if not recognized, and send ICMP if not multicast
                IP6: ..0..... = Option data does not change enroute
            IP6: Length = 8 (0x8)
            IP6: Flags
                IP6: 1....... = Request a binding acknowledgement response.
                IP6: .1...... = Request host to act as home agent.
                IP6: ..1..... = Binding update includes a care-of address.
            IP6: Prefix Length = 0 (0x0)
            IP6: Sequence Number = 6 (0x6)
            IP6: Lifetime = 10000 (0x2710)
*           IP6: Care-of Address = 102:0:8100:a13b:fc24:1500:5e67:133d
```

* The line Care-of Address is a wrong value because the sniffer just understands draft13.

(5) - Binding Acknowledge

```
ETHERNET: ETYPE = 0x86DD : Protocol = Unknown
IP6: Routing (1 left of 1); Auth SPI = 0x409; Dest Opts; Len = 0
    IP6: Version = 6 (0x6)
    IP6: Traffic Class = 0 (0x0)
    IP6: Flow Label = 0 (0x0)
    IP6: Payload Length = 64 (0x40)
    IP6: Next Header = 43 (Routing Header)
    IP6: Hop Limit = 255 (0xFF)
    IP6: Source Address = 3ffe:3103:0:8::3
    IP6: Destination Address = 3ffe:3103:0:9:207:eff:feb9:69fc
    IP6: Routing Header
        IP6: Next Header = 51 (Authentication Header)
        IP6: Length = 2 (0x2)
        IP6: Type = 0 (0x0)
        IP6: Segments Left = 1 (0x1)
        IP6: Reserved
        IP6: Route
            IP6: Address = 3ffe:3103:0:8:207:eff:feb9:69fc
    IP6: Authentication Header
        IP6: Next Header = 60 (Destination Options Header)
        IP6: Length = 4 (0x4)
        IP6: Reserved
        IP6: SPI = 1033 (0x409)
        IP6: Seq Num = 4 (0x4)
        IP6: Auth Data = A3 C5 76 C8 39 0C 5B 55 67 4D 94 D2
    IP6: Destination Options Header
        IP6: Next Header = 59 (None)
        IP6: Length = 1 (0x1)
        IP6: Padding (1 bytes)
            IP6: Type = 0 (Pad1)
```

```
                IP6: 00...... = Skip option if not recognized
                IP6: ..0..... = Option data does not change enroute
        IP6: Binding Acknowledgement Option
            IP6: Type = 7
                IP6: 00...... = Skip option if not recognized
                IP6: ..0..... = Option data does not change enroute
            IP6: Length = 11 (0xB)
            IP6: Status: 0 = Binding accepted.
            IP6: Sequence Number = 6 (0x6)
            IP6: Lifetime = 1000 (0x3E8)
            IP6: Refresh = 800 (0x320)
```

(7) - Echo Request - HA encapsulation

```
ETHERNET: ETYPE = 0x86DD : Protocol = Unknown
IP6: Proto = 0x29; Len = 104
    IP6: Version = 6 (0x6)
    IP6: Traffic Class = 0 (0x0)
    IP6: Flow Label = 0 (0x0)
    IP6: Payload Length = 104 (0x68)
    IP6: Next Header = 41
    IP6: Hop Limit = 255 (0xFF)
    IP6: Source Address = 3ffe:3103:0:8::3
    IP6: Destination Address = 3ffe:3103:0:9:207:eff:feb9:69fc
    IP6: Payload: Number of data bytes remaining = 104 (0x0068)
IP6: Proto = ICMP6; Len = 64
    IP6: Version = 6 (0x6)
    IP6: Traffic Class = 0 (0x0)
    IP6: Flow Label = 0 (0x0)
    IP6: Payload Length = 64 (0x40)
    IP6: Next Header = 58 (ICMP6)
    IP6: Hop Limit = 62 (0x3E)
    IP6: Source Address = 3ffe:3103:0:10:260:8ff:fe46:46e7
    IP6: Destination Address = 3ffe:3103:0:8:207:eff:feb9:69fc
    IP6: Payload: Number of data bytes remaining = 64 (0x0040)
ICMP6: Echo Request; ID = 64548, Seq = 5376
    ICMP6: Checksum = 0xA23B
    ICMP6: Type = 128 (Echo Request)
    ICMP6: Code = 0 (0x0)
    ICMP6: Identifier = 64548 (0xFC24)
    ICMP6: Sequence Number = 5376 (0x1500)
    ICMP6: Data: Number of data bytes remaining = 56 (0x0038)
```

(8) - Echo Reply + CN Binding Update

```
ETHERNET: ETYPE = 0x86DD : Protocol = Unknown
IP6: Dest Opts; Auth SPI = 0x409; Dest Opts; Proto = ICMP6; Len = 64
    IP6: Version = 6 (0x6)
    IP6: Traffic Class = 0 (0x0)
    IP6: Flow Label = 0 (0x0)
    IP6: Payload Length = 128 (0x80)
    IP6: Next Header = 60 (Destination Options Header)
    IP6: Hop Limit = 64 (0x40)
    IP6: Source Address = 3ffe:3103:0:9:207:eff:feb9:69fc
    IP6: Destination Address = 3ffe:3103:0:10:260:8ff:fe46:46e7
    IP6: Destination Options Header
        IP6: Next Header = 51 (Authentication Header)
        IP6: Length = 2 (0x2)
        IP6: Padding (4 bytes)
            IP6: Type = 1 (PadN)
                IP6: 00...... = Skip option if not recognized
                IP6: ..0..... = Option data does not change enroute
            IP6: Length = 2 (0x2)
            IP6: Data = 00 00
        IP6: Home Address Option
            IP6: Type = 201
                IP6: 11...... = Discard packet if not recognized, and send ICMP if not multicast
                IP6: ..0..... = Option data does not change enroute
            IP6: Length = 16 (0x10)
            IP6: Home Address = 3ffe:3103:0:8:207:eff:feb9:69fc
    IP6: Authentication Header
        IP6: Next Header = 60 (Destination Options Header)
        IP6: Length = 4 (0x4)
        IP6: Reserved
        IP6: SPI = 1033 (0x409)
        IP6: Seq Num = 1 (0x1)
        IP6: Auth Data = A6 EC A2 38 D7 C7 4A 62 35 65 ED E7
    IP6: Destination Options Header
        IP6: Next Header = 58 (ICMP6)
        IP6: Length = 1 (0x1)
        IP6: Binding Update Option
            IP6: Type = 198
                IP6: 11...... = Discard packet if not recognized, and send ICMP if not multicast
                IP6: ..0..... = Option data does not change enroute
            IP6: Length = 8 (0x8)
            IP6: Flags
                IP6: ..1..... = Binding update includes a care-of address.
            IP6: Prefix Length = 0 (0x0)
            IP6: Sequence Number = 9 (0x9)
            IP6: Lifetime = 1000 (0x3E8)
*           IP6: Care-of Address = 102:0:8100:a13b:fc24:1500:5e67:133d
ICMP6: Echo Reply; ID = 64548, Seq = 5376
    ICMP6: Checksum = 0xA13B (ERROR: should be 0xA13A)
    ICMP6: Type = 129 (Echo Reply)
    ICMP6: Code = 0 (0x0)
    ICMP6: Identifier = 64548 (0xFC24)
    ICMP6: Sequence Number = 5376 (0x1500)
    ICMP6: Data: Number of data bytes remaining = 56 (0x0038)
```

* The line Care-of Address is a wrong value because the sniffer just understands draft13.

(9) - Echo Request - directly

```
ETHERNET: ETYPE = 0x86DD : Protocol = Unknown
IP6: Routing (1 left of 1); Proto = ICMP6; Len = 64
    IP6: Version = 6 (0x6)
    IP6: Traffic Class = 0 (0x0)
    IP6: Flow Label = 0 (0x0)
    IP6: Payload Length = 88 (0x58)
    IP6: Next Header = 43 (Routing Header)
    IP6: Hop Limit = 63 (0x3F)
    IP6: Source Address = 3ffe:3103:0:10:260:8ff:fe46:46e7
    IP6: Destination Address = 3ffe:3103:0:9:207:eff:feb9:69fc
    IP6: Routing Header
        IP6: Next Header = 58 (ICMP6)
        IP6: Length = 2 (0x2)
        IP6: Type = 0 (0x0)
        IP6: Segments Left = 1 (0x1)
        IP6: Reserved
        IP6: Route
            IP6: Address = 3ffe:3103:0:8:207:eff:feb9:69fc
    IP6: Payload: Number of data bytes remaining = 64 (0x0040)
ICMP6: Echo Request; ID = 64548, Seq = 6144
    ICMP6: Checksum = 0x983B (ERROR: should be 0x983A)
    ICMP6: Type = 128 (Echo Request)
    ICMP6: Code = 0 (0x0)
    ICMP6: Identifier = 64548 (0xFC24)
    ICMP6: Sequence Number = 6144 (0x1800)
    ICMP6: Data: Number of data bytes remaining = 56 (0x0038)
```

(10) - Echo Reply

```
ETHERNET: ETYPE = 0x86DD : Protocol = Unknown
IP6: Dest Opts; Proto = ICMP6; Len = 64
    IP6: Version = 6 (0x6)
    IP6: Traffic Class = 0 (0x0)
    IP6: Flow Label = 0 (0x0)
    IP6: Payload Length = 88 (0x58)
    IP6: Next Header = 60 (Destination Options Header)
    IP6: Hop Limit = 64 (0x40)
    IP6: Source Address = 3ffe:3103:0:9:207:eff:feb9:69fc
    IP6: Destination Address = 3ffe:3103:0:10:260:8ff:fe46:46e7
    IP6: Destination Options Header
        IP6: Next Header = 58 (ICMP6)
        IP6: Length = 2 (0x2)
        IP6: Padding (4 bytes)
            IP6: Type = 1 (PadN)
                IP6: 00...... = Skip option if not recognized
                IP6: ..0..... = Option data does not change enroute
            IP6: Length = 2 (0x2)
            IP6: Data = 00 00
        IP6: Home Address Option
            IP6: Type = 201
                IP6: 11...... = Discard packet if not recognized, and send ICMP if not multicast
                IP6: ..0..... = Option data does not change enroute
            IP6: Length = 16 (0x10)
            IP6: Home Address = 3ffe:3103:0:8:207:eff:feb9:69fc
    IP6: Payload: Number of data bytes remaining = 64 (0x0040)
ICMP6: Echo Reply; ID = 64548, Seq = 6144
    ICMP6: Checksum = 0x973B (ERROR: should be 0x973A)
    ICMP6: Type = 129 (Echo Reply)
    ICMP6: Code = 0 (0x0)
    ICMP6: Identifier = 64548 (0xFC24)
    ICMP6: Sequence Number = 6144 (0x1800)
    ICMP6: Data: Number of data bytes remaining = 56 (0x0038)
```

(12) - Binding Update

```
ETHERNET: ETYPE = 0x86DD : Protocol = Unknown
IP6: Dest Opts; Auth SPI = 0x40a; Dest Opts; Len = 0
    IP6: Version = 6 (0x6)
    IP6: Traffic Class = 0 (0x0)
    IP6: Flow Label = 0 (0x0)
    IP6: Payload Length = 64 (0x40)
    IP6: Next Header = 60 (Destination Options Header)
    IP6: Hop Limit = 255 (0xFF)
    IP6: Source Address = 3ffe:3103:0:8:207:eff:feb9:69fc
    IP6: Destination Address = 3ffe:3103:0:8::3
    IP6: Destination Options Header
        IP6: Next Header = 51 (Authentication Header)
        IP6: Length = 2 (0x2)
        IP6: Padding (4 bytes)
            IP6: Type = 1 (PadN)
                IP6: 00...... = Skip option if not recognized
                IP6: ..0..... = Option data does not change enroute
            IP6: Length = 2 (0x2)
            IP6: Data = 00 00
        IP6: Home Address Option
            IP6: Type = 201
                IP6: 11...... = Discard packet if not recognized, and send ICMP if not multicast
                IP6: ..0..... = Option data does not change enroute
            IP6: Length = 16 (0x10)
            IP6: Home Address = 3ffe:3103:0:8:207:eff:feb9:69fc
    IP6: Authentication Header
        IP6: Next Header = 60 (Destination Options Header)
        IP6: Length = 4 (0x4)
        IP6: Reserved
        IP6: SPI = 1034 (0x40A)
        IP6: Seq Num = 4 (0x4)
```

```
        IP6: Auth Data = 16 12 37 89 41 0B 1C 88 00 8A 81 DC
    IP6: Destination Options Header
        IP6: Next Header = 59 (None)
        IP6: Length = 1 (0x1)
        IP6: Binding Update Option
            IP6: Type = 198
                IP6: 11...... = Discard packet if not recognized, and send ICMP if not multicast
                IP6: ..0..... = Option data does not change enroute
            IP6: Length = 8 (0x8)
            IP6: Flags
                IP6: 1....... = Request a binding acknowledgement response.
                IP6: .1...... = Request host to act as home agent.
                IP6: ..1..... = Binding update includes a care-of address.
            IP6: Prefix Length = 0 (0x0)
            IP6: Sequence Number = 6 (0x6)
            IP6: Lifetime = 10000 (0x2710)
*           IP6: Care-of Address = 102:0:8100:a13b:fc24:1500:5e67:133d
```

* The line Care-of Address is a wrong value because the sniffer just understands draft13.

(13) - Binding Acknowledge

```
ETHERNET: ETYPE = 0x86DD : Protocol = Unknown
IP6: Routing (1 left of 1); Auth SPI = 0x409; Dest Opts; Len = 0
    IP6: Version = 6 (0x6)
    IP6: Traffic Class = 0 (0x0)
    IP6: Flow Label = 0 (0x0)
    IP6: Payload Length = 64 (0x40)
    IP6: Next Header = 43 (Routing Header)
    IP6: Hop Limit = 255 (0xFF)
    IP6: Source Address = 3ffe:3103:0:8::3
    IP6: Destination Address = 3ffe:3103:0:8:207:eff:feb9:69fc
    IP6: Authentication Header
        IP6: Next Header = 60 (Destination Options Header)
        IP6: Length = 4 (0x4)
        IP6: Reserved
        IP6: SPI = 1033 (0x409)
        IP6: Seq Num = 4 (0x4)
        IP6: Auth Data = A3 C5 76 C8 39 0C 5B 55 67 4D 94 D2
    IP6: Destination Options Header
        IP6: Next Header = 59 (None)
        IP6: Length = 1 (0x1)
        IP6: Padding (1 bytes)
            IP6: Type = 0 (Pad1)
                IP6: 00...... = Skip option if not recognized
                IP6: ..0..... = Option data does not change enroute
        IP6: Binding Acknowledgement Option
            IP6: Type = 7
                IP6: 00...... = Skip option if not recognized
                IP6: ..0..... = Option data does not change enroute
            IP6: Length = 11 (0xB)
            IP6: Status: 0 = Binding accepted.
            IP6: Sequence Number = 6 (0x6)
            IP6: Lifetime = 1000 (0x3E8)
            IP6: Refresh = 800 (0x320)
```

(14) - Echo Request

```
ETHERNET: ETYPE = 0x86DD : Protocol = Unknown
IP6: Proto = ICMP6; Len = 40
    IP6: Version = 6 (0x6)
    IP6: Traffic Class = 0 (0x0)
    IP6: Flow Label = 0 (0x0)
    IP6: Payload Length = 40 (0x28)
    IP6: Next Header = 58 (ICMP6)
    IP6: Hop Limit = 126 (0x7E)
    IP6: Source Address = 3ffe:3103:0:10:260:8ff:fe46:46e7
    IP6: Destination Address = 3ffe:3103:0:8:207:eff:feb9:69fc
    IP6: Payload: Number of data bytes remaining = 40 (0x0028)
ICMP6: Echo Request; ID = 23, Seq = 60579
    ICMP6: Checksum = 0x8208
    ICMP6: Type = 128 (Echo Request)
    ICMP6: Code = 0 (0x0)
    ICMP6: Identifier = 23 (0x17)
    ICMP6: Sequence Number = 60579 (0xECA3)
    ICMP6: Data: Number of data bytes remaining = 32 (0x0020)
```

(15) - Echo Reply

```
ETHERNET: ETYPE = 0x86DD : Protocol = Unknown
IP6: Proto = ICMP6; Len = 40
    IP6: Version = 6 (0x6)
    IP6: Traffic Class = 0 (0x0)
    IP6: Flow Label = 0 (0x0)
    IP6: Payload Length = 40 (0x28)
    IP6: Next Header = 58 (ICMP6)
    IP6: Hop Limit = 64 (0x40)
    IP6: Source Address = 3ffe:3103:0:8:207:eff:feb9:69fc
    IP6: Destination Address = 3ffe:3103:0:10:260:8ff:fe46:46e7
    IP6: Payload: Number of data bytes remaining = 40 (0x0028)
ICMP6: Echo Reply; ID = 23, Seq = 60579
    ICMP6: Checksum = 0x8108
    ICMP6: Type = 129 (Echo Reply)
    ICMP6: Code = 0 (0x0)
    ICMP6: Identifier = 23 (0x17)
    ICMP6: Sequence Number = 60579 (0xECA3)
    ICMP6: Data: Number of data bytes remaining = 32 (0x0020)
```

6.2.4.2.4 Scenario 4

In the scenario number 4, we test the situation when there are two MNs on the network. One is in its HN and, the second, in a FN. The Figure 6-13 shows graphically this scenario.

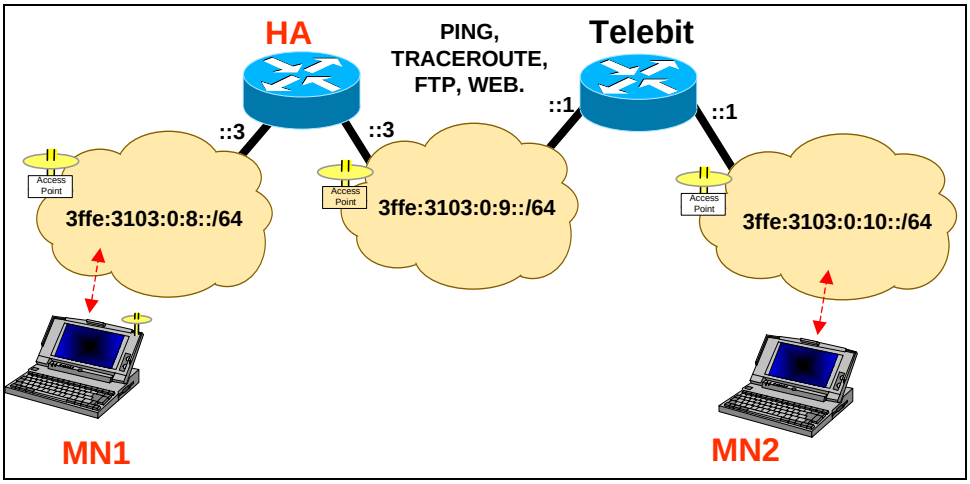


Figure 6-13: WLAN: Scenario 4: The MN1 is on the HN and the MN2 is on a FN.

Characteristics of equipment involved:

Equipment Characteristics	HA	MN1	MN2	FN Router
Platform	*i386*	*i386*	*i386*	*Telebit*
Model	*"Blank" PC*	*"Blank" PC*	*"Blank" PC*	*AXI 462/3*
CPU	*Pentium II 400 MHz (cache 512 KB)*	*Pentium III 650 MHz (cache 256 KB)*	*Pentium II 266 MHz (cache 512 KB)*	*RISC (TBC 103)*
Memory	*64 RAM (swap 256 MB)*	*128 RAM (swap 256 MB)*	*64 RAM (swap 256 MB)*	*16 MB*
Network Interfaces Card	*2 Ethernet cards 3COM series 59x (driver 3c59x)*	*1 Wireless card PCI (802.11) 350 series AIR PCI 352*	*1 Ethernet card 3COM series 59x (driver 3c59x)*	*2 Ethernet cards (TCB 120) Motorola Power PC 750*
Operating System / Kernel	*Linux 7.2 (kernel 2.4.16)*	*Linux 7.2 (kernel 2.4.16)*	*Linux 7.2 (kernel 2.4.16)*	*Telebit OS swtbcpaxos 2.6a14*

** Wireless Access point used: Cisco Access Point 350 SERIES Aironet AIR AP352.*

MN1 location:

Home Network (HN) - 3ffe:3103:0:9::/64 (via WLAN).

MN2 location:

Foreign Network (FN) - 3ffe:3103:0:10::/64 (via LAN).

Applications:

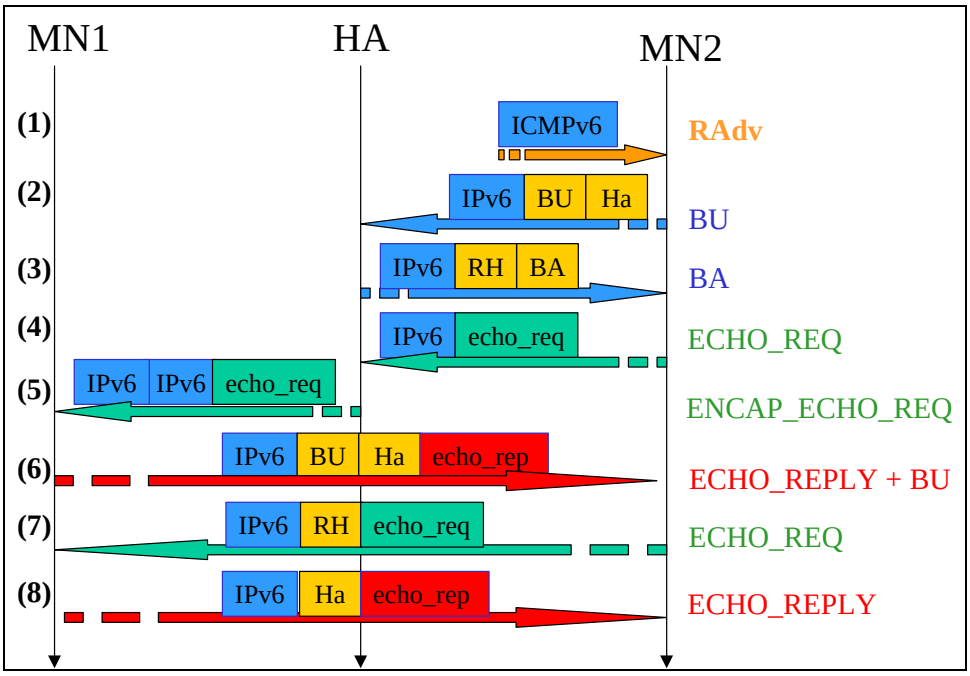
PINGs, FTPs and WWW.

External Behaviour:

Externally it works correctly with all the applications/services tested. After the movement to attach the nodes as specified in the Figure, when the MN is down about 0.5 - 2 seconds, it works properly.
Actually, in terms of functionality, the situation is exactly the same as if the MN1 was a CN.

Schematic DataFlow:

* Ping from the MN2 to MN1.



Packet Analysis:

The Router Advertisements (RAs) are being sent periodically by the Telebit. The prefix announced is the 3ffe:3103:0:10::/64.

A Binding Update (BU) is sent from the MN to the HA, in order to register the CoA acquired, using a Ha Destination Option (DH).

Binding Acknowledge (BA) from the HA confirms the BU. The HA use a Routing Header (RH) to forward the packet.

The first time that the MN sends an *Echo Request* to the MN, it is sent to the MN Home link.

The packet is there intercepted by the HA and, is encapsulated, using IPv6 over IPv6, and sent to the MN.

When the MN receives a packet encapsulated by his HA, it knows that there is a CN that doesn't know it's actual location. For this reason a BU is sent to the CN to actualize the CN Binding Cache (BC) - for this is used again the Ha DH.

The steps are completed and, the next packets from the CN to the MN are sent directly using a Routing Header (RH).

The packets from the MN to the CN are sent directly using a Home address (Ha) DH.

Sniffed packets:

* The particular packets here presented are from the PING application.

(1) - Router Advertisement

```
ETHERNET: ETYPE = 0x86DD : Protocol = Unknown
IP6: Proto = ICMP6; Len = 56
    IP6: Version = 6 (0x6)
    IP6: Traffic Class = 0 (0x0)
    IP6: Flow Label = 0 (0x0)
    IP6: Payload Length = 56 (0x38)
    IP6: Next Header = 58 (ICMP6)
    IP6: Hop Limit = 255 (0xFF)
    IP6: Source Address = fe80::2c0:33ff:fe0c:a0
    IP6: Destination Address = ff02::1
    IP6: Payload: Number of data bytes remaining = 56 (0x0038)
ICMP6: Router Advertisement
    ICMP6: Checksum = 0x5AA8
    ICMP6: Type = 134 (Router Advertisement)
    ICMP6: Code = 0 (0x0)
    ICMP6: Current Hop Limit = 64 (0x40)
    ICMP6: 0....... = Not managed address config
    ICMP6: .0...... = Not other stateful config
    ICMP6: Router Lifetime = 600 (0x258)
```

```
    ICMP6: Reachable Time = 0 (0x0)
    ICMP6: Retransmission Timer = 0 (0x0)
    ICMP6: Prefix = 3ffe:3103:0:10::
        ICMP6: Type = 3 (0x3)
        ICMP6: Length = 4 (0x4)
        ICMP6: Prefix Length = 64 (0x40)
        ICMP6: Site Prefix Length = 0 (0x0)
        ICMP6: 1....... = On-link determination allowed
        ICMP6: .1...... = Autonomous address configuration
        ICMP6: ..0..... = No router address
        ICMP6: ...0.... = Not a site prefix
        ICMP6: .......0 = Not a route prefix
        ICMP6: Valid Lifetime = 2592000 (0x278D00)
        ICMP6: Preferred Lifetime = 604800 (0x93A80)
        ICMP6: Reserved
        ICMP6: Prefix = 3ffe:3103:0:10::
    ICMP6: Source Link-level Address = 00 C0 33 0C 00 10
        ICMP6: Type = 1 (0x1)
        ICMP6: Length = 1 (0x1)
        ICMP6: Source Link-level Address = 00 C0 33 0C 00 10
```

(2) - Binding Update

```
ETHERNET: ETYPE = 0x86DD : Protocol = Unknown
IP6: Dest Opts; Auth SPI = 0x40a; Dest Opts; Len = 0
    IP6: Version = 6 (0x6)
    IP6: Traffic Class = 0 (0x0)
    IP6: Flow Label = 0 (0x0)
    IP6: Payload Length = 64 (0x40)
    IP6: Next Header = 60 (Destination Options Header)
    IP6: Hop Limit = 255 (0xFF)
    IP6: Source Address = 3ffe:3103:0:10:260:8ff:fe46:46e7
    IP6: Destination Address = 3ffe:3103:0:8::3
    IP6: Destination Options Header
        IP6: Next Header = 51 (Authentication Header)
        IP6: Length = 2 (0x2)
        IP6: Padding (4 bytes)
            IP6: Type = 1 (PadN)
                IP6: 00...... = Skip option if not recognized
                IP6: ..0..... = Option data does not change enroute
            IP6: Length = 2 (0x2)
            IP6: Data = 00 00
        IP6: Home Address Option
            IP6: Type = 201
                IP6: 11...... = Discard packet if not recognized, and send ICMP if not multicast
                IP6: ..0..... = Option data does not change enroute
            IP6: Length = 16 (0x10)
            IP6: Home Address = 3ffe:3103:0:8:260:8ff:fe46:46e7
    IP6: Authentication Header
        IP6: Next Header = 60 (Destination Options Header)
        IP6: Length = 4 (0x4)
        IP6: Reserved
        IP6: SPI = 1034 (0x40A)
        IP6: Seq Num = 4 (0x4)
        IP6: Auth Data = 16 12 37 89 41 0B 1C 88 00 8A 81 DC
    IP6: Destination Options Header
        IP6: Next Header = 59 (None)
        IP6: Length = 1 (0x1)
        IP6: Binding Update Option
            IP6: Type = 198
                IP6: 11...... = Discard packet if not recognized, and send ICMP if not multicast
                IP6: ..0..... = Option data does not change enroute
            IP6: Length = 8 (0x8)
            IP6: Flags
                IP6: 1....... = Request a binding acknowledgement response.
                IP6: .1...... = Request host to act as home agent.
                IP6: ..1..... = Binding update includes a care-of address.
            IP6: Prefix Length = 0 (0x0)
            IP6: Sequence Number = 6 (0x6)
            IP6: Lifetime = 10000 (0x2710)
*           IP6: Care-of Address = 102:0:8100:a13b:fc24:1500:5e67:133d
```

* The line Care-of Address is a wrong value because the sniffer just understands draft13.

(3) - Binding Acknowledge

```
ETHERNET: ETYPE = 0x86DD : Protocol = Unknown
IP6: Routing (1 left of 1); Auth SPI = 0x409; Dest Opts; Len = 0
    IP6: Version = 6 (0x6)
    IP6: Traffic Class = 0 (0x0)
    IP6: Flow Label = 0 (0x0)
    IP6: Payload Length = 64 (0x40)
    IP6: Next Header = 43 (Routing Header)
    IP6: Hop Limit = 255 (0xFF)
    IP6: Source Address = 3ffe:3103:0:8::3
    IP6: Destination Address = 3ffe:3103:0:10:260:8ff:fe46:46e7
    IP6: Routing Header
        IP6: Next Header = 51 (Authentication Header)
        IP6: Length = 2 (0x2)
        IP6: Type = 0 (0x0)
        IP6: Segments Left = 1 (0x1)
        IP6: Reserved
        IP6: Route
            IP6: Address = 3ffe:3103:0:8:260:8ff:fe46:46e7
    IP6: Authentication Header
        IP6: Next Header = 60 (Destination Options Header)
        IP6: Length = 4 (0x4)
        IP6: Reserved
        IP6: SPI = 1033 (0x409)
        IP6: Seq Num = 4 (0x4)
```

```
        IP6: Auth Data = A3 C5 76 C8 39 0C 5B 55 67 4D 94 D2
    IP6: Destination Options Header
        IP6: Next Header = 59 (None)
        IP6: Length = 1 (0x1)
        IP6: Padding (1 bytes)
            IP6: Type = 0 (Pad1)
                IP6: 00...... = Skip option if not recognized
                IP6: ..0..... = Option data does not change enroute
        IP6: Binding Acknowledgement Option
            IP6: Type = 7
                IP6: 00...... = Skip option if not recognized
                IP6: ..0..... = Option data does not change enroute
            IP6: Length = 11 (0xB)
            IP6: Status: 0 = Binding accepted.
            IP6: Sequence Number = 6 (0x6)
            IP6: Lifetime = 1000 (0x3E8)
            IP6: Refresh = 800 (0x320)
```

(5) - Echo Request - HA encapsulation

```
ETHERNET: ETYPE = 0x86DD : Protocol = Unknown
IP6: Proto = 0x29; Len = 104
    IP6: Version = 6 (0x6)
    IP6: Traffic Class = 0 (0x0)
    IP6: Flow Label = 0 (0x0)
    IP6: Payload Length = 104 (0x68)
    IP6: Next Header = 41
    IP6: Hop Limit = 255 (0xFF)
    IP6: Source Address = 3ffe:3103:0:8::3
    IP6: Destination Address = 3ffe:3103:0:8:207:eff:feb9:69fc
    IP6: Payload: Number of data bytes remaining = 104 (0x0068)
IP6: Proto = ICMP6; Len = 64
    IP6: Version = 6 (0x6)
    IP6: Traffic Class = 0 (0x0)
    IP6: Flow Label = 0 (0x0)
    IP6: Payload Length = 64 (0x40)
    IP6: Next Header = 58 (ICMP6)
    IP6: Hop Limit = 62 (0x3E)
    IP6: Source Address = 3ffe:3103:0:10:260:8ff:fe46:46e7
    IP6: Destination Address = 3ffe:3103:0:8:207:eff:feb9:69fc
    IP6: Payload: Number of data bytes remaining = 64 (0x0040)
ICMP6: Echo Request; ID = 64548, Seq = 5376
    ICMP6: Checksum = 0xA23B
    ICMP6: Type = 128 (Echo Request)
    ICMP6: Code = 0 (0x0)
    ICMP6: Identifier = 64548 (0xFC24)
    ICMP6: Sequence Number = 5376 (0x1500)
    ICMP6: Data: Number of data bytes remaining = 56 (0x0038)
```

(6) - Echo Reply - MN2 Binding Update

```
ETHERNET: ETYPE = 0x86DD : Protocol = Unknown
IP6: Dest Opts; Auth SPI = 0x409; Dest Opts; Proto = ICMP6; Len = 64
    IP6: Version = 6 (0x6)
    IP6: Traffic Class = 0 (0x0)
    IP6: Flow Label = 0 (0x0)
    IP6: Payload Length = 128 (0x80)
    IP6: Next Header = 60 (Destination Options Header)
    IP6: Hop Limit = 64 (0x40)
    IP6: Source Address = 3ffe:3103:0:9:207:eff:feb9:69fc
    IP6: Destination Address = 3ffe:3103:0:10:260:8ff:fe46:46e7
    IP6: Destination Options Header
        IP6: Next Header = 51 (Authentication Header)
        IP6: Length = 2 (0x2)
        IP6: Padding (4 bytes)
            IP6: Type = 1 (PadN)
                IP6: 00...... = Skip option if not recognized
                IP6: ..0..... = Option data does not change enroute
            IP6: Length = 2 (0x2)
            IP6: Data = 00 00
        IP6: Home Address Option
            IP6: Type = 201
                IP6: 11...... = Discard packet if not recognized, and send ICMP if not multicast
                IP6: ..0..... = Option data does not change enroute
            IP6: Length = 16 (0x10)
            IP6: Home Address = 3ffe:3103:0:8:207:eff:feb9:69fc
    IP6: Authentication Header
        IP6: Next Header = 60 (Destination Options Header)
        IP6: Length = 4 (0x4)
        IP6: Reserved
        IP6: SPI = 1033 (0x409)
        IP6: Seq Num = 1 (0x1)
        IP6: Auth Data = A6 EC A2 38 D7 C7 4A 62 35 65 ED E7
    IP6: Destination Options Header
        IP6: Next Header = 58 (ICMP6)
        IP6: Length = 1 (0x1)
        IP6: Binding Update Option
            IP6: Type = 198
                IP6: 11...... = Discard packet if not recognized, and send ICMP if not multicast
                IP6: ..0..... = Option data does not change enroute
            IP6: Length = 8 (0x8)
            IP6: Flags
                IP6: ..1..... = Binding update includes a care-of address.
            IP6: Prefix Length = 0 (0x0)
            IP6: Sequence Number = 9 (0x9)
            IP6: Lifetime = 1000 (0x3E8)
*           IP6: Care-of Address = 102:0:8100:a13b:fc24:1500:5e67:133d
ICMP6: Echo Reply; ID = 64548, Seq = 5376
    ICMP6: Checksum = 0xA13B (ERROR: should be 0xA13A)
    ICMP6: Type = 129 (Echo Reply)
```

```
    ICMP6: Code = 0 (0x0)
    ICMP6: Identifier = 64548 (0xFC24)
    ICMP6: Sequence Number = 5376 (0x1500)
    ICMP6: Data: Number of data bytes remaining = 56 (0x0038)
```

* The line Care-of Address is a wrong value because the sniffer just understands draft13.

(7) - Echo Request

```
ETHERNET: ETYPE = 0x86DD : Protocol = Unknown
IP6: Dest Opts; Proto = ICMP6; Len = 64
    IP6: Version = 6 (0x6)
    IP6: Traffic Class = 0 (0x0)
    IP6: Flow Label = 0 (0x0)
    IP6: Payload Length = 88 (0x58)
    IP6: Next Header = 60 (Destination Options Header)
    IP6: Hop Limit = 64 (0x40)
    IP6: Source Address = 3ffe:3103:0:9:207:eff:feb9:69fc
    IP6: Destination Address = 3ffe:3103:0:10:260:8ff:fe46:46e7
    IP6: Destination Options Header
        IP6: Next Header = 58 (ICMP6)
        IP6: Length = 2 (0x2)
        IP6: Padding (4 bytes)
            IP6: Type = 1 (PadN)
                IP6: 00...... = Skip option if not recognized
                IP6: ..0..... = Option data does not change enroute
            IP6: Length = 2 (0x2)
            IP6: Data = 00 00
        IP6: Home Address Option
            IP6: Type = 201
                IP6: 11...... = Discard packet if not recognized, and send ICMP if not multicast
                IP6: ..0..... = Option data does not change enroute
            IP6: Length = 16 (0x10)
            IP6: Home Address = 3ffe:3103:0:8:207:eff:feb9:69fc
    IP6: Payload: Number of data bytes remaining = 64 (0x0040)
ICMP6: Echo Request; ID = 64548, Seq = 6144
    ICMP6: Checksum = 0x983B (ERROR: should be 0x983A)
    ICMP6: Type = 128 (Echo Request)
    ICMP6: Code = 0 (0x0)
    ICMP6: Identifier = 64548 (0xFC24)
    ICMP6: Sequence Number = 6144 (0x1800)
    ICMP6: Data: Number of data bytes remaining = 56 (0x0038)
```

(8) - Echo Reply

```
ETHERNET: ETYPE = 0x86DD : Protocol = Unknown
IP6: Routing (1 left of 1); Proto = ICMP6; Len = 64
    IP6: Version = 6 (0x6)
    IP6: Traffic Class = 0 (0x0)
    IP6: Flow Label = 0 (0x0)
    IP6: Payload Length = 88 (0x58)
    IP6: Next Header = 43 (Routing Header)
    IP6: Hop Limit = 63 (0x3F)
    IP6: Source Address = 3ffe:3103:0:10:260:8ff:fe46:46e7
    IP6: Destination Address = 3ffe:3103:0:9:207:eff:feb9:69fc
    IP6: Routing Header
        IP6: Next Header = 58 (ICMP6)
        IP6: Length = 2 (0x2)
        IP6: Type = 0 (0x0)
        IP6: Segments Left = 1 (0x1)
        IP6: Reserved
        IP6: Route
            IP6: Address = 3ffe:3103:0:8:207:eff:feb9:69fc
    IP6: Payload: Number of data bytes remaining = 64 (0x0040)
ICMP6: Echo Reply; ID = 64548, Seq = 6144
    ICMP6: Checksum = 0x973B (ERROR: should be 0x973A)
    ICMP6: Type = 129 (Echo Reply)
    ICMP6: Code = 0 (0x0)
    ICMP6: Identifier = 64548 (0xFC24)
    ICMP6: Sequence Number = 6144 (0x1800)
    ICMP6: Data: Number of data bytes remaining = 56 (0x0038)
```

6.2.4.2.5 Scenario 5

In the scenario number 5, we will test the situation when there are two MNs on the network. One is in a FN and, the second, in another FN; i.e. both are out of their Homes. The Figure 6-14 shows graphically this scenario.

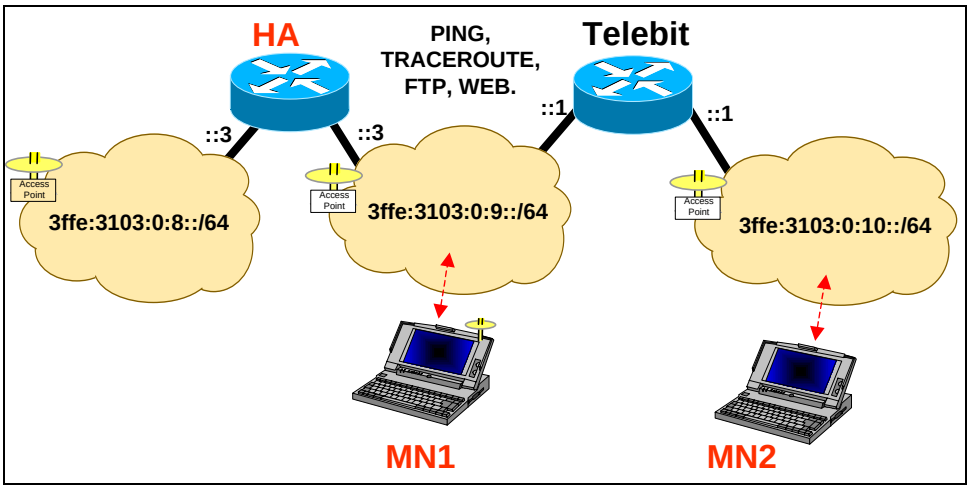


Figure 6-14: WLAN: Scenario 5: The MN1 is on the FN1 and the MN2 is on a FN2.

Characteristics of equipment involved:

Equipment Characteristics	HA	MN1	MN2	FN Router
Platform	*i386*	*i386*	*i386*	*Telebit*
Model	*"Blank" PC*	*"Blank" PC*	*"Blank" PC*	*AXI 462/3*
CPU	*Pentium II 400 MHz (cache 512 KB)*	*Pentium III 650 MHz (cache 256 KB)*	*Pentium II 266 MHz (cache 512 KB)*	*RISC (TBC 103)*
Memory	*64 RAM (swap 256 MB)*	*128 RAM (swap 256 MB)*	*64 RAM (swap 256 MB)*	*16 MB*
Network Interfaces Card	*2 Ethernet cards 3COM series 59x (driver 3c59x)*	*1 Wireless card PCI (802.11) 350 series AIR PCI 352*	*1 Ethernet card 3COM series 59x (driver 3c59x)*	*2 Ethernet cards (TCB 120) Motorola Power PC 750*
Operating System / Kernel	*Linux 7.2 (kernel 2.4.16)*	*Linux 7.2 (kernel 2.4.16)*	*Linux 7.2 (kernel 2.4.16)*	*Telebit OS swtbcpaxos 2.6a14*

** Wireless Access point used: Cisco Access Point 350 SERIES Aironet AIR AP352.*

MN1 location:

Foreign Network 1 (FN) - 3ffe:3103:0:9::/64 (via WLAN).

MN2 location:

Foreign Network 2 (FN) - 3ffe:3103:0:10::/64 (via LAN).

Applications:

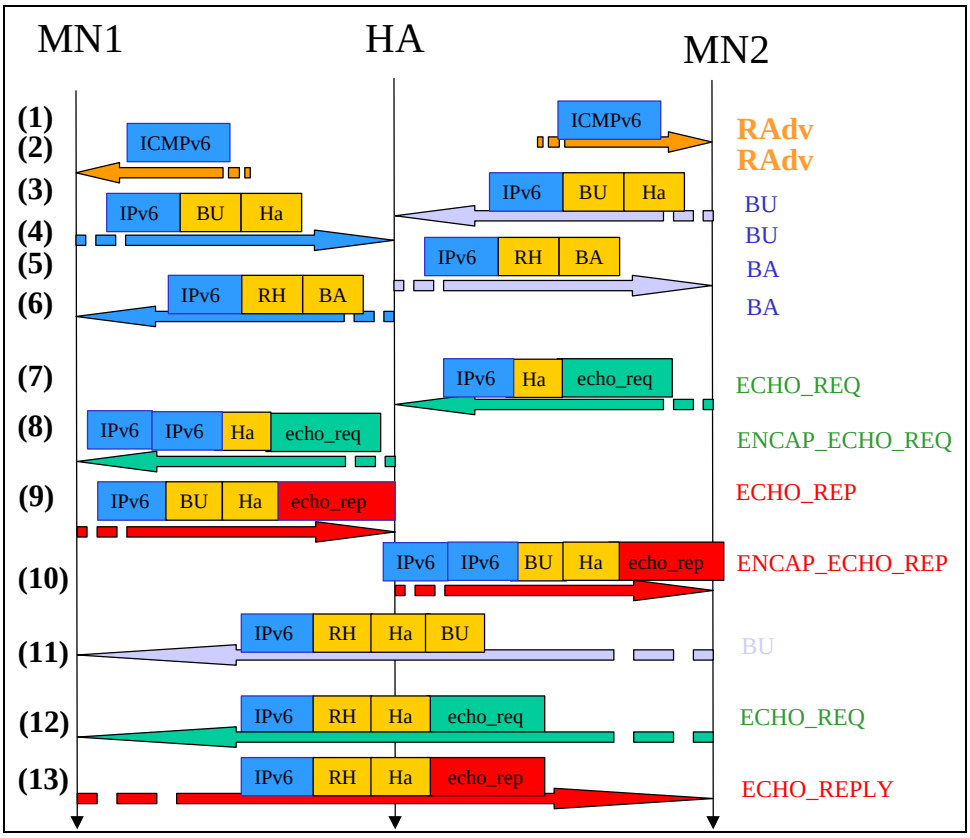
PINGs, FTPs and WWW.

External Behaviour:

Externally it works correctly with all the applications/services tested. After the movement to attach the nodes as specified in the Figure (0.5 - 2 seconds), it works properly.

Schematic DataFlow:

* Ping from the MN2 to MN1.



Packet Analysis:

The Router Advertisements (RAs) are being sent periodically by the Telebit. The prefix announced is the 3ffe:3103:0:10::/64.

The Router Advertisements (RAs) are being sent periodically by the Telebit. The prefix announced is the 3ffe:3103:0:9::/64.

A Binding Update (BU) is sent from the MN2 to the HA, in order to register the CoA acquired, using a Ha Destination Option (DH).

A Binding Update (BU) is sent from the MN1 to the HA, in order to register the CoA acquired, using a Ha Destination Option (DH).

A Binding Acknowledge (BA) from the HA confirms the BU of the MN2. The HA use a Routing Header (RH) to forward the packet.

A Binding Acknowledge (BA) from the HA confirms the BU of the MN1. The HA use a Routing Header (RH) to forward the packet.

The first time that the MN2 sends an *Echo Request* to the MN1, it is sent to the MN1 Home link. Note that the MN2 add a Ha header to the packet, because it knows that is located in a FN.

The packet is there intercepted by the HA and, is encapsulated, using IPv6 over IPv6, and sent to the MN1.

The first time that the MN1 sends an *Echo Reply* to the MN2, it is sent to the MN2 Home link. Note that the MN1 also add a Ha header to the packet, because it knows that is located in a FN.

The packet is there intercepted by the HA and, is encapsulated, using IPv6 over IPv6, and sent to the MN2.

The MN2 send a packet to add an entry in the MN1 BC. Notice that the packet doesn't have payload, but just because in this moment was not packet to be sent. Otherwise the piggybacking mechanism could be used.

Since the Binding Caches (BCs) of each MNs are updates, the communication starts to be made directly. From the MN2 to the MN1, are always sent a Ha to identify the MN2 Home address, and an RH to transport the Ha of the MN1.

Equally, the MN1 send a Ha to identify its Ha and a RH to identify the MN2 Ha.

Sniffed packets:

* The particular packets here presented are from the PING application.

(1) - Router Advertisement

```
ETHERNET: ETYPE = 0x86DD : Protocol = Unknown
IP6: Proto = ICMP6; Len = 56
    IP6: Version = 6 (0x6)
    IP6: Traffic Class = 0 (0x0)
    IP6: Flow Label = 0 (0x0)
    IP6: Payload Length = 56 (0x38)
    IP6: Next Header = 58 (ICMP6)
    IP6: Hop Limit = 255 (0xFF)
    IP6: Source Address = fe80::2c0:33ff:fe0c:a0
    IP6: Destination Address = ff02::1
    IP6: Payload: Number of data bytes remaining = 56 (0x0038)
ICMP6: Router Advertisement
    ICMP6: Checksum = 0x5AA8
    ICMP6: Type = 134 (Router Advertisement)
    ICMP6: Code = 0 (0x0)
    ICMP6: Current Hop Limit = 64 (0x40)
    ICMP6: 0....... = Not managed address config
    ICMP6: .0...... = Not other stateful config
    ICMP6: Router Lifetime = 600 (0x258)
    ICMP6: Reachable Time = 0 (0x0)
    ICMP6: Retransmission Timer = 0 (0x0)
    ICMP6: Prefix = 3ffe:3103:0:10::
        ICMP6: Type = 3 (0x3)
        ICMP6: Length = 4 (0x4)
        ICMP6: Prefix Length = 64 (0x40)
        ICMP6: Site Prefix Length = 0 (0x0)
        ICMP6: 1....... = On-link determination allowed
        ICMP6: .1...... = Autonomous address configuration
        ICMP6: ..0..... = No router address
        ICMP6: ...0.... = Not a site prefix
        ICMP6: .......0 = Not a route prefix
        ICMP6: Valid Lifetime = 2592000 (0x278D00)
        ICMP6: Preferred Lifetime = 604800 (0x93A80)
        ICMP6: Reserved
        ICMP6: Prefix = 3ffe:3103:0:10::
    ICMP6: Source Link-level Address = 00 C0 33 0C 00 A0
        ICMP6: Type = 1 (0x1)
        ICMP6: Length = 1 (0x1)
        ICMP6: Source Link-level Address = 00 C0 33 0C 00 A0
```

(2) - Router Advertisement

```
ETHERNET: ETYPE = 0x86DD : Protocol = Unknown
IP6: Proto = ICMP6; Len = 56
    IP6: Version = 6 (0x6)
    IP6: Traffic Class = 0 (0x0)
    IP6: Flow Label = 0 (0x0)
    IP6: Payload Length = 56 (0x38)
    IP6: Next Header = 58 (ICMP6)
    IP6: Hop Limit = 255 (0xFF)
    IP6: Source Address = fe80::210:5aff:fe67:55bc
    IP6: Destination Address = ff02::1
    IP6: Payload: Number of data bytes remaining = 56 (0x0038)
ICMP6: Router Advertisement
    ICMP6: Checksum = 0x5AA8
    ICMP6: Type = 134 (Router Advertisement)
    ICMP6: Code = 0 (0x0)
    ICMP6: Current Hop Limit = 64 (0x40)
    ICMP6: 0....... = Not managed address config
    ICMP6: .0...... = Not other stateful config
    ICMP6: Router Lifetime = 600 (0x258)
    ICMP6: Reachable Time = 0 (0x0)
    ICMP6: Retransmission Timer = 0 (0x0)
    ICMP6: Prefix = 3ffe:3103:0:9::
        ICMP6: Type = 3 (0x3)
        ICMP6: Length = 4 (0x4)
        ICMP6: Prefix Length = 64 (0x40)
        ICMP6: Site Prefix Length = 0 (0x0)
        ICMP6: 1....... = On-link determination allowed
        ICMP6: .1...... = Autonomous address configuration
        ICMP6: ..0..... = No router address
        ICMP6: ...0.... = Not a site prefix
        ICMP6: .......0 = Not a route prefix
        ICMP6: Valid Lifetime = 2592000 (0x278D00)
        ICMP6: Preferred Lifetime = 604800 (0x93A80)
        ICMP6: Reserved
        ICMP6: Prefix = 3ffe:3103:0:9::
    ICMP6: Source Link-level Address = 00 10 5A 67 55 BC
        ICMP6: Type = 1 (0x1)
        ICMP6: Length = 1 (0x1)
        ICMP6: Source Link-level Address = 00 10 5A 67 55 BC
```

(3) - MN2 Binding Update

```
ETHERNET: ETYPE = 0x86DD : Protocol = Unknown
IP6: Dest Opts; Auth SPI = 0x40a; Dest Opts; Len = 0
    IP6: Version = 6 (0x6)
    IP6: Traffic Class = 0 (0x0)
    IP6: Flow Label = 0 (0x0)
    IP6: Payload Length = 64 (0x40)
    IP6: Next Header = 60 (Destination Options Header)
    IP6: Hop Limit = 255 (0xFF)
    IP6: Source Address = 3ffe:3103:0:10:260:8ff:fe46:46e7
    IP6: Destination Address = 3ffe:3103:0:8::3
    IP6: Destination Options Header
        IP6: Next Header = 51 (Authentication Header)
        IP6: Length = 2 (0x2)
        IP6: Padding (4 bytes)
            IP6: Type = 1 (PadN)
                IP6: 00...... = Skip option if not recognized
                IP6: ..0..... = Option data does not change enroute
            IP6: Length = 2 (0x2)
            IP6: Data = 00 00
        IP6: Home Address Option
            IP6: Type = 201
                IP6: 11...... = Discard packet if not recognized, and send ICMP if not multicast
                IP6: ..0..... = Option data does not change enroute
            IP6: Length = 16 (0x10)
            IP6: Home Address = 3ffe:3103:0:8:260:8ff:fe46:46e7
    IP6: Authentication Header
        IP6: Next Header = 60 (Destination Options Header)
        IP6: Length = 4 (0x4)
        IP6: Reserved
        IP6: SPI = 1034 (0x40A)
        IP6: Seq Num = 4 (0x4)
        IP6: Auth Data = 16 12 37 89 41 0B 1C 88 00 8A 81 DC
    IP6: Destination Options Header
        IP6: Next Header = 59 (None)
        IP6: Length = 1 (0x1)
        IP6: Binding Update Option
            IP6: Type = 198
                IP6: 11...... = Discard packet if not recognized, and send ICMP if not multicast
                IP6: ..0..... = Option data does not change enroute
            IP6: Length = 8 (0x8)
            IP6: Flags
                IP6: 1....... = Request a binding acknowledgement response.
                IP6: .1...... = Request host to act as home agent.
                IP6: ..1..... = Binding update includes a care-of address.
            IP6: Prefix Length = 0 (0x0)
            IP6: Sequence Number = 6 (0x6)
            IP6: Lifetime = 10000 (0x2710)
*           IP6: Care-of Address = 102:0:8100:a13b:fc24:1500:5e67:133d
```

* The line Care-of Address is a wrong value because the sniffer just understands draft13.

(4) - MN1 Binding Update

```
ETHERNET: ETYPE = 0x86DD : Protocol = Unknown
IP6: Dest Opts; Auth SPI = 0x40a; Dest Opts; Len = 0
    IP6: Version = 6 (0x6)
    IP6: Traffic Class = 0 (0x0)
    IP6: Flow Label = 0 (0x0)
    IP6: Payload Length = 64 (0x40)
    IP6: Next Header = 60 (Destination Options Header)
    IP6: Hop Limit = 255 (0xFF)
    IP6: Source Address = 3ffe:3103:0:9:207:eff:feb9:69fc
    IP6: Destination Address = 3ffe:3103:0:8::3
    IP6: Destination Options Header
        IP6: Next Header = 51 (Authentication Header)
        IP6: Length = 2 (0x2)
        IP6: Padding (4 bytes)
            IP6: Type = 1 (PadN)
                IP6: 00...... = Skip option if not recognized
                IP6: ..0..... = Option data does not change enroute
            IP6: Length = 2 (0x2)
            IP6: Data = 00 00
        IP6: Home Address Option
            IP6: Type = 201
                IP6: 11...... = Discard packet if not recognized, and send ICMP if not multicast
                IP6: ..0..... = Option data does not change enroute
            IP6: Length = 16 (0x10)
            IP6: Home Address = 3ffe:3103:0:8:207:eff:feb9:69fc
    IP6: Authentication Header
        IP6: Next Header = 60 (Destination Options Header)
        IP6: Length = 4 (0x4)
        IP6: Reserved
        IP6: SPI = 1034 (0x40A)
        IP6: Seq Num = 4 (0x4)
        IP6: Auth Data = 16 12 37 89 41 0B 1C 88 00 8A 81 DC
    IP6: Destination Options Header
        IP6: Next Header = 59 (None)
        IP6: Length = 1 (0x1)
        IP6: Binding Update Option
            IP6: Type = 198
                IP6: 11...... = Discard packet if not recognized, and send ICMP if not multicast
                IP6: ..0..... = Option data does not change enroute
            IP6: Length = 8 (0x8)
            IP6: Flags
                IP6: 1....... = Request a binding acknowledgement response.
                IP6: .1...... = Request host to act as home agent.
                IP6: ..1..... = Binding update includes a care-of address.
            IP6: Prefix Length = 0 (0x0)
            IP6: Sequence Number = 6 (0x6)
```

```
            IP6: Lifetime = 10000 (0x2710)
*           IP6: Care-of Address = 102:0:8100:a13b:fc24:1500:345e:a456
```

* The line Care-of Address is a wrong value because the sniffer just understands draft13.

(5) - MN2 Binding Acknowledge

```
ETHERNET: ETYPE = 0x86DD : Protocol = Unknown
IP6: Routing (1 left of 1); Auth SPI = 0x409; Dest Opts; Len = 0
    IP6: Version = 6 (0x6)
    IP6: Traffic Class = 0 (0x0)
    IP6: Flow Label = 0 (0x0)
    IP6: Payload Length = 64 (0x40)
    IP6: Next Header = 43 (Routing Header)
    IP6: Hop Limit = 255 (0xFF)
    IP6: Source Address = 3ffe:3103:0:8::3
    IP6: Destination Address = 3ffe:3103:0:10:260:8ff:fe46:46e7
    IP6: Routing Header
        IP6: Next Header = 51 (Authentication Header)
        IP6: Length = 2 (0x2)
        IP6: Type = 0 (0x0)
        IP6: Segments Left = 1 (0x1)
        IP6: Reserved
        IP6: Route
            IP6: Address = 3ffe:3103:0:8:260:8ff:fe46:46e7
    IP6: Authentication Header
        IP6: Next Header = 60 (Destination Options Header)
        IP6: Length = 4 (0x4)
        IP6: Reserved
        IP6: SPI = 1033 (0x409)
        IP6: Seq Num = 4 (0x4)
        IP6: Auth Data = A3 C5 76 C8 39 0C 5B 55 67 4D 94 D2
    IP6: Destination Options Header
        IP6: Next Header = 59 (None)
        IP6: Length = 1 (0x1)
        IP6: Padding (1 bytes)
            IP6: Type = 0 (Pad1)
                IP6: 00...... = Skip option if not recognized
                IP6: ..0..... = Option data does not change enroute
        IP6: Binding Acknowledgement Option
            IP6: Type = 7
                IP6: 00...... = Skip option if not recognized
                IP6: ..0..... = Option data does not change enroute
            IP6: Length = 11 (0xB)
            IP6: Status: 0 = Binding accepted.
            IP6: Sequence Number = 6 (0x6)
            IP6: Lifetime = 1000 (0x3E8)
            IP6: Refresh = 800 (0x320)
```

(6) - MN1 Binding Acknowledge

```
ETHERNET: ETYPE = 0x86DD : Protocol = Unknown
IP6: Routing (1 left of 1); Auth SPI = 0x409; Dest Opts; Len = 0
    IP6: Version = 6 (0x6)
    IP6: Traffic Class = 0 (0x0)
    IP6: Flow Label = 0 (0x0)
    IP6: Payload Length = 64 (0x40)
    IP6: Next Header = 43 (Routing Header)
    IP6: Hop Limit = 255 (0xFF)
    IP6: Source Address = 3ffe:3103:0:8::3
    IP6: Destination Address = 3ffe:3103:0:9:207:eff:feb9:69fc
    IP6: Routing Header
        IP6: Next Header = 51 (Authentication Header)
        IP6: Length = 2 (0x2)
        IP6: Type = 0 (0x0)
        IP6: Segments Left = 1 (0x1)
        IP6: Reserved
        IP6: Route
            IP6: Address = 3ffe:3103:0:8:207:eff:feb9:69fc
    IP6: Authentication Header
        IP6: Next Header = 60 (Destination Options Header)
        IP6: Length = 4 (0x4)
        IP6: Reserved
        IP6: SPI = 1033 (0x409)
        IP6: Seq Num = 4 (0x4)
        IP6: Auth Data = A3 C5 76 C8 39 0C 5B 55 67 4D 94 D2
    IP6: Destination Options Header
        IP6: Next Header = 59 (None)
        IP6: Length = 1 (0x1)
        IP6: Padding (1 bytes)
            IP6: Type = 0 (Pad1)
                IP6: 00...... = Skip option if not recognized
                IP6: ..0..... = Option data does not change enroute
        IP6: Binding Acknowledgement Option
            IP6: Type = 7
                IP6: 00...... = Skip option if not recognized
                IP6: ..0..... = Option data does not change enroute
            IP6: Length = 11 (0xB)
            IP6: Status: 0 = Binding accepted.
            IP6: Sequence Number = 6 (0x6)
            IP6: Lifetime = 1000 (0x3E8)
            IP6: Refresh = 800 (0x320)
```

(7) - MN2 - Echo Request

```
ETHERNET: ETYPE = 0x86DD : Protocol = Unknown
IP6: Dest Opts; Proto = ICMP6; Len = 64
    IP6: Version = 6 (0x6)
    IP6: Traffic Class = 0 (0x0)
```

```
    IP6: Flow Label = 0 (0x0)
    IP6: Payload Length = 88 (0x58)
    IP6: Next Header = 60 (Destination Options Header)
    IP6: Hop Limit = 63 (0x3F)
    IP6: Source Address = 3ffe:3103:0:10:260:8ff:fe46:46e7
    IP6: Destination Address = 3ffe:3103:0:8:207:eff:feb9:69fc
    IP6: Destination Options Header
        IP6: Next Header = 58 (ICMP6)
        IP6: Length = 2 (0x2)
        IP6: Padding (4 bytes)
            IP6: Type = 1 (PadN)
                IP6: 00...... = Skip option if not recognized
                IP6: ..0..... = Option data does not change enroute
            IP6: Length = 2 (0x2)
            IP6: Data = 00 00
        IP6: Home Address Option
            IP6: Type = 201
                IP6: 11...... = Discard packet if not recognized, and send ICMP if not multicast
                IP6: ..0..... = Option data does not change enroute
            IP6: Length = 16 (0x10)
            IP6: Home Address = 3ffe:3103:0:8:260:8ff:fe46:46e7
    IP6: Payload: Number of data bytes remaining = 64 (0x0040)
ICMP6: Echo Request; ID = 31497, Seq = 0
    ICMP6: Checksum = 0x5501 (ERROR: should be 0x54F9)
    ICMP6: Type = 128 (Echo Request)
    ICMP6: Code = 0 (0x0)
    ICMP6: Identifier = 31497 (0x7B09)
    ICMP6: Sequence Number = 0 (0x0)
    ICMP6: Data: Number of data bytes remaining = 56 (0x0038)
```

(8) - MN2 Echo Request - HA encapsulation

```
ETHERNET: ETYPE = 0x86DD : Protocol = Unknown
IP6: Proto = 0x29; Len = 128
    IP6: Version = 6 (0x6)
    IP6: Traffic Class = 0 (0x0)
    IP6: Flow Label = 0 (0x0)
    IP6: Payload Length = 128 (0x80)
    IP6: Next Header = 41
    IP6: Hop Limit = 255 (0xFF)
    IP6: Source Address = 3ffe:3103:0:8::3
    IP6: Destination Address = 3ffe:3103:0:9:207:eff:feb9:69fc
    IP6: Payload: Number of data bytes remaining = 128 (0x0080)
IP6: Dest Opts; Proto = ICMP6; Len = 64
    IP6: Version = 6 (0x6)
    IP6: Traffic Class = 0 (0x0)
    IP6: Flow Label = 0 (0x0)
    IP6: Payload Length = 88 (0x58)
    IP6: Next Header = 60 (Destination Options Header)
    IP6: Hop Limit = 62 (0x3E)
    IP6: Source Address = 3ffe:3103:0:10:260:8ff:fe46:46e7
    IP6: Destination Address = 3ffe:3103:0:8:207:eff:feb9:69fc
    IP6: Destination Options Header
        IP6: Next Header = 58 (ICMP6)
        IP6: Length = 2 (0x2)
        IP6: Padding (4 bytes)
            IP6: Type = 1 (PadN)
                IP6: 00...... = Skip option if not recognized
                IP6: ..0..... = Option data does not change enroute
            IP6: Length = 2 (0x2)
            IP6: Data = 00 00
        IP6: Home Address Option
            IP6: Type = 201
                IP6: 11...... = Discard packet if not recognized, and send ICMP if not multicast
                IP6: ..0..... = Option data does not change enroute
            IP6: Length = 16 (0x10)
            IP6: Home Address = 3ffe:3103:0:8:260:8ff:fe46:46e7
    IP6: Payload: Number of data bytes remaining = 64 (0x0040)
ICMP6: Echo Request; ID = 31497, Seq = 0
    ICMP6: Checksum = 0x5501 (ERROR: should be 0x54F9)
    ICMP6: Type = 128 (Echo Request)
    ICMP6: Code = 0 (0x0)
    ICMP6: Identifier = 31497 (0x7B09)
    ICMP6: Sequence Number = 0 (0x0)
    ICMP6: Data: Number of data bytes remaining = 56 (0x0038)
```

(9) - MN1 Echo Reply - BU to MN2

```
ETHERNET: ETYPE = 0x86DD : Protocol = Unknown
IP6: Dest Opts; Auth SPI = 0x409; Dest Opts; Proto = ICMP6; Len = 64
    IP6: Version = 6 (0x6)
    IP6: Traffic Class = 0 (0x0)
    IP6: Flow Label = 0 (0x0)
    IP6: Payload Length = 128 (0x80)
    IP6: Next Header = 60 (Destination Options Header)
    IP6: Hop Limit = 64 (0x40)
    IP6: Source Address = 3ffe:3103:0:9:207:eff:feb9:69fc
    IP6: Destination Address = 3ffe:3103:0:8:260:8ff:fe46:46e7
    IP6: Destination Options Header
        IP6: Next Header = 51 (Authentication Header)
        IP6: Length = 2 (0x2)
        IP6: Padding (4 bytes)
            IP6: Type = 1 (PadN)
                IP6: 00...... = Skip option if not recognized
                IP6: ..0..... = Option data does not change enroute
            IP6: Length = 2 (0x2)
            IP6: Data = 00 00
        IP6: Home Address Option
            IP6: Type = 201
                IP6: 11...... = Discard packet if not recognized, and send ICMP if not multicast
```

```
                IP6: ..0..... = Option data does not change enroute
            IP6: Length = 16 (0x10)
            IP6: Home Address = 3ffe:3103:0:8:207:eff:feb9:69fc
    IP6: Authentication Header
        IP6: Next Header = 60 (Destination Options Header)
        IP6: Length = 4 (0x4)
        IP6: Reserved
        IP6: SPI = 1033 (0x409)
        IP6: Seq Num = 1 (0x1)
        IP6: Auth Data = 23 4A 57 96 67 1C EE F5 D4 3F 74 14
    IP6: Destination Options Header
        IP6: Next Header = 58 (ICMP6)
        IP6: Length = 1 (0x1)
        IP6: Binding Update Option
            IP6: Type = 198
                IP6: 11...... = Discard packet if not recognized, and send ICMP if not multicast
                IP6: ..0..... = Option data does not change enroute
            IP6: Length = 8 (0x8)
            IP6: Flags
                IP6: ..1..... = Binding update includes a care-of address.
            IP6: Prefix Length = 0 (0x0)
            IP6: Sequence Number = 3 (0x3)
            IP6: Lifetime = 1000 (0x3E8)
*           IP6: Care-of Address = 102:0:8100:a13b:fc24:1500:5e67:133d
ICMP6: Echo Reply; ID = 31497, Seq = 0
    ICMP6: Checksum = 0x5401 (ERROR: should be 0x5400)
    ICMP6: Type = 129 (Echo Reply)
    ICMP6: Code = 0 (0x0)
    ICMP6: Identifier = 31497 (0x7B09)
    ICMP6: Sequence Number = 0 (0x0)
    ICMP6: Data: Number of data bytes remaining = 56 (0x0038)
```

* The line Care-of Address is a wrong value because the sniffer just understands draft13.

(10) - MN1 Echo Reply - BU to MN2 - HA encapsulation

```
ETHERNET: ETYPE = 0x86DD : Protocol = Unknown
IP6: Proto = 0x29; Len = 168
    IP6: Version = 6 (0x6)
    IP6: Traffic Class = 0 (0x0)
    IP6: Flow Label = 0 (0x0)
    IP6: Payload Length = 168 (0xA8)
    IP6: Next Header = 41
    IP6: Hop Limit = 255 (0xFF)
    IP6: Source Address = 3ffe:3103:0:8::3
    IP6: Destination Address = 3ffe:3103:0:10:260:8ff:fe46:46e7
    IP6: Payload: Number of data bytes remaining = 168 (0x00A8)
IP6: Dest Opts; Auth SPI = 0x409; Dest Opts; Proto = ICMP6; Len = 64
    IP6: Version = 6 (0x6)
    IP6: Traffic Class = 0 (0x0)
    IP6: Flow Label = 0 (0x0)
    IP6: Payload Length = 128 (0x80)
    IP6: Next Header = 60 (Destination Options Header)
    IP6: Hop Limit = 62 (0x3E)
    IP6: Source Address = 3ffe:3103:0:9:207:eff:feb9:69fc
    IP6: Destination Address = 3ffe:3103:0:8:260:8ff:fe46:46e7
    IP6: Destination Options Header
        IP6: Next Header = 51 (Authentication Header)
        IP6: Length = 2 (0x2)
        IP6: Padding (4 bytes)
            IP6: Type = 1 (PadN)
                IP6: 00...... = Skip option if not recognized
                IP6: ..0..... = Option data does not change enroute
            IP6: Length = 2 (0x2)
            IP6: Data = 00 00
        IP6: Home Address Option
            IP6: Type = 201
                IP6: 11...... = Discard packet if not recognized, and send ICMP if not multicast
                IP6: ..0..... = Option data does not change enroute
            IP6: Length = 16 (0x10)
            IP6: Home Address = 3ffe:3103:0:8:207:eff:feb9:69fc
    IP6: Authentication Header
        IP6: Next Header = 60 (Destination Options Header)
        IP6: Length = 4 (0x4)
        IP6: Reserved
        IP6: SPI = 1033 (0x409)
        IP6: Seq Num = 1 (0x1)
        IP6: Auth Data = 23 4A 57 96 67 1C EE F5 D4 3F 74 14
    IP6: Destination Options Header
        IP6: Next Header = 58 (ICMP6)
        IP6: Length = 1 (0x1)
        IP6: Binding Update Option
            IP6: Type = 198
                IP6: 11...... = Discard packet if not recognized, and send ICMP if not multicast
                IP6: ..0..... = Option data does not change enroute
            IP6: Length = 8 (0x8)
            IP6: Flags
                IP6: ..1..... = Binding update includes a care-of address.
            IP6: Prefix Length = 0 (0x0)
            IP6: Sequence Number = 3 (0x3)
            IP6: Lifetime = 1000 (0x3E8)
*           IP6: Care-of Address = 102:0:8100:a13b:fc24:1500:5e67:133d
ICMP6: Echo Reply; ID = 31497, Seq = 0
    ICMP6: Checksum = 0x5401 (ERROR: should be 0x5400)
    ICMP6: Type = 129 (Echo Reply)
    ICMP6: Code = 0 (0x0)
    ICMP6: Identifier = 31497 (0x7B09)
    ICMP6: Sequence Number = 0 (0x0)
    ICMP6: Data: Number of data bytes remaining = 56 (0x0038)
```

* The line Care-of Address is a wrong value because the sniffer just understands draft13.

(11) - MN2 BU to MN1

```
ETHERNET: ETYPE = 0x86DD : Protocol = Unknown
IP6: Dest Opts; Routing (1 left of 1); Auth SPI = 0x40a; Dest Opts; Len = 0
    IP6: Version = 6 (0x6)
    IP6: Traffic Class = 0 (0x0)
    IP6: Flow Label = 0 (0x0)
    IP6: Payload Length = 88 (0x58)
    IP6: Next Header = 60 (Destination Options Header)
    IP6: Hop Limit = 254 (0xFE)
    IP6: Source Address = 3ffe:3103:0:10:260:8ff:fe46:46e7
    IP6: Destination Address = 3ffe:3103:0:9:207:eff:feb9:69fc
    IP6: Destination Options Header
        IP6: Next Header = 43 (Routing Header)
        IP6: Length = 2 (0x2)
        IP6: Padding (4 bytes)
            IP6: Type = 1 (PadN)
                IP6: 00...... = Skip option if not recognized
                IP6: ..0..... = Option data does not change enroute
            IP6: Length = 2 (0x2)
            IP6: Data = 00 00
        IP6: Home Address Option
            IP6: Type = 201
                IP6: 11...... = Discard packet if not recognized, and send ICMP if not multicast
                IP6: ..0..... = Option data does not change enroute
            IP6: Length = 16 (0x10)
            IP6: Home Address = 3ffe:3103:0:8:260:8ff:fe46:46e7
    IP6: Routing Header
        IP6: Next Header = 51 (Authentication Header)
        IP6: Length = 2 (0x2)
        IP6: Type = 0 (0x0)
        IP6: Segments Left = 1 (0x1)
        IP6: Reserved
        IP6: Route
            IP6: Address = 3ffe:3103:0:8:207:eff:feb9:69fc
    IP6: Authentication Header
        IP6: Next Header = 60 (Destination Options Header)
        IP6: Length = 4 (0x4)
        IP6: Reserved
        IP6: SPI = 1034 (0x40A)
        IP6: Seq Num = 3 (0x3)
        IP6: Auth Data = F1 F8 AF 6A 56 79 5D A7 4F 92 94 68
    IP6: Destination Options Header
        IP6: Next Header = 59 (None)
        IP6: Length = 1 (0x1)
        IP6: Binding Update Option
            IP6: Type = 198
                IP6: 11...... = Discard packet if not recognized, and send ICMP if not multicast
                IP6: ..0..... = Option data does not change enroute
            IP6: Length = 8 (0x8)
            IP6: Flags
                IP6: ..1..... = Binding update includes a care-of address.
            IP6: Prefix Length = 0 (0x0)
            IP6: Sequence Number = 7 (0x7)
            IP6: Lifetime = 1000 (0x3E8)
*           IP6: Care-of Address = 102:0:8100:a13b:fc24:1500:5e67:133d
```

* The line Care-of Address is a wrong value because the sniffer just understands draft13.

(12) - Echo Request - MN2 to MN1 directly

```
ETHERNET: ETYPE = 0x86DD : Protocol = Unknown
IP6: Proto = 0x29; Len = 128
    IP6: Version = 6 (0x6)
    IP6: Traffic Class = 0 (0x0)
    IP6: Flow Label = 0 (0x0)
    IP6: Payload Length = 128 (0x80)
    IP6: Next Header = 41
    IP6: Hop Limit = 255 (0xFF)
    IP6: Source Address = 3ffe:3103:0:8::3
    IP6: Destination Address = 3ffe:3103:0:9:207:eff:feb9:69fc
    IP6: Payload: Number of data bytes remaining = 128 (0x0080)
IP6: Dest Opts; Proto = ICMP6; Len = 64
    IP6: Version = 6 (0x6)
    IP6: Traffic Class = 0 (0x0)
    IP6: Flow Label = 0 (0x0)
    IP6: Payload Length = 88 (0x58)
    IP6: Next Header = 60 (Destination Options Header)
    IP6: Hop Limit = 62 (0x3E)
    IP6: Source Address = 3ffe:3103:0:10:260:8ff:fe46:46e7
    IP6: Destination Address = 3ffe:3103:0:8:207:eff:feb9:69fc
    IP6: Destination Options Header
        IP6: Next Header = 58 (ICMP6)
        IP6: Length = 2 (0x2)
        IP6: Padding (4 bytes)
            IP6: Type = 1 (PadN)
                IP6: 00...... = Skip option if not recognized
                IP6: ..0..... = Option data does not change enroute
            IP6: Length = 2 (0x2)
            IP6: Data = 00 00
        IP6: Home Address Option
            IP6: Type = 201
                IP6: 11...... = Discard packet if not recognized, and send ICMP if not multicast
                IP6: ..0..... = Option data does not change enroute
            IP6: Length = 16 (0x10)
            IP6: Home Address = 3ffe:3103:0:8:260:8ff:fe46:46e7
    IP6: Payload: Number of data bytes remaining = 64 (0x0040)
ICMP6: Echo Request; ID = 31497, Seq = 256
    ICMP6: Checksum = 0x860A (ERROR: should be 0x8602)
```

```
    ICMP6: Type = 128 (Echo Request)
    ICMP6: Code = 0 (0x0)
    ICMP6: Identifier = 31497 (0x7B09)
    ICMP6: Sequence Number = 256 (0x100)
    ICMP6: Data: Number of data bytes remaining = 56 (0x0038)
```

(13) - Echo Reply - MN1 to MN2 directly

```
ETHERNET: ETYPE = 0x86DD : Protocol = Unknown
IP6: Dest Opts; Routing (1 left of 1); Proto = ICMP6; Len = 64
    IP6: Version = 6 (0x6)
    IP6: Traffic Class = 0 (0x0)
    IP6: Flow Label = 0 (0x0)
    IP6: Payload Length = 112 (0x70)
    IP6: Next Header = 60 (Destination Options Header)
    IP6: Hop Limit = 64 (0x40)
    IP6: Source Address = 3ffe:3103:0:9:207:eff:feb9:69fc
    IP6: Destination Address = 3ffe:3103:0:10:260:8ff:fe46:46e7
    IP6: Destination Options Header
        IP6: Next Header = 43 (Routing Header)
        IP6: Length = 2 (0x2)
        IP6: Padding (4 bytes)
            IP6: Type = 1 (PadN)
                IP6: 00...... = Skip option if not recognized
                IP6: ..0..... = Option data does not change enroute
            IP6: Length = 2 (0x2)
            IP6: Data = 00 00
        IP6: Home Address Option
            IP6: Type = 201
                IP6: 11...... = Discard packet if not recognized, and send ICMP if not multicast
                IP6: ..0..... = Option data does not change enroute
            IP6: Length = 16 (0x10)
            IP6: Home Address = 3ffe:3103:0:8:207:eff:feb9:69fc
    IP6: Routing Header
        IP6: Next Header = 58 (ICMP6)
        IP6: Length = 2 (0x2)
        IP6: Type = 0 (0x0)
        IP6: Segments Left = 1 (0x1)
        IP6: Reserved
        IP6: Route
            IP6: Address = 3ffe:3103:0:8:260:8ff:fe46:46e7
    IP6: Payload: Number of data bytes remaining = 64 (0x0040)
ICMP6: Echo Reply; ID = 31497, Seq = 768
    ICMP6: Checksum = 0x810A (ERROR: should be 0x8101)
    ICMP6: Type = 129 (Echo Reply)
    ICMP6: Code = 0 (0x0)
    ICMP6: Identifier = 31497 (0x7B09)
    ICMP6: Sequence Number = 768 (0x300)
    ICMP6: Data: Number of data bytes remaining = 56 (0x0038)
```

- ### Handover on Applications

The main goal of the MIPv6 is to allow the applications behave as if they were always on its HN and never have changed their point of attachment to the network. For this reason, is very interesting to evaluate how different applications react to handover times similar to the ones referred above (about 3-5 seconds).

Two different sorts of IPv6 applications must be considered: the connection non-oriented (like UDP, ICMP) and the connection oriented (like TCP). To representing the former ones, the ping application is used (ICMP), while for the latter case the application used is the WWW (TCP). This separation must be done, because connection oriented applications should be more sensible to packets loss, due to the necessity to maintain the connection up. Actually, one of the main interests of this experiment is to see if connection oriented protocols like TCP, are able to maintain the connection up, even during handover times of a few seconds.

The behaviour described below is related to a wired scenario, where at the beginning the CN and the MN are located on the Correspondent and Home networks, respectively. After a few seconds, the MN is moved to a FN.

- **Ping**

The behaviour noticed with the ping application is described in the next 3 Figures (Figure 6-15, Figure 6-16 Figure 6-17). The actions performed were the following: a ping was started from the CN to the MN and, after a few seconds, the MN is moved to the FN.

What is noticeable to the user is that the ping is started as normally, and during the handover some packets are dropped (2 packets in that case). When the connection comes up again, the ping continues to work transparently as in the beginning.

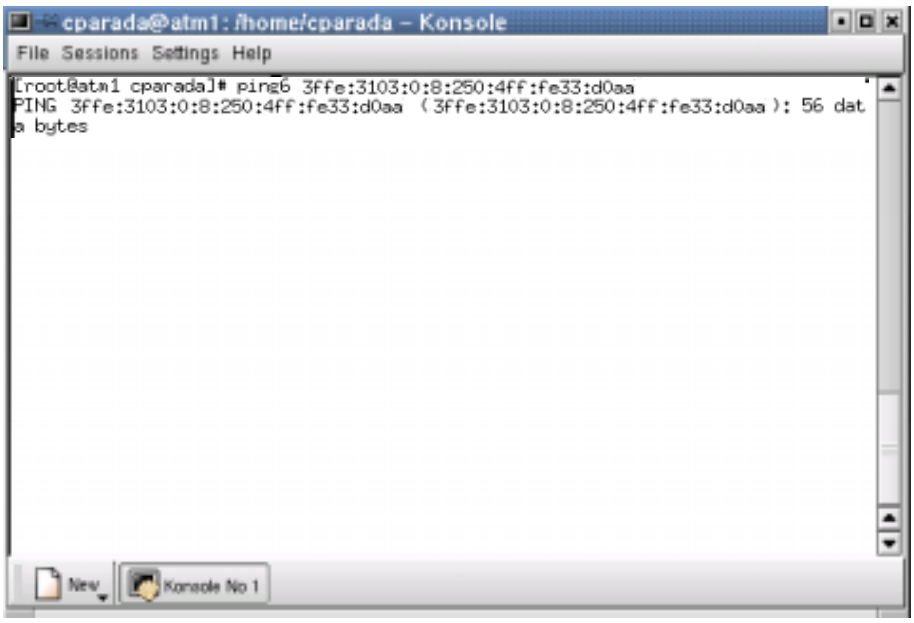


Figure 6-15: Start of the ping application (from CN to MN).

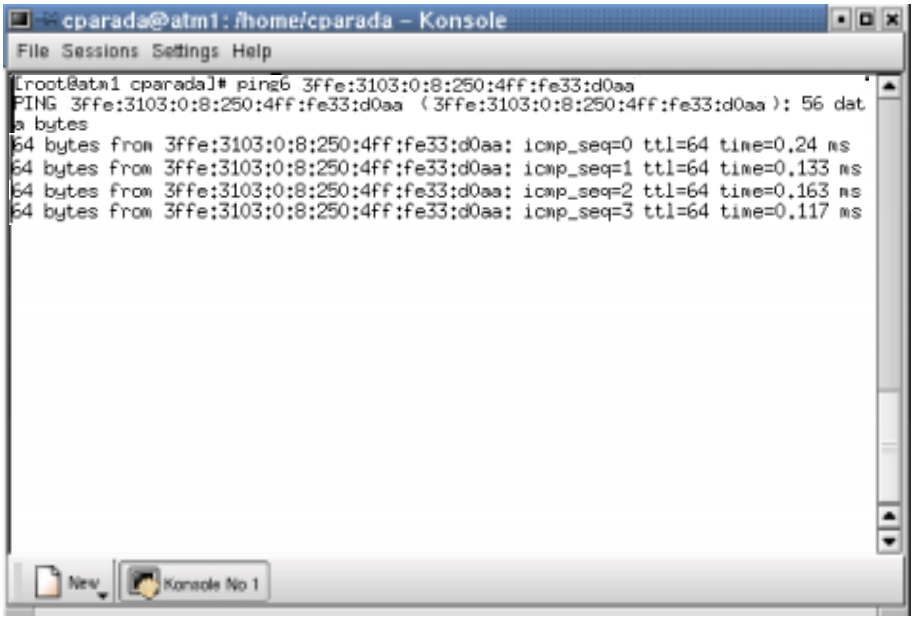


Figure 6-16: Middle of the ping application (handover is happening).

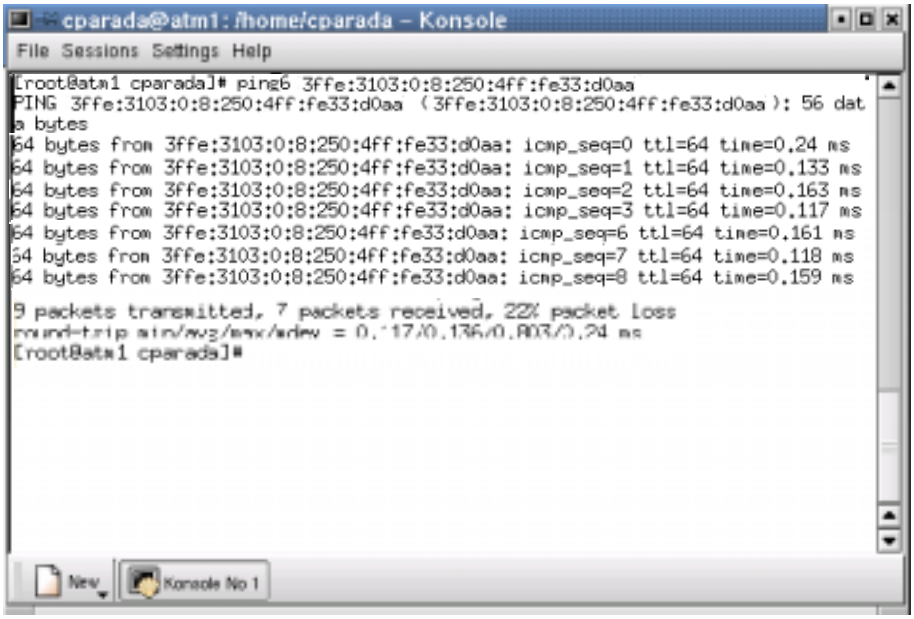


Figure 6-17: End of the ping application (with Ctrl+C).

6.2.5.2 WWW

The behaviour noticed with the a www (http) application is described in the next 3 Figures (Figure 6-18, Figure 6-19 and Figure 6-20). The actions taken were the following: from the MN, located in its HN, the main web page of the CN web server (*thttp 2.21b* IPv6 server) was accessed, using a browser (*Netscape* 6.0). Then, the download a text file with 10MB size was started, making a "*save link as*" on the proper link in the main page (file10.txt) - see the Figures. A few seconds later, when the file is around half downloaded, the MN is moved to a FN and the handover happens.

In that case, what is noticeable to the user is that the page is downloaded at the beginning as usually and, when the handover takes place, the download is stopped. After a few seconds, the download goes on from the point where it was stopped, and the file is completely and correctly downloaded.

Although the behaviour seems very similar to the ping one for the user, the true is that in this case a TCP connection was established between the we b client and the web server; i.e. between the MN and the CN. Thus, the result is very interesting, because this means that the TCP connection survived to a few second of handover times, don't breaking.

The maintenance of a TCP connection up, is one of the main characteristics that the mobility must insure. In this particular case it is insured, but this doesn't mean that will happen the same for any other web implementations, nor for any other protocols, like ftp, and so on. On the other hand, in many cases the maintenance of a TCP connection is not enough for many applications like the VoIP or videoconferencing ones.

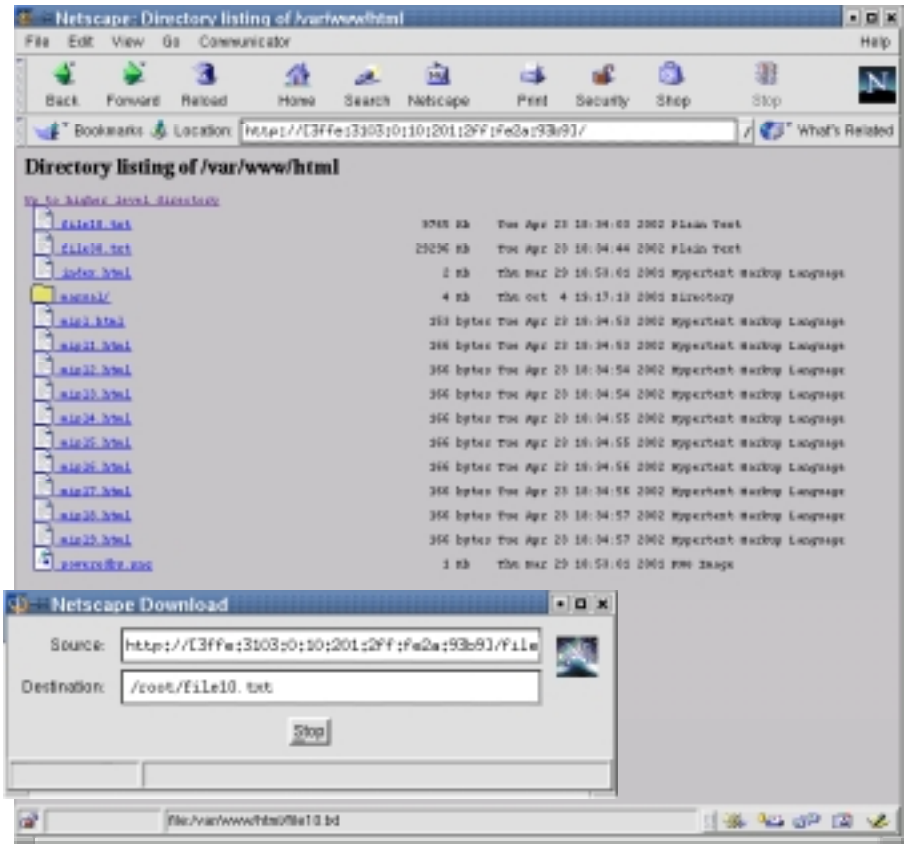


Figure 6-18: Start of the web Application (download a 10MB text file).

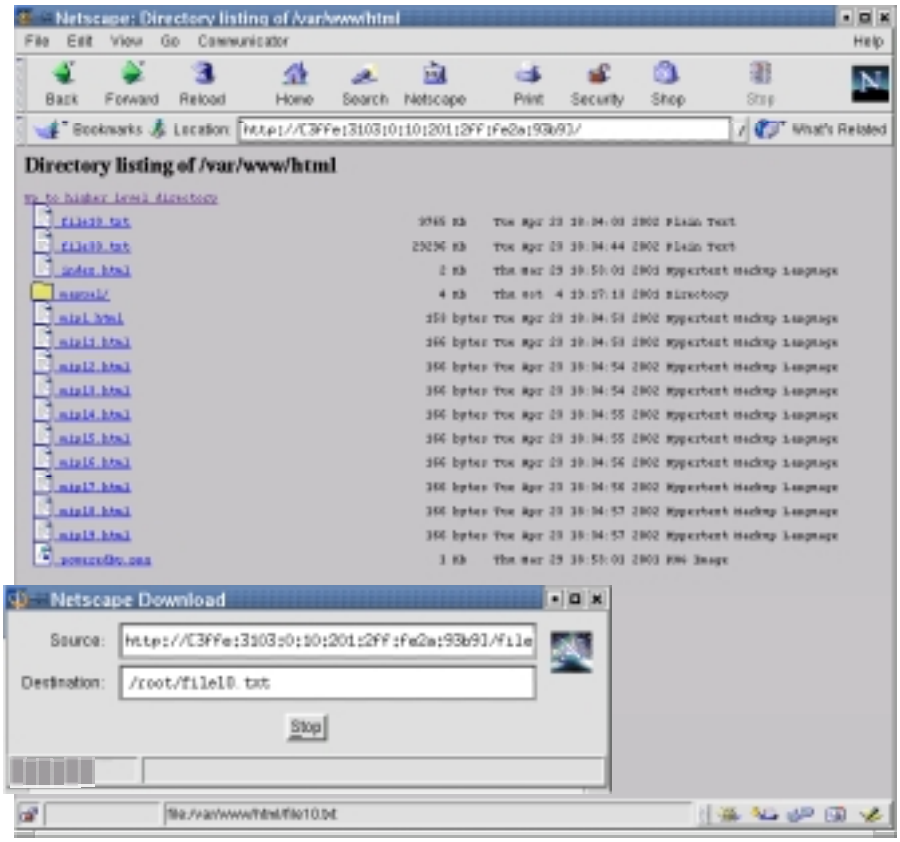


Figure 6-19: Middle of the web application (Handover is happening).

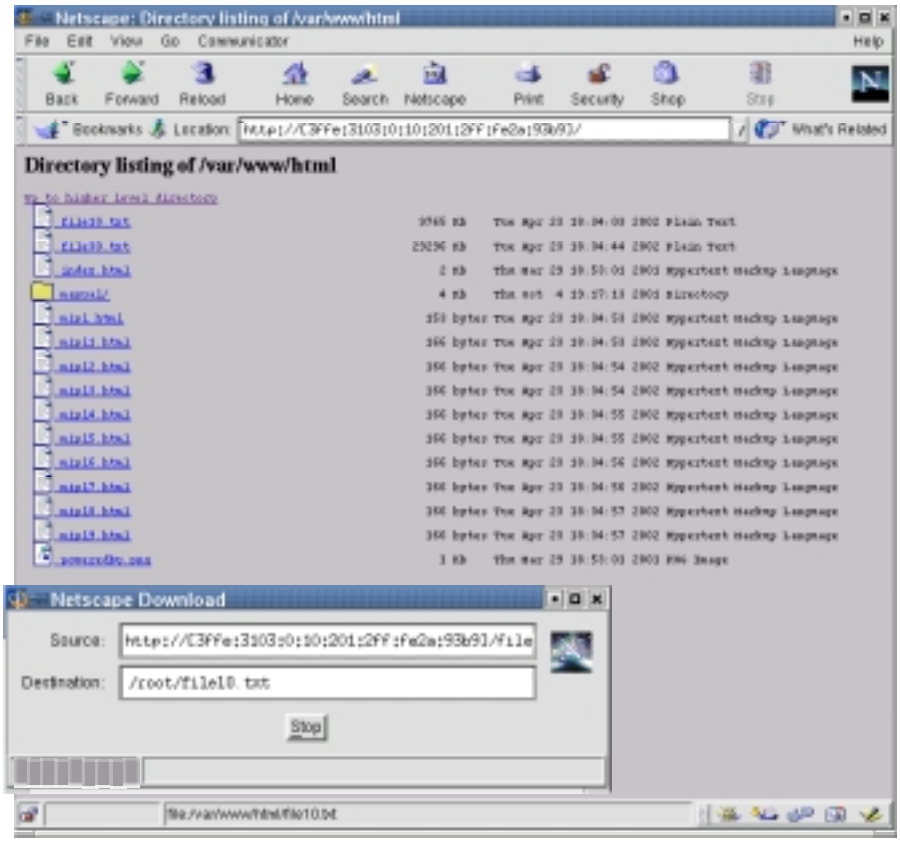


Figure 6-20: End of the web Application (10MB text file downloaded successfully).

6.2.6 Interoperability tests

Besides the Linux, there are many other MIPv6 platforms implementing different versions of the Mobility drafts. This section aims to know how some of the more representative implementations are configured and how they behave.

This section will focus on the Windows, Cisco and also de Linux implementations. The former two platforms implement the version 13 of the MIPv6 draft, while the Linux one implements the version 15.

But if it is important to test how each implementation works, it is also very important to know how the implementations inter-operate between them, in order to get heterogeneous scenarios working correctly. The case when the interoperability is most compromised, should be when different platforms implementing different draft versions are working together. This tests could also bring an additionally information about the compatibility between the draft versions.

So, in the next subsections, different scenarios of interoperability are shown, when the most representative situation are covered, in order to achieve a clear vision of what inter-operates with what.

Of course, there are substantial differences between the Linux implementation and the others, if we use authentication mechanisms presented, as optional, on the draft 15. Thus, in the next tests the authentication mechanism on Linux platforms was disabled.

The concrete configurations used for those tests for the agents involved are located on the Annexes 6.3.1.

6.2.6.1 Cisco HA; Windows MN; Windows CN

The tests described in this section were performed using a Cisco HA and a Windows MN and CN. Both implementations are based on draft 13; for this reason, it is not expected a big compatibility problems, because the differences should only refers to small detail on each implementation.

Two sequences of scenarios were tested in order to evaluate the behaviour of each implementation, as well as the interoperability capabilities between them.

6.2.6.1.1 Cisco HA; Windows MN; Windows CN

The Figure 6-21 shows schematically the first sequence, where three networks are available: HN, FN and CN. The mobile node starts in its HN (a), after that is moved to the FN (b) and backs to its HN again (c). The configurations of every entity can be seen on Annexes 6.3.2.1 and 6.3.2.2, for Cisco and Windows configurations, respectively.

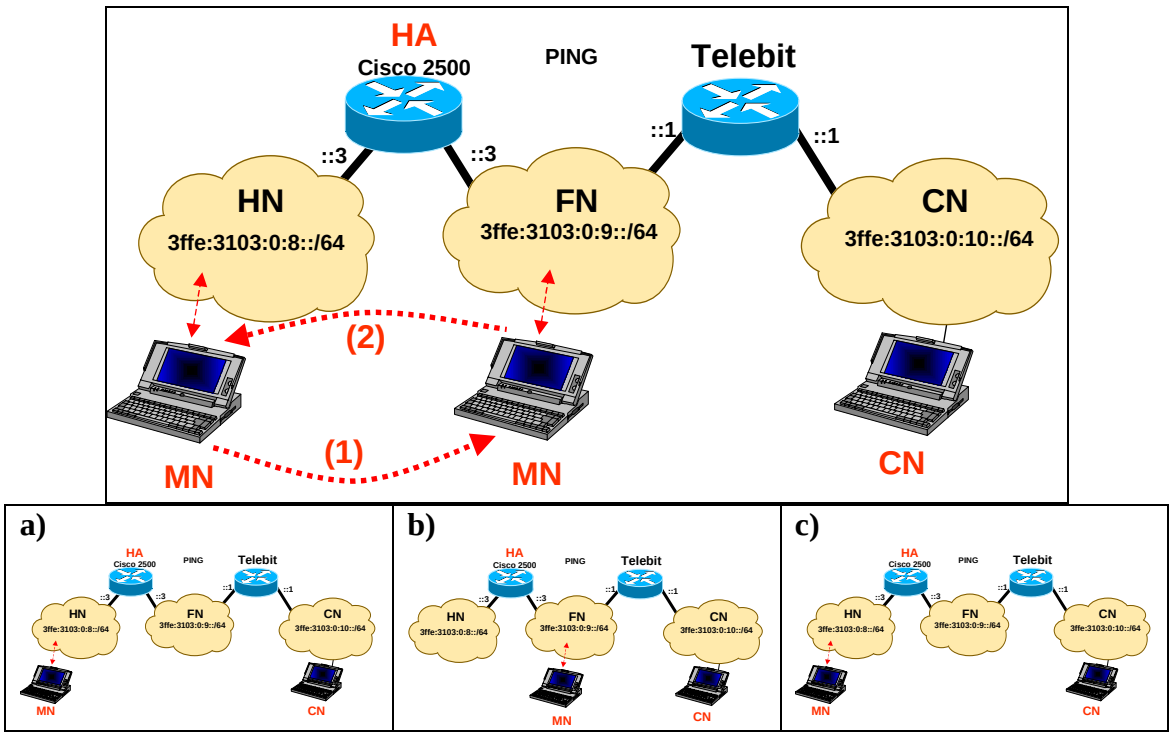


Figure 6-21: Cisco HA; Windows MN; Windows CN.

Technical characteristics of the equipment used

Equipment Characteristics	HA	MN	CN	FN Router
Platform	*Cisco 3600*	*i386*	*i386*	*Telebit*
Model	*Cisco 3640*	*"Blank" PC*	*"Blank" PC*	*AXI 462/3*
CPU	*R4700 100 MHz*	*Pentium III 300 MHz (cache 256 KB)*	*Pentium II 266 MHz (cache 512 KB)*	*RISC (TBC 103)*
Memory	*30 MB*	*64 MB RAM (swap 256 MB)*	*64 MB RAM (swap 256 MB)*	*16 MB*
Network Interfaces Card	*2 FastEthernet card*	*1 Ethernet card*	*1 Ethernet card*	*2 Ethernet cards (TCB 120) Motorola Power PC 750*
Operating System / Kernel	*Cisco IOS experimental 12.2(20010831:093907)*	*Windows 2000*	*Windows 2000 + SP2*	*Telebit OS Swtbcpaxos 2.6a14*

Most significant sniffed packets

These are some of the sniffed packets interchanged between the entities.
End.

BU: MN to HA

```
ETHERNET: ETYPE = 0x86DD : Protocol = Unknown
IP6: Dest Opts; Dest Opts; Len = 0
    IP6: Version = 6 (0x6)
    IP6: Traffic Class = 0 (0x0)
    IP6: Flow Label = 0 (0x0)
    IP6: Payload Length = 40 (0x28)
    IP6: Next Header = 60 (Destination Options Header)
    IP6: Hop Limit = 64 (0x40)
    IP6: Source Address = 3ffe:3103:0:9:280:c7ff:fe2c:eeb7
    IP6: Destination Address = 3ffe:3103:0:8::3
    IP6: Destination Options Header
        IP6: Next Header = 60 (Destination Options Header)
        IP6: Length = 2 (0x2)
        IP6: Padding (4 bytes)
            IP6: Type = 1 (PadN)
                IP6: 00...... = Skip option if not recognized
                IP6: ..0..... = Option data does not change enroute
            IP6: Length = 2 (0x2)
            IP6: Data = 00 00
        IP6: Home Address Option
            IP6: Type = 201
                IP6: 11...... = Discard packet if not recognized, and send ICMP if not multicast
                IP6: ..0..... = Option data does not change enroute
            IP6: Length = 16 (0x10)
            IP6: Home Address = 3ffe:3103:0:8:280:c7ff:fe2c:eeb7
    IP6: Destination Options Header
        IP6: Next Header = 59 (None)
        IP6: Length = 1 (0x1)
        IP6: Padding (4 bytes)
            IP6: Type = 1 (PadN)
                IP6: 00...... = Skip option if not recognized
                IP6: ..0..... = Option data does not change enroute
            IP6: Length = 2 (0x2)
            IP6: Data = 26 27
        IP6: Binding Update Option
            IP6: Type = 198
                IP6: 11...... = Discard packet if not recognized, and send ICMP if not multicast
                IP6: ..0..... = Option data does not change enroute
            IP6: Length = 8 (0x8)
            IP6: Flags
                IP6: 1....... = Request a binding acknowledgement response.
                IP6: .1...... = Request host to act as home agent.
            IP6: Prefix Length = 64 (0x40)
            IP6: Sequence Number = 710 (0x2C6)
            IP6: Lifetime = 300 (0x12C)
```

BA: HA to MN

```
ETHERNET: ETYPE = 0x86DD : Protocol = Unknown
IP6: Routing (1 left of 1); Dest Opts; Len = 0
    IP6: Version = 6 (0x6)
    IP6: Traffic Class = 0 (0x0)
```

```
    IP6: Flow Label = 0 (0x0)
    IP6: Payload Length = 40 (0x28)
    IP6: Next Header = 43 (Routing Header)
    IP6: Hop Limit = 64 (0x40)
    IP6: Source Address = 3ffe:3103:0:8::3
    IP6: Destination Address = 3ffe:3103:0:9:280:c7ff:fe2c:eeb7
    IP6: Routing Header
        IP6: Next Header = 60 (Destination Options Header)
        IP6: Length = 2 (0x2)
        IP6: Type = 0 (0x0)
        IP6: Segments Left = 1 (0x1)
        IP6: Reserved
        IP6: Route
            IP6: Address = 3ffe:3103:0:8:280:c7ff:fe2c:eeb7
    IP6: Destination Options Header
        IP6: Next Header = 59 (None)
        IP6: Length = 1 (0x1)
        IP6: Padding (1 bytes)
            IP6: Type = 0 (Pad1)
                IP6: 00...... = Skip option if not recognized
                IP6: ..0..... = Option data does not change enroute
        IP6: Binding Acknowledgement Option
            IP6: Type = 7
                IP6: 00...... = Skip option if not recognized
                IP6: ..0..... = Option data does not change enroute
            IP6: Length = 11 (0xB)
            IP6: Status: 0 = Binding accepted.
            IP6: Sequence Number = 710 (0x2C6)
            IP6: Lifetime = 0 (0x0)
            IP6: Refresh = 9830400 (0x960000)
```

IPv6/IPv6 encapsulation: HA to MN

```
ETHERNET: ETYPE = 0x86DD : Protocol = Unknown
IP6: Proto = 0x29; Len = 80
    IP6: Version = 6 (0x6)
    IP6: Traffic Class = 0 (0x0)
    IP6: Flow Label = 0 (0x0)
    IP6: Payload Length = 80 (0x50)
    IP6: Next Header = 41
    IP6: Hop Limit = 125 (0x7D)
    IP6: Source Address = 3ffe:3103:0:8::3
    IP6: Destination Address = 3ffe:3103:0:9:280:c7ff:fe2c:eeb7
    IP6: Payload: Number of data bytes remaining = 80 (0x0050)
IP6: Proto = ICMP6; Len = 40
    IP6: Version = 6 (0x6)
    IP6: Traffic Class = 0 (0x0)
    IP6: Flow Label = 0 (0x0)
    IP6: Payload Length = 40 (0x28)
    IP6: Next Header = 58 (ICMP6)
    IP6: Hop Limit = 126 (0x7E)
    IP6: Source Address = 3ffe:3103:0:10:2a0:c9ff:fe9f:9499
    IP6: Destination Address = 3ffe:3103:0:8:280:c7ff:fe2c:eeb7
    IP6: Payload: Number of data bytes remaining = 40 (0x0028)
ICMP6: Echo Reply; ID = 0, Seq = 482
    ICMP6: Checksum = 0xD8BE
    ICMP6: Type = 129 (Echo Reply)
    ICMP6: Code = 0 (0x0)
    ICMP6: Identifier = 0 (0x0)
    ICMP6: Sequence Number = 482 (0x1E2)
    ICMP6: Data: Number of data bytes remaining = 32 (0x0020)
```

BU: MN to CN

```
ETHERNET: ETYPE = 0x86DD : Protocol = Unknown
IP6: Dest Opts; Proto = ICMP6; Len = 40
    IP6: Version = 6 (0x6)
    IP6: Traffic Class = 0 (0x0)
    IP6: Flow Label = 0 (0x0)
    IP6: Payload Length = 64 (0x40)
    IP6: Next Header = 60 (Destination Options Header)
    IP6: Hop Limit = 64 (0x40)
    IP6: Source Address = 3ffe:3103:0:9:280:c7ff:fe2c:eeb7
    IP6: Destination Address = 3ffe:3103:0:10:2a0:c9ff:fe9f:9499
    IP6: Destination Options Header
        IP6: Next Header = 60 (Destination Options Header)
        IP6: Length = 2 (0x2)
        IP6: Padding (4 bytes)
            IP6: Type = 1 (PadN)
                IP6: 00...... = Skip option if not recognized
                IP6: ..0..... = Option data does not change enroute
            IP6: Length = 2 (0x2)
            IP6: Data = 02 40
        IP6: Home Address Option
            IP6: Type = 201
                IP6: 11...... = Discard packet if not recognized, and send ICMP if not multicast
                IP6: ..0..... = Option data does not change enroute
            IP6: Length = 16 (0x10)
            IP6: Home Address = 3ffe:3103:0:8:280:c7ff:fe2c:eeb7
    IP6: Payload: Number of data bytes remaining = 40 (0x0028)
IP6: Destination Options Header
        IP6: Next Header = 58 (ICMP6)
        IP6: Length = 1 (0x1)
        IP6: Binding Update Option
            IP6: Type = 198
                IP6: 11...... = Discard packet if not recognized, and send ICMP if not multicast
                IP6: ..0..... = Option data does not change enroute
            IP6: Length = 8 (0x8)
            IP6: Flags
                IP6: ..1..... = Binding update includes a care-of address.
```

```
            IP6: Prefix Length = 0 (0x0)
            IP6: Sequence Number = 3 (0x3)
            IP6: Lifetime = 1000 (0x3E8)
ICMP6: Echo Request; ID = 0, Seq = 482
    ICMP6: Checksum = 0xD9BE
    ICMP6: Type = 128 (Echo Request)
    ICMP6: Code = 0 (0x0)
    ICMP6: Identifier = 0 (0x0)
    ICMP6: Sequence Number = 482 (0x1E2)
    ICMP6: Data: Number of data bytes remaining = 32 (0x0020)
```

PING: CN to MN

```
ETHERNET: ETYPE = 0x86DD : Protocol = Unknown
IP6: Routing (1 left of 1); Proto = ICMP6; Len = 64
    IP6: Version = 6 (0x6)
    IP6: Traffic Class = 0 (0x0)
    IP6: Flow Label = 0 (0x0)
    IP6: Payload Length = 88 (0x58)
    IP6: Next Header = 43 (Routing Header)
    IP6: Hop Limit = 63 (0x3F)
    IP6: Source Address = 3ffe:3103:0:10:201:2ff:fe2a:93b9
    IP6: Destination Address = 3ffe:3103:0:9:280:c7ff:fe2c:eeb7
    IP6: Routing Header
        IP6: Next Header = 58 (ICMP6)
        IP6: Length = 2 (0x2)
        IP6: Type = 0 (0x0)
        IP6: Segments Left = 1 (0x1)
        IP6: Reserved
        IP6: Route
            IP6: Address = 3ffe:3103:0:8:280:c7ff:fe2c:eeb7
    IP6: Payload: Number of data bytes remaining = 64 (0x0040)
ICMP6: Echo Request; ID = 47631, Seq = 14376
    ICMP6: Checksum = 0x03A4 (ERROR: should be 0x03A3)
    ICMP6: Type = 128 (Echo Request)
    ICMP6: Code = 0 (0x0)
    ICMP6: Identifier = 47631 (0xBA0F)
    ICMP6: Sequence Number = 14376 (0x3828)
    ICMP6: Data: Number of data bytes remaining = 56 (0x0038)
```

PING: MN to CN

```
ETHERNET: ETYPE = 0x86DD : Protocol = Unknown
IP6: Dest Opts; Proto = ICMP6; Len = 64
    IP6: Version = 6 (0x6)
    IP6: Traffic Class = 0 (0x0)
    IP6: Flow Label = 0 (0x0)
    IP6: Payload Length = 88 (0x58)
    IP6: Next Header = 60 (Destination Options Header)
    IP6: Hop Limit = 64 (0x40)
    IP6: Source Address = 3ffe:3103:0:9:280:c7ff:fe2c:eeb7
    IP6: Destination Address = 3ffe:3103:0:10:201:2ff:fe2a:93b9
    IP6: Destination Options Header
        IP6: Next Header = 58 (ICMP6)
        IP6: Length = 2 (0x2)
        IP6: Padding (4 bytes)
            IP6: Type = 1 (PadN)
                IP6: 00...... = Skip option if not recognized
                IP6: ..0..... = Option data does not change enroute
            IP6: Length = 2 (0x2)
            IP6: Data = 00 00
        IP6: Home Address Option
            IP6: Type = 201
                IP6: 11...... = Discard packet if not recognized, and send ICMP if not multicast
                IP6: ..0..... = Option data does not change enroute
            IP6: Length = 16 (0x10)
            IP6: Home Address = 3ffe:3103:0:8:280:c7ff:fe2c:eeb7
    IP6: Payload: Number of data bytes remaining = 64 (0x0040)
ICMP6: Echo Reply; ID = 47631, Seq = 14376
    ICMP6: Checksum = 0x02A4 (ERROR: should be 0x02A3)
    ICMP6: Type = 129 (Echo Reply)
    ICMP6: Code = 0 (0x0)
    ICMP6: Identifier = 47631 (0xBA0F)
    ICMP6: Sequence Number = 14376 (0x3828)
    ICMP6: Data: Number of data bytes remaining = 56 (0x0038)
```

6.2.6.1.2 Cisco HA; Windows MN1; Windows MN2

The second sequence of tests can be seen in the Figure 6-22. In these tests, two MNs are present. At the beginning both are on its HN (a). After, the MN2 is moved to the FN2 (b), and finally the MN1 is moved to the FN1 (c). The configurations of every agent can be seen on Annexes 6.3.2.1and 6.3.2.2, for Cisco and Windows configurations, respectively. Is important to note on the Windows implementation is mandatory to configure the MN functionality and at the same time the CN, to allow the PCs to work as expected. This happen because the MN functionality doesn't include automatically the CN one.

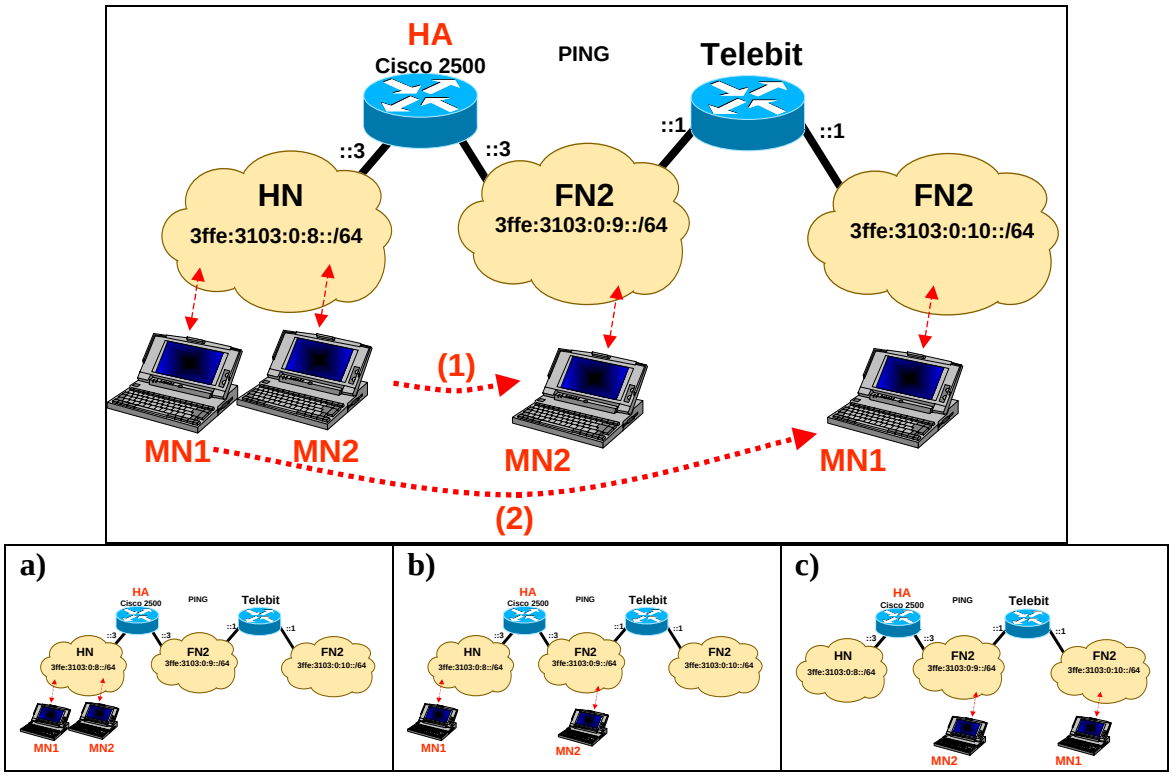


Figure 6-22: Cisco HA; Windows MN1; Windows MN2.

Technical characteristics of the equipment used

Equipment Characteristics	HA	MN1	MN2	FN Router
Platform	*Cisco 3600*	*i386*	*i386*	*Telebit*
Model	*Cisco 3640*	*"Blank" PC*	*"Blank" PC*	*AXI 462/3*
CPU	*R4700 100 MHz*	*Pentium III 300 MHz (cache 256 KB)*	*Pentium II 266 MHz (cache 512 KB)*	*RISC (TBC 103)*
Memory	*30 MB*	*64 MB RAM (swap 256 MB)*	*64 MB RAM (swap 256 MB)*	*16 MB*
Network Interfaces Card	*2 FastEthernet card*	*1 Ethernet card*	*1 Ethernet card*	*2 Ethernet cards (TCB 120) Motorola Power PC 750*
Operating System / Kernel	*Cisco IOS experimental 12.2(20010831:093907)*	*Windows 2000*	*Windows 2000 + SP2*	*Telebit OS Swtbcpaxos 2.6a14*

Most significant sniffed packets

PING: MN1 to MN2

```
ETHERNET: ETYPE = 0x86DD : Protocol = Unknown
IP6: Routing (1 left of 1); Dest Opts; Proto = ICMP6; Len = 40
    IP6: Version = 6 (0x6)
    IP6: Traffic Class = 0 (0x0)
    IP6: Flow Label = 0 (0x0)
    IP6: Payload Length = 88 (0x58)
    IP6: Next Header = 43 (Routing Header)
    IP6: Hop Limit = 63 (0x3F)
    IP6: Source Address = 3ffe:3103:0:10:2a0:c9ff:fe9f:9499
    IP6: Destination Address = 3ffe:3103:0:9:280:c7ff:fe2c:eeb7
    IP6: Routing Header
        IP6: Next Header = 60 (Destination Options Header)
        IP6: Length = 2 (0x2)
        IP6: Type = 0 (0x0)
        IP6: Segments Left = 1 (0x1)
        IP6: Reserved
        IP6: Route
            IP6: Address = 3ffe:3103:0:8:280:c7ff:fe2c:eeb7
    IP6: Destination Options Header
        IP6: Next Header = 58 (ICMP6)
        IP6: Length = 2 (0x2)
        IP6: Padding (4 bytes)
            IP6: Type = 1 (PadN)
                IP6: 00...... = Skip option if not recognized
                IP6: ..0..... = Option data does not change enroute
            IP6: Length = 2 (0x2)
            IP6: Data = 3F FE
        IP6: Home Address Option
            IP6: Type = 201
                IP6: 11...... = Discard packet if not recognized, and send ICMP if not multicast
                IP6: ..0..... = Option data does not change enroute
            IP6: Length = 16 (0x10)
            IP6: Home Address = 3ffe:3103:0:8:2a0:c9ff:fe9f:9499
    IP6: Payload: Number of data bytes remaining = 40 (0x0028)
ICMP6: Echo Request; ID = 0, Seq = 249
    ICMP6: Checksum = 0xDAAF (ERROR: should be 0xDAA6)
    ICMP6: Type = 128 (Echo Request)
    ICMP6: Code = 0 (0x0)
    ICMP6: Identifier = 0 (0x0)
    ICMP6: Sequence Number = 249 (0xF9)
    ICMP6: Data: Number of data bytes remaining = 32 (0x0020)
```

PING: MN2 to MN1

```
ETHERNET: ETYPE = 0x86DD : Protocol = Unknown
IP6: Routing (1 left of 1); Dest Opts; Proto = ICMP6; Len = 40
    IP6: Version = 6 (0x6)
    IP6: Traffic Class = 0 (0x0)
    IP6: Flow Label = 0 (0x0)
    IP6: Payload Length = 88 (0x58)
    IP6: Next Header = 43 (Routing Header)
    IP6: Hop Limit = 64 (0x40)
    IP6: Source Address = 3ffe:3103:0:9:280:c7ff:fe2c:eeb7
    IP6: Destination Address = 3ffe:3103:0:10:2a0:c9ff:fe9f:9499
    IP6: Routing Header
        IP6: Next Header = 60 (Destination Options Header)
        IP6: Length = 2 (0x2)
        IP6: Type = 0 (0x0)
```

```
        IP6: Segments Left = 1 (0x1)
        IP6: Reserved
        IP6: Route
            IP6: Address = 3ffe:3103:0:8:2a0:c9ff:fe9f:9499
    IP6: Destination Options Header
        IP6: Next Header = 58 (ICMP6)
        IP6: Length = 2 (0x2)
        IP6: Padding (4 bytes)
            IP6: Type = 1 (PadN)
                IP6: 00...... = Skip option if not recognized
                IP6: ..0..... = Option data does not change enroute
            IP6: Length = 2 (0x2)
            IP6: Data = BA 0F
        IP6: Home Address Option
            IP6: Type = 201
                IP6: 11...... = Discard packet if not recognized, and send ICMP if not multicast
                IP6: ..0..... = Option data does not change enroute
            IP6: Length = 16 (0x10)
            IP6: Home Address = 3ffe:3103:0:8:280:c7ff:fe2c:eeb7
    IP6: Payload: Number of data bytes remaining = 40 (0x0028)
ICMP6: Echo Reply; ID = 0, Seq = 249
    ICMP6: Checksum = 0xD9AF (ERROR: should be 0xD9A6)
    ICMP6: Type = 129 (Echo Reply)
    ICMP6: Code = 0 (0x0)
    ICMP6: Identifier = 0 (0x0)
    ICMP6: Sequence Number = 249 (0xF9)
    ICMP6: Data: Number of data bytes remaining = 32 (0x0020)
```

End.

6.2.6.1.3 Conclusions

This set of tests allows us to get some information about the implementations.

The first one is the correct behaviour of the Windows, both as a CN and as a MN. Although the implementation has some instability, which in some cases makes a "blue screens" and in some others obligates to shut down the IPv6 stack, the functionality seems to be completely and well implemented.

On the other hand, the Cisco implementation is also well implemented and no problems were detected. Other functionalities like restriction on MN, different lifetime, and other are already well implemented. The information that can be consulted on the router is not too much, but it is helpful.

Finally, we can also conclude that both implementations are complete compatible.

6.2.6.2 Linux HA; Linux and Windows (MN and CN)

The tests described in this section were performed using a Linux HA, being the MNs and CNs, either a Windows or a Linux PCs. In this case, the test involves three different implementations and also two different versions of the MIPv6 draft (15 and 13). Here, is intended to check how much compatibility we can achieve between not just the implementation, but the draft versions.

To check all the possibilities there are tree scenarios to be analyzed, covering the three possible combinations.

6.2.6.2.1 Linux HA; Linux MN; Windows CN

The first scenario is shown in the Figure 6-23. It includes the Linux HA, a Linux MN and a Windows CN. This should be the easier scenario to achieve compatibility, because the different draft implementation is acting as a CN, which is the less complex task in mobility.

The scenario is equal to the one implemented in the section above. The MN is started in its HN (a), after that it is moved to the FN (b), and finally is moved back to its HN (c).

The configurations of every entity can be seen on Annexes 6.3.2.7, 6.3.2.6 and 6.3.2.2, for Linux HA, Linux MN and Windows CN configurations, respectively.

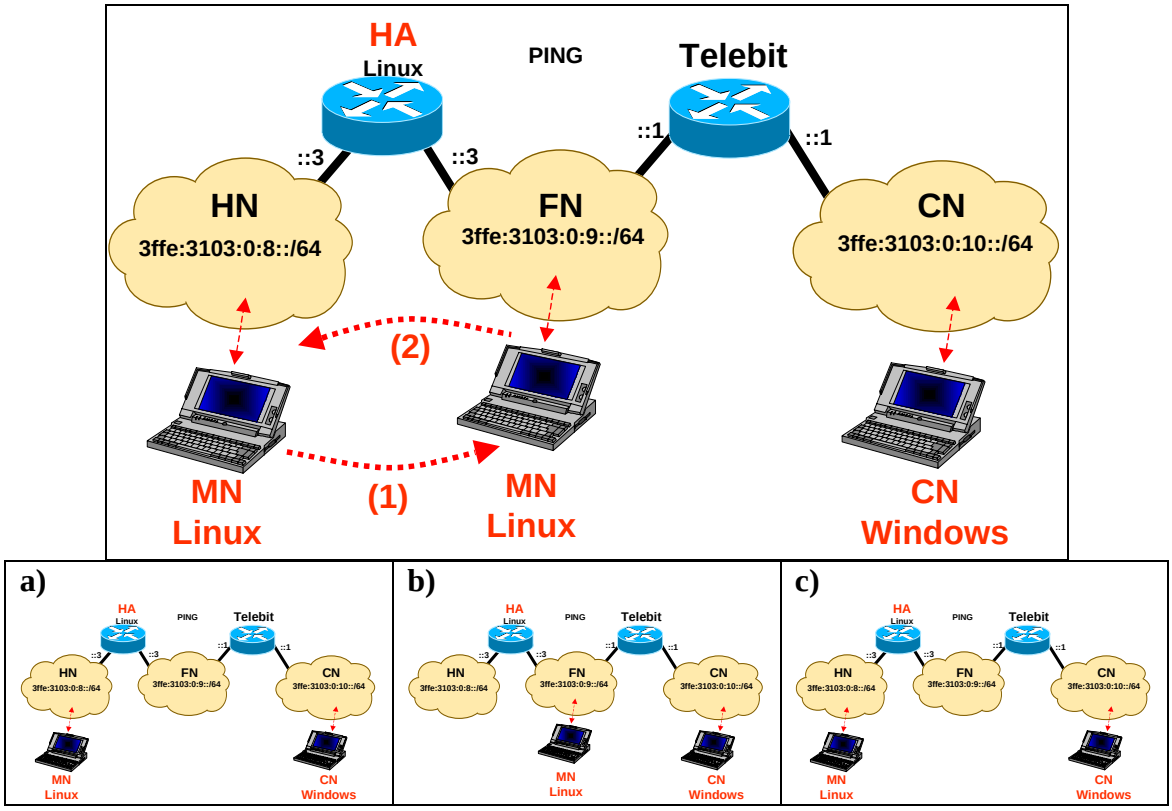


Figure 6-23: Linux HA; Linux MN; Windows CN.

Technical characteristics of the equipment used

Equipment Characteristics	HA	MN	CN	FN Router
Platform	*i386*	*i386*	*i386*	*Telebit*
Model	*"Blank" PC*	*"Blank" PC*	*"Blank" PC*	*AXI 462/3*
CPU	*Pentium II 400 MHz (cache 512 KB)*	*Pentium III 650 MHz (cache 256 KB)*	*Pentium II 266 MHz (cache 512 KB)*	*RISC (TBC 103)*
Memory	*64 RAM (swap 256 MB)*	*128 RAM (swap 256 MB)*	*64 RAM*	*16 MB*
Network Interfaces Card	*2 Ethernet cards 3COM series 59x (driver 3c59x)*	*1 Ethernet card 3COM series 59x (driver 3c59x)*	*1 Ethernet card 3COM series 59x*	*2 Ethernet cards (TCB 120) Motorola Power PC 750*
Operating System / Kernel	*Linux 7.2 (kernel 2.4.16)*	*Linux 7.2 (kernel 2.4.16)*	*Windows 2000*	*Telebit OS Swtbcpaxos 2.6a14*

Most significant sniffed packets

BU: MN to HA

```
ETHERNET: ETYPE = 0x86DD : Protocol = Unknown
IP6: Dest Opts; Dest Opts; Len = 0
    IP6: Version = 6 (0x6)
    IP6: Traffic Class = 0 (0x0)
    IP6: Flow Label = 0 (0x0)
    IP6: Payload Length = 40 (0x28)
    IP6: Next Header = 60 (Destination Options Header)
    IP6: Hop Limit = 255 (0xFF)
    IP6: Source Address = 3ffe:3103:0:9:201:2ff:fe2a:93b9
    IP6: Destination Address = 3ffe:3103:0:8::3
    IP6: Destination Options Header
        IP6: Next Header = 60 (Destination Options Header)
        IP6: Length = 2 (0x2)
        IP6: Padding (4 bytes)
            IP6: Type = 1 (PadN)
                IP6: 00...... = Skip option if not recognized
                IP6: ..0..... = Option data does not change enroute
            IP6: Length = 2 (0x2)
            IP6: Data = 00 00
        IP6: Home Address Option
            IP6: Type = 201
                IP6: 11...... = Discard packet if not recognized, and send ICMP if not multicast
                IP6: ..0..... = Option data does not change enroute
            IP6: Length = 16 (0x10)
            IP6: Home Address = 3ffe:3103:0:8:201:2ff:fe2a:93b9
    IP6: Destination Options Header
        IP6: Next Header = 59 (None)
        IP6: Length = 1 (0x1)
        IP6: Binding Update Option
            IP6: Type = 198
                IP6: 11...... = Discard packet if not recognized, and send ICMP if not multicast
                IP6: ..0..... = Option data does not change enroute
            IP6: Length = 8 (0x8)
            IP6: Flags
                IP6: 1....... = Request a binding acknowledgement response.
                IP6: .1...... = Request host to act as home agent.
                IP6: ..1..... = Binding update includes a care-of address.
            IP6: Prefix Length = 0 (0x0)
            IP6: Sequence Number = 0 (0x0)
            IP6: Lifetime = 10000 (0x2710)
*           IP6: Care-of Address = 102:0:bd53:6e00::5e00:0
```

* The line Care-of Address is a wrong value because the sniffer just understands draft13.

BA: HA to MN

```
ETHERNET: ETYPE = 0x86DD : Protocol = Unknown
IP6: Routing (1 left of 1); Dest Opts; Len = 0
    IP6: Version = 6 (0x6)
    IP6: Traffic Class = 0 (0x0)
    IP6: Flow Label = 0 (0x0)
    IP6: Payload Length = 40 (0x28)
    IP6: Next Header = 43 (Routing Header)
    IP6: Hop Limit = 255 (0xFF)
    IP6: Source Address = 3ffe:3103:0:8::3
    IP6: Destination Address = 3ffe:3103:0:9:201:2ff:fe2a:93b9
```

```
    IP6: Routing Header
        IP6: Next Header = 60 (Destination Options Header)
        IP6: Length = 2 (0x2)
        IP6: Type = 0 (0x0)
        IP6: Segments Left = 1 (0x1)
        IP6: Reserved
        IP6: Route
            IP6: Address = 3ffe:3103:0:8:201:2ff:fe2a:93b9
    IP6: Destination Options Header
        IP6: Next Header = 59 (None)
        IP6: Length = 1 (0x1)
        IP6: Padding (1 bytes)
            IP6: Type = 0 (Pad1)
                IP6: 00...... = Skip option if not recognized
                IP6: ..0..... = Option data does not change enroute
        IP6: Binding Acknowledgement Option
            IP6: Type = 7
                IP6: 00...... = Skip option if not recognized
                IP6: ..0..... = Option data does not change enroute
            IP6: Length = 11 (0xB)
            IP6: Status: 0 = Binding accepted.
            IP6: Sequence Number = 0 (0x0)
            IP6: Lifetime = 1000 (0x3E8)
            IP6: Refresh = 800 (0x320)
```

IPv6/IPv6 encapsulation: HA to MN

```
ETHERNET: ETYPE = 0x86DD : Protocol = Unknown
IP6: Proto = 0x29; Len = 80
    IP6: Version = 6 (0x6)
    IP6: Traffic Class = 0 (0x0)
    IP6: Flow Label = 0 (0x0)
    IP6: Payload Length = 80 (0x50)
    IP6: Next Header = 41
    IP6: Hop Limit = 255 (0xFF)
    IP6: Source Address = 3ffe:3103:0:8::3
    IP6: Destination Address = 3ffe:3103:0:9:201:2ff:fe2a:93b9
    IP6: Payload: Number of data bytes remaining = 80 (0x0050)
IP6: Proto = ICMP6; Len = 40
    IP6: Version = 6 (0x6)
    IP6: Traffic Class = 0 (0x0)
    IP6: Flow Label = 0 (0x0)
    IP6: Payload Length = 40 (0x28)
    IP6: Next Header = 58 (ICMP6)
    IP6: Hop Limit = 63 (0x3F)
    IP6: Source Address = 3ffe:3103:0:10:280:c7ff:fe2c:eeb7
    IP6: Destination Address = 3ffe:3103:0:8:201:2ff:fe2a:93b9
    IP6: Payload: Number of data bytes remaining = 40 (0x0028)
ICMP6: Echo Request; ID = 0, Seq = 411
    ICMP6: Checksum = 0xA302
    ICMP6: Type = 128 (Echo Request)
    ICMP6: Code = 0 (0x0)
    ICMP6: Identifier = 0 (0x0)
    ICMP6: Sequence Number = 411 (0x19B)
    ICMP6: Data: Number of data bytes remaining = 32 (0x0020)
```

BU: MN to CN

```
ETHERNET: ETYPE = 0x86DD : Protocol = Unknown
IP6: Dest Opts; Dest Opts; Proto = ICMP6; Len = 40
    IP6: Version = 6 (0x6)
    IP6: Traffic Class = 0 (0x0)
    IP6: Flow Label = 0 (0x0)
    IP6: Payload Length = 80 (0x50)
    IP6: Next Header = 60 (Destination Options Header)
    IP6: Hop Limit = 64 (0x40)
    IP6: Source Address = 3ffe:3103:0:9:201:2ff:fe2a:93b9
    IP6: Destination Address = 3ffe:3103:0:10:280:c7ff:fe2c:eeb7
    IP6: Destination Options Header
        IP6: Next Header = 60 (Destination Options Header)
        IP6: Length = 2 (0x2)
        IP6: Padding (4 bytes)
            IP6: Type = 1 (PadN)
                IP6: 00...... = Skip option if not recognized
                IP6: ..0..... = Option data does not change enroute
            IP6: Length = 2 (0x2)
            IP6: Data = 00 00
        IP6: Home Address Option
            IP6: Type = 201
                IP6: 11...... = Discard packet if not recognized, and send ICMP if not multicast
                IP6: ..0..... = Option data does not change enroute
            IP6: Length = 16 (0x10)
            IP6: Home Address = 3ffe:3103:0:8:201:2ff:fe2a:93b9
    IP6: Destination Options Header
        IP6: Next Header = 58 (ICMP6)
        IP6: Length = 1 (0x1)
        IP6: Binding Update Option
            IP6: Type = 198
                IP6: 11...... = Discard packet if not recognized, and send ICMP if not multicast
                IP6: ..0..... = Option data does not change enroute
            IP6: Length = 8 (0x8)
            IP6: Flags
                IP6: ..1..... = Binding update includes a care-of address.
            IP6: Prefix Length = 0 (0x0)
            IP6: Sequence Number = 3 (0x3)
            IP6: Lifetime = 1000 (0x3E8)
*           IP6: Care-of Address = 102:0:8100:a201:0:19c:6162:6364
ICMP6: Echo Reply; ID = 0, Seq = 412
    ICMP6: Checksum = 0xA201 (ERROR: should be 0xA200)
    ICMP6: Type = 129 (Echo Reply)
```

```
    ICMP6: Code = 0 (0x0)
    ICMP6: Identifier = 0 (0x0)
    ICMP6: Sequence Number = 412 (0x19C)
    ICMP6: Data: Number of data bytes remaining = 32 (0x0020)
```

* The line Care-of Address is a wrong value because the sniffer just understands draft13.

PING: CN to MN

```
ETHERNET: ETYPE = 0x86DD : Protocol = Unknown
IP6: Routing (1 left of 1); Proto = ICMP6; Len = 40
    IP6: Version = 6 (0x6)
    IP6: Traffic Class = 0 (0x0)
    IP6: Flow Label = 0 (0x0)
    IP6: Payload Length = 64 (0x40)
    IP6: Next Header = 43 (Routing Header)
    IP6: Hop Limit = 63 (0x3F)
    IP6: Source Address = 3ffe:3103:0:10:280:c7ff:fe2c:eeb7
    IP6: Destination Address = 3ffe:3103:0:9:201:2ff:fe2a:93b9
    IP6: Routing Header
        IP6: Next Header = 58 (ICMP6)
        IP6: Length = 2 (0x2)
        IP6: Type = 0 (0x0)
        IP6: Segments Left = 1 (0x1)
        IP6: Reserved
        IP6: Route
            IP6: Address = 3ffe:3103:0:8:201:2ff:fe2a:93b9
    IP6: Payload: Number of data bytes remaining = 40 (0x0028)
ICMP6: Echo Request; ID = 0, Seq = 413
    ICMP6: Checksum = 0xA300 (ERROR: should be 0xA2FF)
    ICMP6: Type = 128 (Echo Request)
    ICMP6: Code = 0 (0x0)
    ICMP6: Identifier = 0 (0x0)
    ICMP6: Sequence Number = 413 (0x19D)
    ICMP6: Data: Number of data bytes remaining = 32 (0x0020)
```

PING: MN to CN

```
ETHERNET: ETYPE = 0x86DD : Protocol = Unknown
IP6: Dest Opts; Proto = ICMP6; Len = 40
    IP6: Version = 6 (0x6)
    IP6: Traffic Class = 0 (0x0)
    IP6: Flow Label = 0 (0x0)
    IP6: Payload Length = 64 (0x40)
    IP6: Next Header = 60 (Destination Options Header)
    IP6: Hop Limit = 64 (0x40)
    IP6: Source Address = 3ffe:3103:0:9:201:2ff:fe2a:93b9
    IP6: Destination Address = 3ffe:3103:0:10:280:c7ff:fe2c:eeb7
    IP6: Destination Options Header
        IP6: Next Header = 58 (ICMP6)
        IP6: Length = 2 (0x2)
        IP6: Padding (4 bytes)
            IP6: Type = 1 (PadN)
                IP6: 00...... = Skip option if not recognized
                IP6: ..0..... = Option data does not change enroute
            IP6: Length = 2 (0x2)
            IP6: Data = 00 00
        IP6: Home Address Option
            IP6: Type = 201
                IP6: 11...... = Discard packet if not recognized, and send ICMP if not multicast
                IP6: ..0..... = Option data does not change enroute
            IP6: Length = 16 (0x10)
            IP6: Home Address = 3ffe:3103:0:8:201:2ff:fe2a:93b9
    IP6: Payload: Number of data bytes remaining = 40 (0x0028)
ICMP6: Echo Reply; ID = 0, Seq = 413
    ICMP6: Checksum = 0xA200 (ERROR: should be 0xA1FF)
    ICMP6: Type = 129 (Echo Reply)
    ICMP6: Code = 0 (0x0)
    ICMP6: Identifier = 0 (0x0)
    ICMP6: Sequence Number = 413 (0x19D)
    ICMP6: Data: Number of data bytes remaining = 32 (0x0020)
```

6.2.6.2.2 Linux HA; Windows MN; Linux CN

The second scenario is shown in the Figure 6-24. It includes the Linux HA, a Windows MN and a Linux CN.

The scenario is equal to the one implemented in the above section. The MN is started in its HN (a), after that is moved to the FN (b), and finally it is moved back to its HN (c).

The configurations of every entity can be seen the annexes.

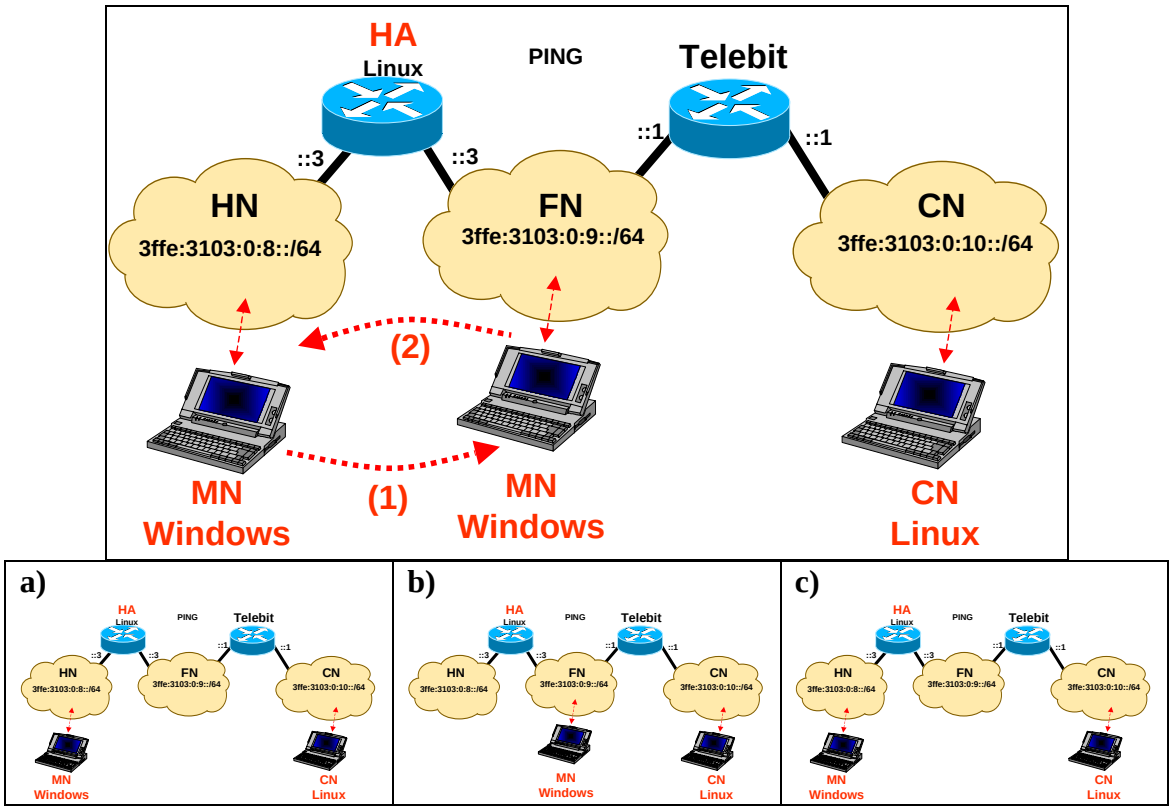


Figure 6-24: Linux HA; Windows MN; Linux CN.

Technical characteristics of the equipment used

Equipment Characteristics	HA	CN	MN	FN Router
Platform	*i386*	*i386*	*i386*	*Telebit*
Model	*"Blank" PC*	*"Blank" PC*	*"Blank" PC*	*AXI 462/3*
CPU	*Pentium II 400 MHz (cache 512 KB)*	*Pentium III 650 MHz (cache 256 KB)*	*Pentium II 266 MHz (cache 512 KB)*	*RISC (TBC 103)*
Memory	*64 RAM (swap 256 MB)*	*128 RAM (swap 256 MB)*	*64 RAM*	*16 MB*
Network Interfaces Card	*2 Ethernet cards 3COM series 59x (driver 3c59x)*	*1 Ethernet card 3COM series 59x (driver 3c59x)*	*1 Ethernet card 3COM series 59x*	*2 Ethernet cards (TCB 120) Motorola Power PC 750*
Operating System / Kernel	*Linux 7.2 (kernel 2.4.16)*	*Linux 7.2 (kernel 2.4.16)*	*Windows 2000*	*Telebit OS Swtbcpaxos 2.6a14*

Most significant sniffed packets

BU: MN to HA

```
ETHERNET: ETYPE = 0x86DD : Protocol = Unknown
IP6: Dest Opts; Dest Opts; Len = 0
    IP6: Version = 6 (0x6)
    IP6: Traffic Class = 0 (0x0)
    IP6: Flow Label = 0 (0x0)
    IP6: Payload Length = 40 (0x28)
    IP6: Next Header = 60 (Destination Options Header)
    IP6: Hop Limit = 64 (0x40)
    IP6: Source Address = 3ffe:3103:0:9:280:c7ff:fe2c:eeb7
    IP6: Destination Address = 3ffe:3103:0:8::3
    IP6: Destination Options Header
        IP6: Next Header = 60 (Destination Options Header)
        IP6: Length = 2 (0x2)
        IP6: Padding (4 bytes)
            IP6: Type = 1 (PadN)
                IP6: 00...... = Skip option if not recognized
                IP6: ..0..... = Option data does not change enroute
            IP6: Length = 2 (0x2)
            IP6: Data = 00 00
        IP6: Home Address Option
            IP6: Type = 201
                IP6: 11...... = Discard packet if not recognized, and send ICMP if not multicast
                IP6: ..0..... = Option data does not change enroute
            IP6: Length = 16 (0x10)
            IP6: Home Address = 3ffe:3103:0:8:280:c7ff:fe2c:eeb7
    IP6: Destination Options Header
        IP6: Next Header = 59 (None)
        IP6: Length = 1 (0x1)
        IP6: Padding (4 bytes)
            IP6: Type = 1 (PadN)
                IP6: 00...... = Skip option if not recognized
                IP6: ..0..... = Option data does not change enroute
            IP6: Length = 2 (0x2)
            IP6: Data = 00 00
        IP6: Binding Update Option
            IP6: Type = 198
                IP6: 11...... = Discard packet if not recognized, and send ICMP if not multicast
                IP6: ..0..... = Option data does not change enroute
            IP6: Length = 8 (0x8)
            IP6: Flags
                IP6: 1....... = Request a binding acknowledgement response.
                IP6: .1...... = Request host to act as home agent.
            IP6: Prefix Length = 64 (0x40)
            IP6: Sequence Number = 19 (0x13)
            IP6: Lifetime = 300 (0x12C)
```

BA: HA to MN

```
ETHERNET: ETYPE = 0x86DD : Protocol = Unknown
IP6: Routing (1 left of 1); Dest Opts; Len = 0
    IP6: Version = 6 (0x6)
    IP6: Traffic Class = 0 (0x0)
    IP6: Flow Label = 0 (0x0)
    IP6: Payload Length = 40 (0x28)
```

```
    IP6: Next Header = 43 (Routing Header)
    IP6: Hop Limit = 255 (0xFF)
    IP6: Source Address = 3ffe:3103:0:8::3
    IP6: Destination Address = 3ffe:3103:0:9:280:c7ff:fe2c:eeb7
    IP6: Routing Header
        IP6: Next Header = 60 (Destination Options Header)
        IP6: Length = 2 (0x2)
        IP6: Type = 0 (0x0)
        IP6: Segments Left = 1 (0x1)
        IP6: Reserved
        IP6: Route
            IP6: Address = 3ffe:3103:0:8:280:c7ff:fe2c:eeb7
    IP6: Destination Options Header
        IP6: Next Header = 59 (None)
        IP6: Length = 1 (0x1)
        IP6: Padding (1 bytes)
            IP6: Type = 0 (Pad1)
                IP6: 00...... = Skip option if not recognized
                IP6: ..0..... = Option data does not change enroute
        IP6: Binding Acknowledgement Option
            IP6: Type = 7
                IP6: 00...... = Skip option if not recognized
                IP6: ..0..... = Option data does not change enroute
            IP6: Length = 11 (0xB)
            IP6: Status: 0 = Binding accepted.
            IP6: Sequence Number = 19 (0x13)
            IP6: Lifetime = 300 (0x12C)
            IP6: Refresh = 240 (0xF0)
```

IPv6/IPv6 encapsulation: HA to MN

```
ETHERNET: ETYPE = 0x86DD : Protocol = Unknown
IP6: Proto = 0x29; Len = 104
    IP6: Version = 6 (0x6)
    IP6: Traffic Class = 0 (0x0)
    IP6: Flow Label = 0 (0x0)
    IP6: Payload Length = 104 (0x68)
    IP6: Next Header = 41
    IP6: Hop Limit = 255 (0xFF)
    IP6: Source Address = 3ffe:3103:0:8::3
    IP6: Destination Address = 3ffe:3103:0:9:280:c7ff:fe2c:eeb7
    IP6: Payload: Number of data bytes remaining = 104 (0x0068)
IP6: Proto = ICMP6; Len = 64
    IP6: Version = 6 (0x6)
    IP6: Traffic Class = 0 (0x0)
    IP6: Flow Label = 0 (0x0)
    IP6: Payload Length = 64 (0x40)
    IP6: Next Header = 58 (ICMP6)
    IP6: Hop Limit = 62 (0x3E)
    IP6: Source Address = 3ffe:3103:0:10:201:2ff:fe2a:93b9
    IP6: Destination Address = 3ffe:3103:0:8:280:c7ff:fe2c:eeb7
    IP6: Payload: Number of data bytes remaining = 64 (0x0040)
ICMP6: Echo Request; ID = 15628, Seq = 4864
    ICMP6: Checksum = 0x66AE
    ICMP6: Type = 128 (Echo Request)
    ICMP6: Code = 0 (0x0)
    ICMP6: Identifier = 15628 (0x3D0C)
    ICMP6: Sequence Number = 4864 (0x1300)
    ICMP6: Data: Number of data bytes remaining = 56 (0x0038)
```

BU: MN to CN

```
ETHERNET: ETYPE = 0x86DD : Protocol = Unknown
IP6: Dest Opts; Dest Opts; Len = 0
    IP6: Version = 6 (0x6)
    IP6: Traffic Class = 0 (0x0)
    IP6: Flow Label = 0 (0x0)
    IP6: Payload Length = 40 (0x28)
    IP6: Next Header = 60 (Destination Options Header)
    IP6: Hop Limit = 64 (0x40)
    IP6: Source Address = 3ffe:3103:0:9:280:c7ff:fe2c:eeb7
    IP6: Destination Address = 3ffe:3103:0:10:201:2ff:fe2a:93b9
    IP6: Destination Options Header
        IP6: Next Header = 60 (Destination Options Header)
        IP6: Length = 2 (0x2)
        IP6: Padding (4 bytes)
            IP6: Type = 1 (PadN)
                IP6: 00...... = Skip option if not recognized
                IP6: ..0..... = Option data does not change enroute
            IP6: Length = 2 (0x2)
            IP6: Data = 00 00
        IP6: Home Address Option
            IP6: Type = 201
                IP6: 11...... = Discard packet if not recognized, and send ICMP if not multicast
                IP6: ..0..... = Option data does not change enroute
            IP6: Length = 16 (0x10)
            IP6: Home Address = 3ffe:3103:0:8:280:c7ff:fe2c:eeb7
    IP6: Destination Options Header
        IP6: Next Header = 59 (None)
        IP6: Length = 1 (0x1)
        IP6: Padding (4 bytes)
            IP6: Type = 1 (PadN)
                IP6: 00...... = Skip option if not recognized
                IP6: ..0..... = Option data does not change enroute
            IP6: Length = 2 (0x2)
            IP6: Data = 00 00
        IP6: Binding Update Option
            IP6: Type = 198
                IP6: 11...... = Discard packet if not recognized, and send ICMP if not multicast
                IP6: ..0..... = Option data does not change enroute
```

```
            IP6: Length = 8 (0x8)
            IP6: Flags
                IP6: 1....... = Request a binding acknowledgement response.
            IP6: Prefix Length = 64 (0x40)
            IP6: Sequence Number = 20 (0x14)
            IP6: Lifetime = 300 (0x12C)
```

PING: CN to MN

```
ETHERNET: ETYPE = 0x86DD : Protocol = Unknown
IP6: Routing (1 left of 1); Proto = ICMP6; Len = 64
    IP6: Version = 6 (0x6)
    IP6: Traffic Class = 0 (0x0)
    IP6: Flow Label = 0 (0x0)
    IP6: Payload Length = 88 (0x58)
    IP6: Next Header = 43 (Routing Header)
    IP6: Hop Limit = 63 (0x3F)
    IP6: Source Address = 3ffe:3103:0:10:201:2ff:fe2a:93b9
    IP6: Destination Address = 3ffe:3103:0:9:280:c7ff:fe2c:eeb7
    IP6: Routing Header
        IP6: Next Header = 58 (ICMP6)
        IP6: Length = 2 (0x2)
        IP6: Type = 0 (0x0)
        IP6: Segments Left = 1 (0x1)
        IP6: Reserved
        IP6: Route
            IP6: Address = 3ffe:3103:0:8:280:c7ff:fe2c:eeb7
    IP6: Payload: Number of data bytes remaining = 64 (0x0040)
ICMP6: Echo Request; ID = 15628, Seq = 5120
    ICMP6: Checksum = 0x5CAE (ERROR: should be 0x5CAD)
    ICMP6: Type = 128 (Echo Request)
    ICMP6: Code = 0 (0x0)
    ICMP6: Identifier = 15628 (0x3D0C)
    ICMP6: Sequence Number = 5120 (0x1400)
    ICMP6: Data: Number of data bytes remaining = 56 (0x0038)
```

PING: MN to CN

```
ETHERNET: ETYPE = 0x86DD : Protocol = Unknown
IP6: Dest Opts; Proto = ICMP6; Len = 64
    IP6: Version = 6 (0x6)
    IP6: Traffic Class = 0 (0x0)
    IP6: Flow Label = 0 (0x0)
    IP6: Payload Length = 88 (0x58)
    IP6: Next Header = 60 (Destination Options Header)
    IP6: Hop Limit = 64 (0x40)
    IP6: Source Address = 3ffe:3103:0:9:280:c7ff:fe2c:eeb7
    IP6: Destination Address = 3ffe:3103:0:10:201:2ff:fe2a:93b9
    IP6: Destination Options Header
        IP6: Next Header = 58 (ICMP6)
        IP6: Length = 2 (0x2)
        IP6: Padding (4 bytes)
            IP6: Type = 1 (PadN)
                IP6: 00...... = Skip option if not recognized
                IP6: ..0..... = Option data does not change enroute
            IP6: Length = 2 (0x2)
            IP6: Data = 00 00
        IP6: Home Address Option
            IP6: Type = 201
                IP6: 11...... = Discard packet if not recognized, and send ICMP if not multicast
                IP6: ..0..... = Option data does not change enroute
            IP6: Length = 16 (0x10)
            IP6: Home Address = 3ffe:3103:0:8:280:c7ff:fe2c:eeb7
    IP6: Payload: Number of data bytes remaining = 64 (0x0040)
ICMP6: Echo Reply; ID = 15628, Seq = 5120
    ICMP6: Checksum = 0x5BAE (ERROR: should be 0x5BAD)
    ICMP6: Type = 129 (Echo Reply)
    ICMP6: Code = 0 (0x0)
    ICMP6: Identifier = 15628 (0x3D0C)
    ICMP6: Sequence Number = 5120 (0x1400)
    ICMP6: Data: Number of data bytes remaining = 56 (0x0038)
```

6.2.6.2.3 Linux HA; Windows MN2; Linux MN1

Finally, the last combinations with the Linux HA (excluding, of course, the CN and MN being Linux), the scenario where both PCs, Windows and Linux, are MNs. This scenario joins both functionalities previously tested.

Thus, the Figure 6-25 shows the scenario when two MNs are present, where the MN1 is a Linux PC and the MN2 is a Windows. Firstly, the MNs start in its HN (a), and then the MN2 moves to the FN2 (b). Later, the MN1 moves to the FN1, being both acting as a MN in different FNs (c).

The configurations of every entity can be seen on Annexes 6.3.2.7, 6.3.2.4 and 6.3.2.6, for Linux HA, Windows MN2 and Linux MN1 configurations, respectively.

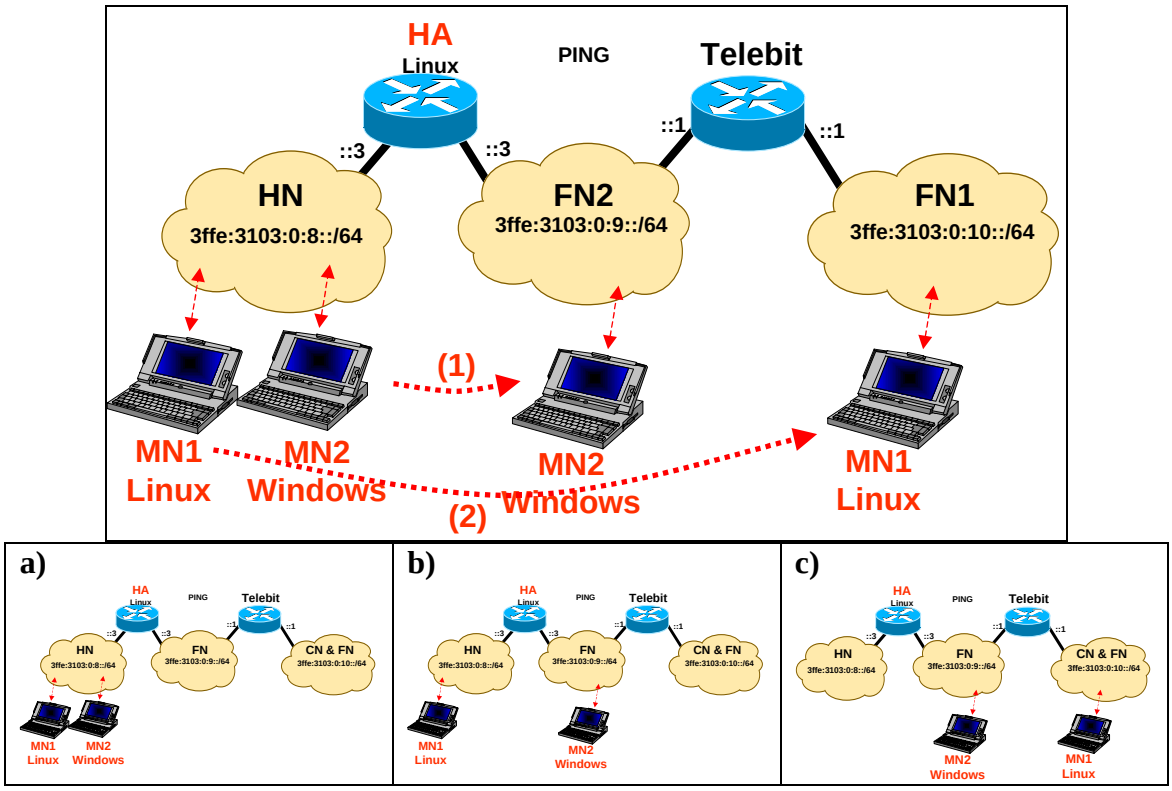


Figure 6-25: Linux HA; Windows MN2; Linux MN1.

Technical characteristics of the equipment used

Equipment Characteristics	HA	MN1	MN2	FN Router
Platform	*i386*	*i386*	*i386*	*Telebit*
Model	*"Blank" PC*	*"Blank" PC*	*"Blank" PC*	*AXI 462/3*
CPU	*Pentium II 400 MHz (cache 512 KB)*	*Pentium III 650 MHz (cache 256 KB)*	*Pentium II 266 MHz (cache 512 KB)*	*RISC (TBC 103)*
Memory	*64 RAM (swap 256 MB)*	*128 RAM (swap 256 MB)*	*64 RAM*	*16 MB*
Network Interfaces Card	*2 Ethernet cards 3COM series 59x (driver 3c59x)*	*1 Ethernet card 3COM series 59x (driver 3c59x)*	*1 Ethernet card 3COM series 59x*	*2 Ethernet cards (TCB 120) Motorola Power PC 750*
Operating System / Kernel	*Linux 7.2 (kernel 2.4.16)*	*Linux 7.2 (kernel 2.4.16)*	*Windows 2000*	*Telebit OS Swtbcpaxos 2.6a14*

Most significant sniffed packets

PING: MN1 to MN2

```
ETHERNET: ETYPE = 0x86DD : Protocol = Unknown
IP6: Dest Opts; Routing (1 left of 1); Proto = ICMP6; Len = 64
    IP6: Version = 6 (0x6)
    IP6: Traffic Class = 0 (0x0)
    IP6: Flow Label = 0 (0x0)
    IP6: Payload Length = 112 (0x70)
    IP6: Next Header = 60 (Destination Options Header)
    IP6: Hop Limit = 63 (0x3F)
    IP6: Source Address = 3ffe:3103:0:10:201:2ff:fe2a:93b9
    IP6: Destination Address = 3ffe:3103:0:9:280:c7ff:fe2c:eeb7
    IP6: Destination Options Header
        IP6: Next Header = 43 (Routing Header)
        IP6: Length = 2 (0x2)
        IP6: Padding (4 bytes)
            IP6: Type = 1 (PadN)
                IP6: 00...... = Skip option if not recognized
                IP6: ..0..... = Option data does not change enroute
            IP6: Length = 2 (0x2)
            IP6: Data = 00 00
        IP6: Home Address Option
            IP6: Type = 201
                IP6: 11...... = Discard packet if not recognized, and send ICMP if not multicast
                IP6: ..0..... = Option data does not change enroute
            IP6: Length = 16 (0x10)
            IP6: Home Address = 3ffe:3103:0:8:201:2ff:fe2a:93b9
    IP6: Routing Header
        IP6: Next Header = 58 (ICMP6)
        IP6: Length = 2 (0x2)
        IP6: Type = 0 (0x0)
        IP6: Segments Left = 1 (0x1)
        IP6: Reserved
        IP6: Route
            IP6: Address = 3ffe:3103:0:8:280:c7ff:fe2c:eeb7
    IP6: Payload: Number of data bytes remaining = 64 (0x0040)
ICMP6: Echo Request; ID = 47631, Seq = 47360
    ICMP6: Checksum = 0x0582 (ERROR: should be 0x0579)
    ICMP6: Type = 128 (Echo Request)
    ICMP6: Code = 0 (0x0)
    ICMP6: Identifier = 47631 (0xBA0F)
    ICMP6: Sequence Number = 47360 (0xB900)
    ICMP6: Data: Number of data bytes remaining = 56 (0x0038)
```

PING: MN2 to MN1

```
ETHERNET: ETYPE = 0x86DD : Protocol = Unknown
IP6: Routing (1 left of 1); Dest Opts; Proto = ICMP6; Len = 64
    IP6: Version = 6 (0x6)
    IP6: Traffic Class = 0 (0x0)
    IP6: Flow Label = 0 (0x0)
    IP6: Payload Length = 112 (0x70)
    IP6: Next Header = 43 (Routing Header)
    IP6: Hop Limit = 64 (0x40)
    IP6: Source Address = 3ffe:3103:0:9:280:c7ff:fe2c:eeb7
    IP6: Destination Address = 3ffe:3103:0:10:201:2ff:fe2a:93b9
```

```
    IP6: Routing Header
        IP6: Next Header = 60 (Destination Options Header)
        IP6: Length = 2 (0x2)
        IP6: Type = 0 (0x0)
        IP6: Segments Left = 1 (0x1)
        IP6: Reserved
        IP6: Route
            IP6: Address = 3ffe:3103:0:8:201:2ff:fe2a:93b9
    IP6: Destination Options Header
        IP6: Next Header = 58 (ICMP6)
        IP6: Length = 2 (0x2)
        IP6: Padding (4 bytes)
            IP6: Type = 1 (PadN)
                IP6: 00...... = Skip option if not recognized
                IP6: ..0..... = Option data does not change enroute
            IP6: Length = 2 (0x2)
            IP6: Data = 00 00
        IP6: Home Address Option
            IP6: Type = 201
                IP6: 11...... = Discard packet if not recognized, and send ICMP if not multicast
                IP6: ..0..... = Option data does not change enroute
            IP6: Length = 16 (0x10)
            IP6: Home Address = 3ffe:3103:0:8:280:c7ff:fe2c:eeb7
    IP6: Payload: Number of data bytes remaining = 64 (0x0040)
ICMP6: Echo Reply; ID = 47631, Seq = 47360
    ICMP6: Checksum = 0x0482 (ERROR: should be 0x0479)
    ICMP6: Type = 129 (Echo Reply)
    ICMP6: Code = 0 (0x0)
    ICMP6: Identifier = 47631 (0xBA0F)
    ICMP6: Sequence Number = 47360 (0xB900)
    ICMP6: Data: Number of data bytes remaining = 56 (0x0038)
```

6.2.6.2.4 Conclusions

Tacking into account that everything works on the first scenario, it is possible to identify some kinds of compatibilities between the implementations and the draft versions. These compatibilities are just the ones referred to the interaction between the Linux MN and the Windows CN (because there is no interaction between a CN and the HA). It can be seen, that the CN understood perfectly the Ha (Destination Option) sent by the MN when it moved to the FN. This is clear because the echo replays achieved were considered as the answers to the Echo Requests. Furthermore, we noticed that the BU sent by the MN to the CN was already understood, firstly because the binding cache reflected that fact, and secondly, because the next Echo Requests were sent to the MN using the appropriate mobility mechanism; i.e. using a RH.

As well as for the first scenario, also in the second everything works. So, it is possible to identify more interoperations beyond the ones identified above about the Ha and the BU. In this case, the MN must already interact with the HA. And what can be seen is that the Windows MN is able to register their Ha in the HA, and the latter send back an Acknowledgement to confirm that fact. On the other hand, the same happen when this registration is requested to the CN. The Windows CN add an entry in its binding cache and start to send directly the packets using RHs. The other interaction is when the HA encapsulates the packet IPv6 over IPv6, and redirect the packet to the MN. This part is already well understood by the Windows MN, so the compatibility is clear.

Finally and, as expected, in the last scenario every entity involved is working correctly. Just through this scenario is possible to show all the functionalities working correctly, because a complete compatibility is necessary in order get this scenario running.

We can conclude that it can be achieved a complete interoperability of MNs and CNs, using a Linux HA (without security mechanisms, remember).

6.2.6.3 Cisco HA; Linux and Windows (MN and CN)

The tests described in this section were performed using a Cisco HA, being the MNs and CNs, either a Windows or a Linux PCs. As in the previous section, the tests involve two different implementations and also two different versions of the MIPv6 draft (15 and 13). Here, is intended to check how much compatibility we can achieve with Windows and Linux using a Cisco HA.

In order to understand the degree of compatibility, some scenarios will be analysed, covering all the possibilities.

6.2.6.3.1 Cisco HA; Windows MN; Linux CN

The first scenario is shown in Figure 6-26. It includes the Cisco HA, a Linux CN and a Windows MN. This should be the easier scenario to achieve compatibility between the Cisco HA and Linux, because the CN functionality is a less complex task in mobility. On the other hand, as we concluded before, there are compatibility between MN and CN functionalities between the Windows and Linux platforms.

In this scenario the MN is started in its HN (a), after that it is moved to the FN (b), and finally it is moved back to its HN (c).

The configurations of every entity can be seen on Annexes 6.3.2, 6.3.2.3 and 6.3.2.5, for Cisco HA, Windows MN and Linux CN configurations, respectively.

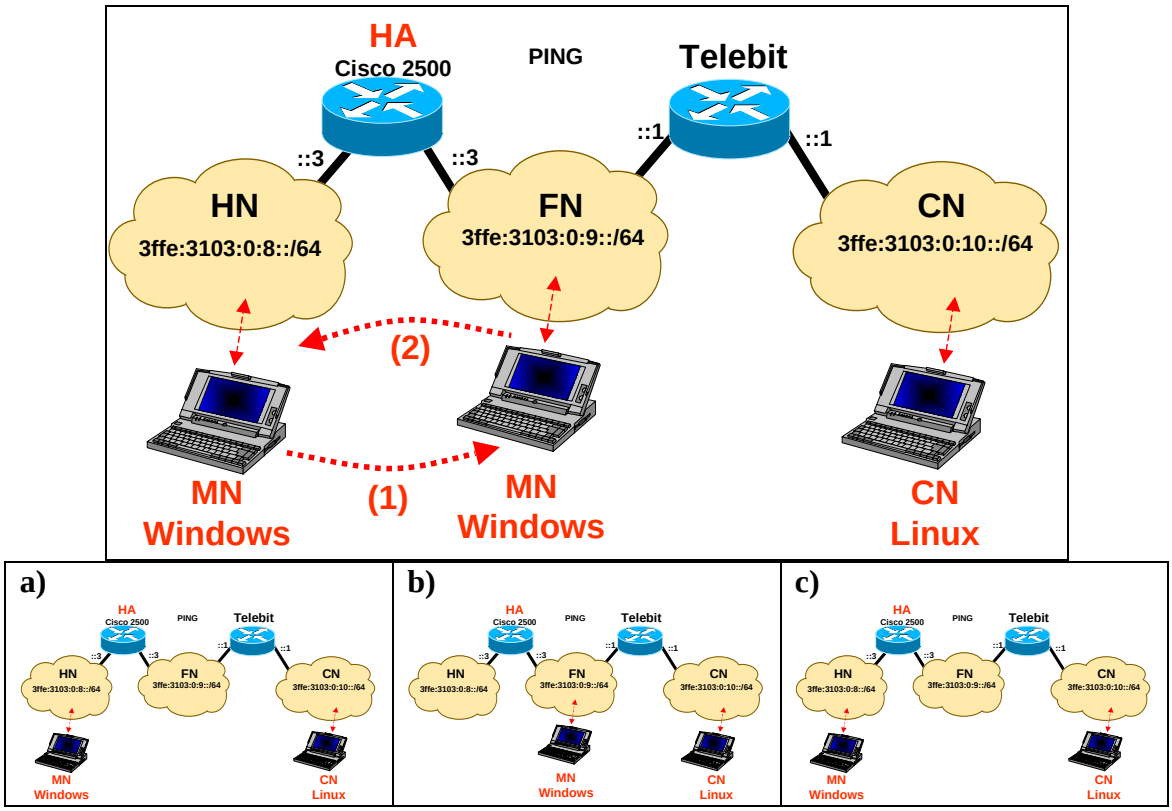


Figure 6-26: Cisco HA; Windows MN; Linux CN.

Technical characteristics of the equipment used

Equipment Characteristics	HA	CN	MN	FN Router
Platform	*Cisco 3600*	*i386*	*i386*	*Telebit*
Model	*Cisco 3640*	*"Blank" PC*	*"Blank" PC*	*AXI 462/3*
CPU	*R4700 100 MHz*	*Pentium III 650 MHz (cache 256 KB)*	*Pentium II 266 MHz (cache 512 KB)*	*RISC (TBC 103)*
Memory	*30 MB*	*128 RAM (swap 256 MB)*	*64 RAM*	*16 MB*
Network Interfaces Card	*2 FastEthernet*	*1 Ethernet card 3COM series 59x (driver 3c59x)*	*1 Ethernet card 3COM series 59x*	*2 Ethernet cards (TCB 120) Motorola Power PC 750*
Operating System / Kernel	*Cisco IOS experimental 12.2(20010831:093907)*	*Linux 7.2 (kernel 2.4.16)*	*Windows 2000*	*Telebit OS Swtbcpaxos 2.6a14*

Most significant sniffed packets

BU: MN to HA

```
ETHERNET: ETYPE = 0x86DD : Protocol = Unknown
IP6: Dest Opts; Dest Opts; Len = 0
    IP6: Version = 6 (0x6)
    IP6: Traffic Class = 0 (0x0)
    IP6: Flow Label = 0 (0x0)
    IP6: Payload Length = 40 (0x28)
    IP6: Next Header = 60 (Destination Options Header)
    IP6: Hop Limit = 64 (0x40)
    IP6: Source Address = 3ffe:3103:0:9:280:c7ff:fe2c:eeb7
    IP6: Destination Address = 3ffe:3103:0:8::3
    IP6: Destination Options Header
        IP6: Next Header = 60 (Destination Options Header)
        IP6: Length = 2 (0x2)
        IP6: Padding (4 bytes)
            IP6: Type = 1 (PadN)
                IP6: 00...... = Skip option if not recognized
                IP6: ..0..... = Option data does not change enroute
            IP6: Length = 2 (0x2)
            IP6: Data = 00 00
        IP6: Home Address Option
            IP6: Type = 201
                IP6: 11...... = Discard packet if not recognized, and send ICMP if not multicast
                IP6: ..0..... = Option data does not change enroute
            IP6: Length = 16 (0x10)
            IP6: Home Address = 3ffe:3103:0:8:280:c7ff:fe2c:eeb7
    IP6: Destination Options Header
        IP6: Next Header = 59 (None)
        IP6: Length = 1 (0x1)
        IP6: Padding (4 bytes)
            IP6: Type = 1 (PadN)
                IP6: 00...... = Skip option if not recognized
                IP6: ..0..... = Option data does not change enroute
            IP6: Length = 2 (0x2)
            IP6: Data = 00 00
        IP6: Binding Update Option
            IP6: Type = 198
                IP6: 11...... = Discard packet if not recognized, and send ICMP if not multicast
                IP6: ..0..... = Option data does not change enroute
            IP6: Length = 8 (0x8)
            IP6: Flags
                IP6: 1....... = Request a binding acknowledgement response.
                IP6: .1...... = Request host to act as home agent.
            IP6: Prefix Length = 64 (0x40)
            IP6: Sequence Number = 19 (0x13)
            IP6: Lifetime = 300 (0x12C)
```

BA: HA to MN

```
ETHERNET: ETYPE = 0x86DD : Protocol = Unknown
IP6: Routing (1 left of 1); Dest Opts; Len = 0
    IP6: Version = 6 (0x6)
    IP6: Traffic Class = 0 (0x0)
    IP6: Flow Label = 0 (0x0)
    IP6: Payload Length = 40 (0x28)
    IP6: Next Header = 43 (Routing Header)
    IP6: Hop Limit = 255 (0xFF)
    IP6: Source Address = 3ffe:3103:0:8::3
    IP6: Destination Address = 3ffe:3103:0:9:280:c7ff:fe2c:eeb7
    IP6: Routing Header
        IP6: Next Header = 60 (Destination Options Header)
        IP6: Length = 2 (0x2)
        IP6: Type = 0 (0x0)
        IP6: Segments Left = 1 (0x1)
        IP6: Reserved
        IP6: Route
            IP6: Address = 3ffe:3103:0:8:280:c7ff:fe2c:eeb7
    IP6: Destination Options Header
        IP6: Next Header = 59 (None)
        IP6: Length = 1 (0x1)
        IP6: Padding (1 bytes)
            IP6: Type = 0 (Pad1)
                IP6: 00...... = Skip option if not recognized
                IP6: ..0..... = Option data does not change enroute
        IP6: Binding Acknowledgement Option
            IP6: Type = 7
                IP6: 00...... = Skip option if not recognized
                IP6: ..0..... = Option data does not change enroute
            IP6: Length = 11 (0xB)
            IP6: Status: 0 = Binding accepted.
            IP6: Sequence Number = 19 (0x13)
            IP6: Lifetime = 300 (0x12C)
            IP6: Refresh = 240 (0xF0)
```

IPv6/IPv6 encapsulation: HA to MN

```
ETHERNET: ETYPE = 0x86DD : Protocol = Unknown
IP6: Proto = 0x29; Len = 104
    IP6: Version = 6 (0x6)
    IP6: Traffic Class = 0 (0x0)
    IP6: Flow Label = 0 (0x0)
    IP6: Payload Length = 104 (0x68)
    IP6: Next Header = 41
    IP6: Hop Limit = 255 (0xFF)
    IP6: Source Address = 3ffe:3103:0:8::3
    IP6: Destination Address = 3ffe:3103:0:9:280:c7ff:fe2c:eeb7
    IP6: Payload: Number of data bytes remaining = 104 (0x0068)
IP6: Proto = ICMP6; Len = 64
    IP6: Version = 6 (0x6)
    IP6: Traffic Class = 0 (0x0)
    IP6: Flow Label = 0 (0x0)
    IP6: Payload Length = 64 (0x40)
    IP6: Next Header = 58 (ICMP6)
    IP6: Hop Limit = 62 (0x3E)
    IP6: Source Address = 3ffe:3103:0:10:201:2ff:fe2a:93b9
    IP6: Destination Address = 3ffe:3103:0:8:280:c7ff:fe2c:eeb7
    IP6: Payload: Number of data bytes remaining = 64 (0x0040)
ICMP6: Echo Request; ID = 15628, Seq = 4864
    ICMP6: Checksum = 0x66AE
    ICMP6: Type = 128 (Echo Request)
    ICMP6: Code = 0 (0x0)
    ICMP6: Identifier = 15628 (0x3D0C)
    ICMP6: Sequence Number = 4864 (0x1300)
    ICMP6: Data: Number of data bytes remaining = 56 (0x0038)
```

BU: MN to CN

```
ETHERNET: ETYPE = 0x86DD : Protocol = Unknown
IP6: Dest Opts; Dest Opts; Len = 0
    IP6: Version = 6 (0x6)
    IP6: Traffic Class = 0 (0x0)
    IP6: Flow Label = 0 (0x0)
    IP6: Payload Length = 40 (0x28)
    IP6: Next Header = 60 (Destination Options Header)
    IP6: Hop Limit = 64 (0x40)
    IP6: Source Address = 3ffe:3103:0:9:280:c7ff:fe2c:eeb7
    IP6: Destination Address = 3ffe:3103:0:10:201:2ff:fe2a:93b9
    IP6: Destination Options Header
        IP6: Next Header = 60 (Destination Options Header)
        IP6: Length = 2 (0x2)
        IP6: Padding (4 bytes)
            IP6: Type = 1 (PadN)
                IP6: 00...... = Skip option if not recognized
                IP6: ..0..... = Option data does not change enroute
            IP6: Length = 2 (0x2)
            IP6: Data = 00 00
        IP6: Home Address Option
            IP6: Type = 201
                IP6: 11...... = Discard packet if not recognized, and send ICMP if not multicast
                IP6: ..0..... = Option data does not change enroute
            IP6: Length = 16 (0x10)
            IP6: Home Address = 3ffe:3103:0:8:280:c7ff:fe2c:eeb7
    IP6: Destination Options Header
        IP6: Next Header = 59 (None)
        IP6: Length = 1 (0x1)
        IP6: Padding (4 bytes)
            IP6: Type = 1 (PadN)
                IP6: 00...... = Skip option if not recognized
                IP6: ..0..... = Option data does not change enroute
```

```
            IP6: Length = 2 (0x2)
            IP6: Data = 00 00
        IP6: Binding Update Option
            IP6: Type = 198
                IP6: 11...... = Discard packet if not recognized, and send ICMP if not multicast
                IP6: ..0..... = Option data does not change enroute
            IP6: Length = 8 (0x8)
            IP6: Flags
                IP6: 1....... = Request a binding acknowledgement response.
            IP6: Prefix Length = 64 (0x40)
            IP6: Sequence Number = 20 (0x14)
            IP6: Lifetime = 300 (0x12C)
```

PING: CN to MN

```
ETHERNET: ETYPE = 0x86DD : Protocol = Unknown
IP6: Routing (1 left of 1); Proto = ICMP6; Len = 64
    IP6: Version = 6 (0x6)
    IP6: Traffic Class = 0 (0x0)
    IP6: Flow Label = 0 (0x0)
    IP6: Payload Length = 88 (0x58)
    IP6: Next Header = 43 (Routing Header)
    IP6: Hop Limit = 63 (0x3F)
    IP6: Source Address = 3ffe:3103:0:10:201:2ff:fe2a:93b9
    IP6: Destination Address = 3ffe:3103:0:9:280:c7ff:fe2c:eeb7
    IP6: Routing Header
        IP6: Next Header = 58 (ICMP6)
        IP6: Length = 2 (0x2)
        IP6: Type = 0 (0x0)
        IP6: Segments Left = 1 (0x1)
        IP6: Reserved
        IP6: Route
            IP6: Address = 3ffe:3103:0:8:280:c7ff:fe2c:eeb7
    IP6: Payload: Number of data bytes remaining = 64 (0x0040)
ICMP6: Echo Request; ID = 15628, Seq = 5120
    ICMP6: Checksum = 0x5CAE (ERROR: should be 0x5CAD)
    ICMP6: Type = 128 (Echo Request)
    ICMP6: Code = 0 (0x0)
    ICMP6: Identifier = 15628 (0x3D0C)
    ICMP6: Sequence Number = 5120 (0x1400)
    ICMP6: Data: Number of data bytes remaining = 56 (0x0038)
```

PING: MN to CN

```
ETHERNET: ETYPE = 0x86DD : Protocol = Unknown
IP6: Dest Opts; Proto = ICMP6; Len = 64
    IP6: Version = 6 (0x6)
    IP6: Traffic Class = 0 (0x0)
    IP6: Flow Label = 0 (0x0)
    IP6: Payload Length = 88 (0x58)
    IP6: Next Header = 60 (Destination Options Header)
    IP6: Hop Limit = 64 (0x40)
    IP6: Source Address = 3ffe:3103:0:9:280:c7ff:fe2c:eeb7
    IP6: Destination Address = 3ffe:3103:0:10:201:2ff:fe2a:93b9
    IP6: Destination Options Header
        IP6: Next Header = 58 (ICMP6)
        IP6: Length = 2 (0x2)
        IP6: Padding (4 bytes)
            IP6: Type = 1 (PadN)
                IP6: 00...... = Skip option if not recognized
                IP6: ..0..... = Option data does not change enroute
            IP6: Length = 2 (0x2)
            IP6: Data = 00 00
        IP6: Home Address Option
            IP6: Type = 201
                IP6: 11...... = Discard packet if not recognized, and send ICMP if not multicast
                IP6: ..0..... = Option data does not change enroute
            IP6: Length = 16 (0x10)
            IP6: Home Address = 3ffe:3103:0:8:280:c7ff:fe2c:eeb7
    IP6: Payload: Number of data bytes remaining = 64 (0x0040)
ICMP6: Echo Reply; ID = 15628, Seq = 5120
    ICMP6: Checksum = 0x5BAE (ERROR: should be 0x5BAD)
    ICMP6: Type = 129 (Echo Reply)
    ICMP6: Code = 0 (0x0)
    ICMP6: Identifier = 15628 (0x3D0C)
    ICMP6: Sequence Number = 5120 (0x1400)
    ICMP6: Data: Number of data bytes remaining = 56 (0x0038)
```

6.2.6.3.2 Cisco HA; Linux MN; Windows CN

The second scenario is shown in Figure 6-27. It includes the Cisco HA, a Linux CN and a Windows MN. This scenario will really check the compatibility no just between the Linux MN and the Windows CN but also, and most important, between the Linux MN and the Cisco HA.

In this scenario the MN is started in its HN (a), after that it is moved to the FN (b), and finally is moved back to its HN (c).

The configurations of every entity can be seen on Annexes 6.3.2, 6.3.2.6 and 6.3.2.2, for Cisco HA, Linux MN and Windows CN configurations, respectively.

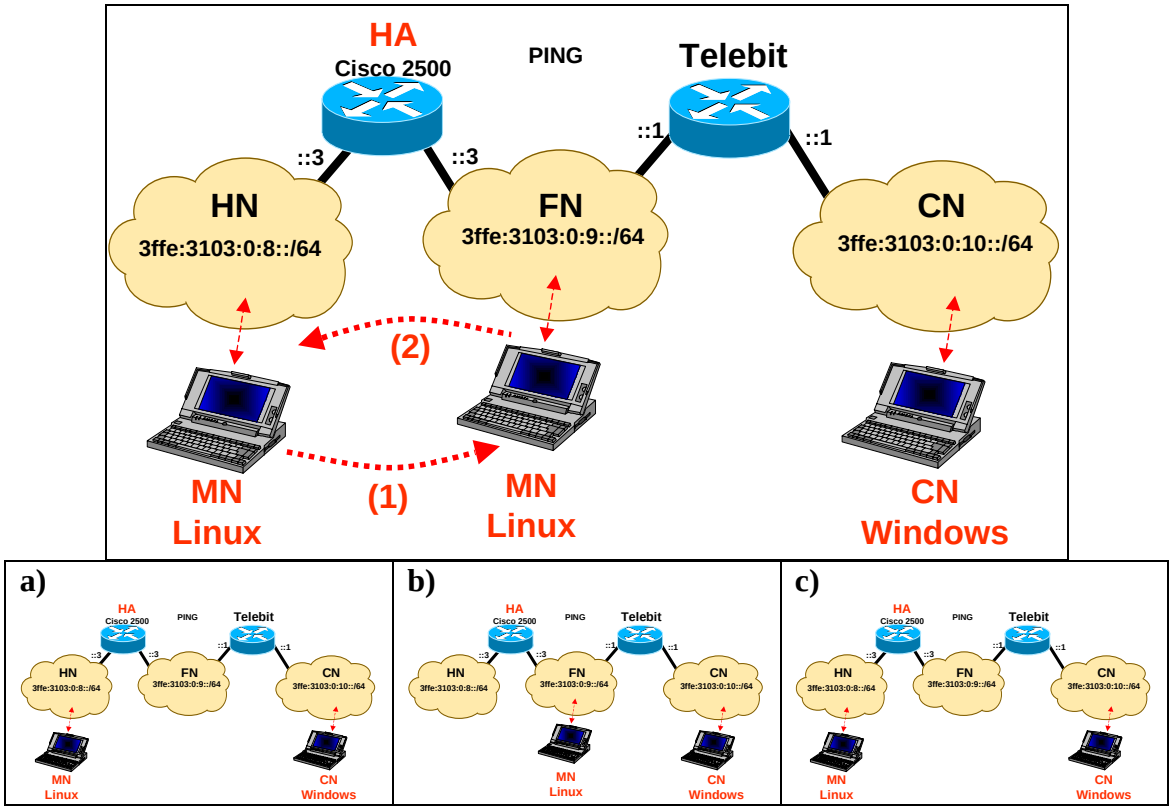


Figure 6-27: Cisco HA; Linux MN; Windows CN.

Technical characteristics of the equipment used

Equipment Characteristics	HA	MN	CN	FN Router
Platform	*Cisco 3600*	*i386*	*I386*	*Telebit*
Model	*Cisco 3640*	*"Blank" PC*	*"Blank" PC*	*AXI 462/3*
CPU	*R4700 100 MHz*	*Pentium III 650 MHz (cache 256 KB)*	*Pentium II 266 MHz (cache 512 KB)*	*RISC (TBC 103)*
Memory	*30 MB*	*128 RAM (swap 256 MB)*	*64 RAM*	*16 MB*
Network Interfaces Card	*2 FastEthernet*	*1 Ethernet card 3COM series 59x (driver 3c59x)*	*1 Ethernet card 3COM series 59x*	*2 Ethernet cards (TCB 120) Motorola Power PC 750*
Operating System / Kernel	*Cisco IOS experimental 12.2(20010831:093907)*	*Linux 7.2 (kernel 2.4.16)*	*Windows 2000*	*Telebit OS Swtbcpaxos 2.6a14*

Most significant sniffed packets

BU: MN to HA

```
ETHERNET: ETYPE = 0x86DD : Protocol = Unknown
IP6: Dest Opts; Dest Opts; Len = 0
    IP6: Version = 6 (0x6)
    IP6: Traffic Class = 0 (0x0)
    IP6: Flow Label = 0 (0x0)
    IP6: Payload Length = 40 (0x28)
    IP6: Next Header = 60 (Destination Options Header)
    IP6: Hop Limit = 254 (0xFE)
    IP6: Source Address = 3ffe:3103:0:9:201:2ff:fe2a:93b9
    IP6: Destination Address = 3ffe:3103:0:8::3
    IP6: Destination Options Header
        IP6: Next Header = 60 (Destination Options Header)
        IP6: Length = 2 (0x2)
        IP6: Padding (4 bytes)
            IP6: Type = 1 (PadN)
                IP6: 00...... = Skip option if not recognized
                IP6: ..0..... = Option data does not change enroute
            IP6: Length = 2 (0x2)
            IP6: Data = 00 00
        IP6: Home Address Option
            IP6: Type = 201
                IP6: 11...... = Discard packet if not recognized, and send ICMP if not multicast
                IP6: ..0..... = Option data does not change enroute
            IP6: Length = 16 (0x10)
            IP6: Home Address = 3ffe:3103:0:8:201:2ff:fe2a:93b9
    IP6: Destination Options Header
        IP6: Next Header = 59 (None)
        IP6: Length = 1 (0x1)
        IP6: Binding Update Option
            IP6: Type = 198
                IP6: 11...... = Discard packet if not recognized, and send ICMP if not multicast
                IP6: ..0..... = Option data does not change enroute
            IP6: Length = 8 (0x8)
            IP6: Flags
                IP6: 1....... = Request a binding acknowledgement response.
                IP6: .1...... = Request host to act as home agent.
                IP6: ..1..... = Binding update includes a care-of address.
            IP6: Prefix Length = 0 (0x0)
            IP6: Sequence Number = 7 (0x7)
            IP6: Lifetime = 120 (0x78)
            IP6: Care-of Address = 102:0:226e:8c00::5e00:0
```

* The line Care-of Address is a wrong value because the sniffer just understands draft13.

BA: HA to MN

```
ETHERNET: ETYPE = 0x86DD : Protocol = Unknown
```

```
IP6: Routing (1 left of 1); Dest Opts; Len = 0
    IP6: Version = 6 (0x6)
    IP6: Traffic Class = 0 (0x0)
    IP6: Flow Label = 0 (0x0)
    IP6: Payload Length = 40 (0x28)
    IP6: Next Header = 43 (Routing Header)
    IP6: Hop Limit = 64 (0x40)
    IP6: Source Address = 3ffe:3103:0:8::3
    IP6: Destination Address = 3ffe:3103:0:9:201:2ff:fe2a:93b9
    IP6: Routing Header
        IP6: Next Header = 60 (Destination Options Header)
        IP6: Length = 2 (0x2)
        IP6: Type = 0 (0x0)
        IP6: Segments Left = 1 (0x1)
        IP6: Reserved
        IP6: Route
            IP6: Address = 3ffe:3103:0:8:201:2ff:fe2a:93b9
    IP6: Destination Options Header
        IP6: Next Header = 59 (None)
        IP6: Length = 1 (0x1)
        IP6: Padding (1 bytes)
            IP6: Type = 0 (Pad1)
                IP6: 00...... = Skip option if not recognized
                IP6: ..0..... = Option data does not change enroute
        IP6: Binding Acknowledgement Option
            IP6: Type = 7
                IP6: 00...... = Skip option if not recognized
                IP6: ..0..... = Option data does not change enroute
            IP6: Length = 11 (0xB)
            IP6: Status: 136 = Binding rejected (incorrect interface identifier length).
            IP6: Sequence Number = 7 (0x7)
            IP6: Lifetime = 0 (0x0)
            IP6: Refresh = 0 (0x0)
```

6.2.6.3.3 Conclusions

Regarding to the first scenario, it worked correctly what means that there exists some compatibility. But with this scenario, it can just be concluded that a Linux CN is compatible with the Windows MN. Nothing can be concluded about the compatibility between the Cisco HA and Linux, because there is no any interaction. As said for the reverse case, the Linux CN understand the Ha Destination Option, accepts the registration of the MN in its binding cache, and sends directly the packets to the MN through a RH. So it can be concluded that exists that the reverse compatibility between the Linux and the Windows.

The second scenario tested didn't work correctly. The problem can be seen above on the packets captured with the sniffer. Just at the beginning, the BU is sent by the Linux MN to the Cisco HA and a binding with a negative acknowledge is sent back to the MN. The reason is *bad prefix length*, but the problem is actually that the HA is trying to read the prefix of the care-of address in the wrong place of the packet, due to the different versions implemented and the different location of the fields.

If you take a look to the BU which is not accepted, it has a flag saying that the BU includes the CoA, but the CoA is wrong and the prefix is set to 0; i.e. invalid (remember that the sniffer understands draft 13, and this is how the Cisco HA sees these fields). Is the latter one that is returned as an error from the HA, as can be seen on the BA sent.

Is also very interesting to notice that the Windows accepts these kind of BUs (and implements also the draft 13). The BUs were accepted when the HA was a Linux, the MN a Linux and the CN a Windows! The reason should be that the Windows doesn't recognise the flag that says if the CoA is the one included in the IPv6 source address or is specified on the BU. On the other hand, the Windows seems that don't verify if the prefix size is valid or not.

Anyway, these aspects that seems negatives gives to the Windows platform compatibility with the Linux one, even implementing different version of the mobility draft. In can be considered the possibility that the Windows implementation was modified in order to disregards these details to allow the interoperability with draft 15.

6.2.7 Conclusions

The IP mobility capabilities are very important in order to provide to the devices the necessary transparency to the upper layer protocols like TCP, UDP, and applications when changing their point of attachment to the network. This way, the movement of a Mobile Node (MN) between different links doesn't imply either a restart of a connection, or the reconfiguration of the network parameters. On the other hand, the node could be always contacted, thanks to the use of the same IPv6 address as the MN identification in each link. Actually, the mobility does not provide the "all-IP" capabilities at all. To implement this it is necessary to reduce considerably the handover times, which can be made using Fast-Handover mechanisms over the Mobility.

6.2.7.1 Linux tests

In this document we described the tests of one MIPv6 implementation, running on the free Linux platform. The MIPL (Mobile IPv6 for Linux) implementation is one of the most tested in the world right now, and offers an interesting set of functionalities based on the draft version number 15 (version 0.9.1).

This document has started talking about how the Mobile protocol, defined by the IETF *mobileip* Working Group, works. After that, a typical testbed used for the tests has been introduced, referring the main existent equipment and links. The tests have been separated in two main components: the first one where the handover was made with LAN technologies (Hubs), and the second one, where the handover was made using Wireless LAN (WLAN) technologies. For each component, five main scenarios were tested, representing the main situations of where IP Mobility can play an important role. From the situation where the MN is in the HN, to one where the MN is in a FN, or another where two MNs are present, and each one either in its HN or in a FN.

As expected, the main differences we achieved between the two sets of scenarios, were the handover times. In fact, these times are very different, being higher for the LAN technology. Of course, this is because the WLAN technology is faster than the LAN recovering the signal and starting the communications. In the LAN, when the cable is unplugged, the interface goes "down" and the restart is longer. Anyway, the approximated time interval for the handover presented, can hide the truth, because we just refer the time during which the application are not responding. However, in the LAN handover, some time should be subtracted in order to take in consideration the time in which we change physically the cable from one Hub to another.

In terms of functionality the tests performed, show that the implementation is working properly. However, the implementation still has some instability in some situations. Sometimes, after a period of tests, mobility seems that goes down, and in some cases the connection between the entities are not restarted, and is necessary to restart the mobility or even the machines in order to have them working properly again. For example, one thing that many times makes the mobility crash is when the handover is performed very fast consecutively. If the times that a MN is attached to each link is less than 4 - 5 seconds, usually the MN crash easier.

Finally, and despite some instability, we can conclude that in general, the Linux implementation is working correctly. Another interesting property, which is very important for researchers, is that this implementation follows very near the IETF evolutions.

6.2.7.2 Windows, Cisco and Linux tests

The interoperability tests performed show to us what are the implementation and the functionalities that can work together. The interoperability tests had since the beginning and extra-complexity, taking into account that not all the platforms implement the same draft version - Linux implements the draft 15 and, Windows and Cisco implement draft 13. Basically the tests were made covering almost all the possible combinations both between the platforms and the functionalities.

Firstly, we started the interoperability tests using the Cisco HA, and testing the Windows CNs and MNs. What we concluded was that in every situation the scenarios work as expected and the packet exchange were the correct. We can say that for the main functionalities at least, the implementations are completely interoperable. However, this is an expected result, taking into account that both implement the same draft version - 13.

On the second set of interoperability tests we performed used a Linux HA, and the CNs and MNs were either Linux or Windows. The results obtained was very interesting, because we concluded that even implementing different draft version, with considerable differences between of them, the implementations are compatibles in all circumstances. Of course, this result can only be achieved if we don't use the authentication mechanisms on the Linux (authentication is optional in draft 15).

Finally, we have tested a Cisco HA, with CNs and MNs, either been a Linux or Windows implementations. In this case, we noticed that not all the functionalities inter-operate ones with each others. What we concluded was that the Linux CN can work with Windows MN and Cisco HA, but the Linux MN doesn't work with the Cisco HA. This happen because the fields related to the BU header changed from the draft version 13 to 15, and the value of the prefix length are misinterpreted. We have already noticed that the Windows, implementing already the draft 13 interprets correctly the Linux BUs. This should be due to the fact that the Windows doesn't check some fields, or because some changes could be made in order to make the implementation draft15-compatible.

6.2.8 Future work

The future work to be made in this area will be to test the remaining main IP Mobility implementations. At the same time, the test of interoperability between them is very important in order to assess the always-important capacity of the different equipment to work together. So far, this is a very hard work, due to the large amount of draft versions, and the different stages of development done by the vendors. This work, can be perhaps more effective when the MIPv6 becomes a RFC, and the implementations converge to the same point.

Important issues for MIPv6 are other technologies, defined by the IETF that complements the main protocol. A good example are the Fast-Handover and the HMIPv6 implementations, which allow the protocol to minimise the handover time, being an aspect of critical importance in applications with real-time constraints, like voice, video, among others. The test of both implementation is a work that we intent to do in the near future.

On the other hand, IPv6 mobility will be a very powerful technology, supported by the evolution of the public wireless networks. However, it is necessary to integrate this technology with many others, making it inter-operate, and taking advantage of mixing technologies. One example can be the capacity to provide mobility facilities for VPN clients. This kind of mix scenarios are not too much explored, but could be extremely interesting in the future. Actually, the mobility facilities for VPN clients, is a work that we intend to perform in the near future.

Already very important and not very much studied yet, is the business model that the ISPs can use to provide mobility services to their clients. Usually the mobility tests are performed at a low scale, and in local networks. However, there is not clear how it can be made in a higher scale, and how the deployment of the different mobile entities can be. This is also a study that we will perform in the near times.

Also, the unavoidable billing issues are not very clear yet. The mobility will increase the difficulty to bill the customers, because it became very difficult to know the amount of bandwidth they are using.

Our intention is to continue to work in the Mobility area during this project in order to accomplish all the items above referred.

6.3 Appendix C: PTIN MIPv6 tests annexes

6.3.1 MIPL Tests

6.3.1.1 HA configurations

network-mip6.conf

```
# MIPL Mobile IPv6 Configuration file

# Should this node act as a home agent (ha), mobile node (mn) or
# correspondent node (cn). HA and MN both have CN functionality
# embedded. Default: cn.
FUNCTIONALITY=ha

# In error situations it may be desired to get more detailed
# information what is happening. Increase this value to get more
# messages from the module (default: 0).
DEBUGLEVEL=1

# Should unicasts to node's site-local address be tunnelled when mobile
# node is not in its home network (default: yes).
#TUNNEL_SITELOCAL=yes

# Number of tunnel devices MIPL can use. Limits the number of simultaneous
# bindings between different HAs and MNs.
TUNNEL_NR=5

# Home address for mobile node with prefix length. Example:
# 3FFE:2620:6:1234:ABCD::1/48 (Don't use the example value!)
#HOMEADDRESS=3ffe:3103:0:8:260:8ff:fe46:46e7/64

# Home agent's address for mobile node.
#HOMEAGENT=3ffe:3103:0:8::3/64

# Path for a file containing list of IPv6 addresses (and other
# information) of mobile nodes for which this node is allowed to act
# as a home agent.
MOBILENODEFILE=/etc/mipv6_acl.conf

#Uncomment this to use the new authentication mechanism from draft 15.
#Otherwise the siganling is authenticated only if you compiled AH support
#AUTHENTICATION=draft15

# Configuration file for security associations used for authentication of
# MIPv6 signaling.
SAFILE=/etc/mipv6_sas.conf

# Mobile nodes are allowed to send Router Solicitation messages at a
# shorter minimum interval than normal IPv6 nodes (default: 4
# seconds). This value controls the mobile nodes minimum Router
# Solicitation interval in seconds (default: 1).
#RTR_SOLICITATION_INTERVAL=1

# After sending MAX_RTR_SOLICITATIONS (defined by IPv6, default: 3)
# mobile node reduces Router Solicitation send rate with binary
# exponential back-off mechanism until the maximum send time limit is
# reached. This option sets the maximum send time in seconds.
#RTR_SOLICITATION_MAX_SENDTIME=5
```

mipv6-acl.conf

```
###
# Example:
# Allow all nodes with 3ffe:2620:6:1234:: prefix to register except
# nodes 3ffe:2620:6:1234:abcd::.
#
# DENY 3ffe:2620:6:1234:abcd::/80
# ALLOW 3ffe:2620:6:1234::/64
ALLOW ::/0
```

mipv6-sas.conf

```
# These all SAs are due to cover all the scenarios.
3ffe:3103:0:8:201:2ff:fe2a:93b9;1033;ALG_AUTH_HMAC_MD5;outbound;0;abcdefghijkl
1234567890
3ffe:3103:0:8:201:2ff:fe2a:93b9;1034;ALG_AUTH_HMAC_MD5;inbound;0;1234567890abc
defghijkl
3ffe:3103:0:8:207:eff:feb9:69fc;1033;ALG_AUTH_HMAC_MD5;outbound;0;abcdefghijkl
1234567890
3ffe:3103:0:8:207:eff:feb9:69fc;1034;ALG_AUTH_HMAC_MD5;inbound;0;1234567890abc
defghijkl
3ffe:3103:0:8:260:8ff:fe46:46e7;1033;ALG_AUTH_HMAC_MD5;outbound;0;abcdefghijkl
1234567890
3ffe:3103:0:8:260:8ff:fe46:46e7;1034;ALG_AUTH_HMAC_MD5;inbound;0;1234567890abc
defghijkl
```

radvd.conf

```
interface eth0
{
        AdvSendAdvert on;
        AdvDefaultLifetime 600;
#
# These settings cause advertisements to be sent every 3-10 seconds.  This
# range is good for 6to4 with a dynamic IPv4 address, but can be greatly
# increased when not using 6to4 prefixes.
#
        MinRtrAdvInterval 0.5;
        MaxRtrAdvInterval 1.5;
#
# Disable Mobile IPv6 support
#
        AdvHomeAgentFlag on;
#
# example of a standard prefix
#
        prefix 3ffe:3103:0:8::0/64
{
                AdvOnLink on;
                AdvAutonomous on;
                AdvRouterAddr off;
        };
};
```

6.3.1.2 CN configurations

network-mip6.conf

```
# MIPL Mobile IPv6 Configuration file

# Should this node act as a home agent (ha), mobile node (mn) or
# correspondent node (cn). HA and MN both have CN functionality
# embedded. Default: cn.
```

```
FUNCTIONALITY=cn

# In error situations it may be desired to get more detailed
# information what is happening. Increase this value to get more
# messages from the module (default: 0).
DEBUGLEVEL=1

# Should unicasts to node's site-local address be tunnelled when mobile
# node is not in its home network (default: yes).
#TUNNEL_SITELOCAL=yes

# Number of tunnel devices MIPL can use. Limits the number of simultaneous
# bindings between different HAs and MNs.
TUNNEL_NR=2

# Home address for mobile node with prefix length. Example:
# 3FFE:2620:6:1234:ABCD::1/48 (Don't use the example value!)
#HOMEADDRESS=3ffe:3103:0:8:260:8ff:fe46:46e7/64

# Home agent's address for mobile node.
#HOMEAGENT=3ffe:3103:0:8::3/64

# Path for a file containing list of IPv6 addresses (and other
# information) of mobile nodes for which this node is allowed to act
# as a home agent.
MOBILENODEFILE=/etc/mipv6_acl.conf

#Uncomment this to use the new authentication mechanism from draft 15.
#Otherwise the siganling is authenticated only if you compiled AH support
#AUTHENTICATION=draft15

# Configuration file for security associations used for authentication of
# MIPv6 signaling.
SAFILE=/etc/mipv6_sas.conf

# Mobile nodes are allowed to send Router Solicitation messages at a
# shorter minimum interval than normal IPv6 nodes (default: 4
# seconds). This value controls the mobile nodes minimum Router
# Solicitation interval in seconds (default: 1).
#RTR_SOLICITATION_INTERVAL=1

# After sending MAX_RTR_SOLICITATIONS (defined by IPv6, default: 3)
# mobile node reduces Router Solicitation send rate with binary
# exponential back-off mechanism until the maximum send time limit is
# reached. This option sets the maximum send time in seconds.
#RTR_SOLICITATION_MAX_SENDTIME=5
```

mipv6-acl.conf

```
###
# Example:
# Allow all nodes with 3ffe:2620:6:1234:: prefix to register except
# nodes 3ffe:2620:6:1234:abcd::.
#
# DENY 3ffe:2620:6:1234:abcd::/80
# ALLOW 3ffe:2620:6:1234::/64
ALLOW ::/0
```

mipv6-sas.conf

```
# These all SAs are due to cover all the scenarios.
3ffe:3103:0:8:201:2ff:fe2a:93b9;1033;ALG_AUTH_HMAC_MD5;outbound;0;abcdefghijkl
1234567890
```

```
3ffe:3103:0:8:201:2ff:fe2a:93b9;1034;ALG_AUTH_HMAC_MD5;inbound;0;1234567890abc
defghijkl
3ffe:3103:0:8:207:eff:feb9:69fc;1033;ALG_AUTH_HMAC_MD5;outbound;0;abcdefghijkl
1234567890
3ffe:3103:0:8:207:eff:feb9:69fc;1034;ALG_AUTH_HMAC_MD5;inbound;0;1234567890abc
defghijkl
3ffe:3103:0:9:201:2ff:fe2a:93b9;1033;ALG_AUTH_HMAC_MD5;outbound;0;abcdefghijkl
1234567890
3ffe:3103:0:9:201:2ff:fe2a:93b9;1034;ALG_AUTH_HMAC_MD5;inbound;0;1234567890abc
defghijkl
3ffe:3103:0:10:201:2ff:fe2a:93b9;1033;ALG_AUTH_HMAC_MD5;outbound;0;abcdefghijk
l1234567890
3ffe:3103:0:10:201:2ff:fe2a:93b9;1034;ALG_AUTH_HMAC_MD5;inbound;0;1234567890ab
cdefghijkl
```

6.3.1.3 MNs configurations

MN1:

network-mip6.conf

```
# MIPL Mobile IPv6 Configuration file

# Should this node act as a home agent (ha), mobile node (mn) or
# correspondent node (cn). HA and MN both have CN functionality
# embedded. Default: cn.
FUNCTIONALITY=mn

# In error situations it may be desired to get more detailed
# information what is happening. Increase this value to get more
# messages from the module (default: 0).
DEBUGLEVEL=1

# Should unicasts to node's site-local address be tunnelled when mobile
# node is not in its home network (default: yes).
#TUNNEL_SITELOCAL=yes

# Number of tunnel devices MIPL can use. Limits the number of simultaneous
# bindings between different HAs and MNs.
TUNNEL_NR=2

# Home address for mobile node with prefix length. Example:
# 3FFE:2620:6:1234:ABCD::1/48 (Don't use the example value!)
HOMEADDRESS=3ffe:3103:0:8:201:2ff:fe2a:93b9/64

# Home agent's address for mobile node.
HOMEAGENT=3ffe:3103:0:8::3/64

# Path for a file containing list of IPv6 addresses (and other
# information) of mobile nodes for which this node is allowed to act
# as a home agent.
MOBILENODEFILE=/etc/mipv6_acl.conf

#Uncomment this to use the new authentication mechanism from draft 15.
#Otherwise the siganling is authenticated only if you compiled AH support
#AUTHENTICATION=draft15

# Configuration file for security associations used for authentication of
# MIPv6 signaling.
SAFILE=/etc/mipv6_sas.conf

# Mobile nodes are allowed to send Router Solicitation messages at a
# shorter minimum interval than normal IPv6 nodes (default: 4
# seconds). This value controls the mobile nodes minimum Router
# Solicitation interval in seconds (default: 1).
RTR_SOLICITATION_INTERVAL=1
```

```
# After sending MAX_RTR_SOLICITATIONS (defined by IPv6, default: 3)
# mobile node reduces Router Solicitation send rate with binary
# exponential back-off mechanism until the maximum send time limit is
# reached. This option sets the maximum send time in seconds.
RTR_SOLICITATION_MAX_SENDTIME=5
```

mipv6-acl.conf

```
###
# Example:
# Allow all nodes with 3ffe:2620:6:1234:: prefix to register except
# nodes 3ffe:2620:6:1234:abcd::.
#
# DENY 3ffe:2620:6:1234:abcd::/80
# ALLOW 3ffe:2620:6:1234::/64
ALLOW ::/0
```

mipv6-sas.conf

```
# These all SAs are due to cover all the scenarios.
3ffe:3103:0:9::3;1033;ALG_AUTH_HMAC_MD5;inbound;0;abcdefghijkl1234567890
3ffe:3103:0:9::3;1034;ALG_AUTH_HMAC_MD5;outbound;0;1234567890abcdefghijkl
3ffe:3103:0:8::3;1033;ALG_AUTH_HMAC_MD5;inbound;0;abcdefghijkl1234567890
3ffe:3103:0:8::3;1034;ALG_AUTH_HMAC_MD5;outbound;0;1234567890abcdefghijkl
3ffe:3103:0:8:260:8ff:fe46:46e7;1033;ALG_AUTH_HMAC_MD5;outbound;0;abcdefghijkl
1234567890
3ffe:3103:0:8:260:8ff:fe46:46e7;1034;ALG_AUTH_HMAC_MD5;inbound;0;1234567890abc
defghijkl
3ffe:3103:0:8:207:eff:feb9:69fc;1033;ALG_AUTH_HMAC_MD5;outbound;0;abcdefghijkl
1234567890
3ffe:3103:0:8:207:eff:feb9:69fc;1034;ALG_AUTH_HMAC_MD5;inbound;0;1234567890abc
defghijkl
3ffe:3103:0:9:260:8ff:fe46:46e7;1033;ALG_AUTH_HMAC_MD5;outbound;0;abcdefghijkl
1234567890
3ffe:3103:0:9:260:8ff:fe46:46e7;1034;ALG_AUTH_HMAC_MD5;inbound;0;1234567890abc
defghijkl
3ffe:3103:0:10:260:8ff:fe46:46e7;1033;ALG_AUTH_HMAC_MD5;outbound;0;abcdefghijk
l1234567890
3ffe:3103:0:10:260:8ff:fe46:46e7;1034;ALG_AUTH_HMAC_MD5;inbound;0;1234567890ab
cdefghijkl
```

MN2:

network-mip6.conf

```
# MIPL Mobile IPv6 Configuration file

# Should this node act as a home agent (ha), mobile node (mn) or
# correspondent node (cn). HA and MN both have CN functionality
# embedded. Default: cn.
FUNCTIONALITY=mn

# In error situations it may be desired to get more detailed
# information what is happening. Increase this value to get more
# messages from the module (default: 0).
DEBUGLEVEL=1

# Should unicasts to node's site-local address be tunnelled when mobile
# node is not in its home network (default: yes).
#TUNNEL_SITELOCAL=yes
```

```
# Number of tunnel devices MIPL can use. Limits the number of simultaneous
# bindings between different HAs and MNs.
TUNNEL_NR=2

# Home address for mobile node with prefix length. Example:
# 3FFE:2620:6:1234:ABCD::1/48 (Don't use the example value!)
HOMEADDRESS=3ffe:3103:0:8:260:8ff:fe46:46e7/64

# Home agent's address for mobile node.
HOMEAGENT=3ffe:3103:0:8::3/64

# Path for a file containing list of IPv6 addresses (and other
# information) of mobile nodes for which this node is allowed to act
# as a home agent.
MOBILENODEFILE=/etc/mipv6_acl.conf

#Uncomment this to use the new authentication mechanism from draft 15.
#Otherwise the siganling is authenticated only if you compiled AH support
#AUTHENTICATION=draft15

# Configuration file for security associations used for authentication of
# MIPv6 signaling.
SAFILE=/etc/mipv6_sas.conf

# Mobile nodes are allowed to send Router Solicitation messages at a
# shorter minimum interval than normal IPv6 nodes (default: 4
# seconds). This value controls the mobile nodes minimum Router
# Solicitation interval in seconds (default: 1).
RTR_SOLICITATION_INTERVAL=1

# After sending MAX_RTR_SOLICITATIONS (defined by IPv6, default: 3)
# mobile node reduces Router Solicitation send rate with binary
# exponential back-off mechanism until the maximum send time limit is
# reached. This option sets the maximum send time in seconds.
RTR_SOLICITATION_MAX_SENDTIME=5
```

mipv6-acl.conf

```
###
# Example:
# Allow all nodes with 3ffe:2620:6:1234:: prefix to register except
# nodes 3ffe:2620:6:1234:abcd::.
#
# DENY 3ffe:2620:6:1234:abcd::/80
# ALLOW 3ffe:2620:6:1234::/64
ALLOW ::/0
```

mipv6-sas.conf

```
# These all SAs are due to cover all the scenarios.
3ffe:3103:0:9::3;1033;ALG_AUTH_HMAC_MD5;inbound;0;abcdefghijkl1234567890
3ffe:3103:0:9::3;1034;ALG_AUTH_HMAC_MD5;outbound;0;1234567890abcdefghijkl
3ffe:3103:0:8::3;1033;ALG_AUTH_HMAC_MD5;inbound;0;abcdefghijkl1234567890
3ffe:3103:0:8::3;1034;ALG_AUTH_HMAC_MD5;outbound;0;1234567890abcdefghijkl
3ffe:3103:0:8:201:2ff:fe2a:93b9;1033;ALG_AUTH_HMAC_MD5;outbound;0;abcdefghijkl
1234567890
3ffe:3103:0:8:201:2ff:fe2a:93b9;1034;ALG_AUTH_HMAC_MD5;inbound;0;1234567890abc
defghijkl
3ffe:3103:0:8:207:eff:feb9:69fc;1033;ALG_AUTH_HMAC_MD5;outbound;0;abcdefghijkl
1234567890
3ffe:3103:0:8:207:eff:feb9:69fc;1034;ALG_AUTH_HMAC_MD5;inbound;0;1234567890abc
defghijkl
```

```
3ffe:3103:0:9:201:2ff:fe2a:93b9;1033;ALG_AUTH_HMAC_MD5;outbound;0;abcdefghijkl
1234567890
3ffe:3103:0:9:201:2ff:fe2a:93b9;1034;ALG_AUTH_HMAC_MD5;inbound;0;1234567890abc
defghijkl
3ffe:3103:0:10:201:2ff:fe2a:93b9;1033;ALG_AUTH_HMAC_MD5;outbound;0;abcdefghijk
l1234567890
3ffe:3103:0:10:201:2ff:fe2a:93b9;1034;ALG_AUTH_HMAC_MD5;inbound;0;1234567890ab
cdefghijkl
```

6.3.2 Windows, Cisco and Linux interoperability Tests

6.3.2.1 CISCO HA configurations

HA
Router(config)# **ipv6 mobile**
Router(config)# **interface FastEthernet 0/0**
Router(config-if)# **ipv6 mobile home-agent enable**

other options can be configured to configure lifetimes, limit MN addresses, etc.
Router(config-if)# **ipv6 mobile home-agent ?**

Some consulting commands:
see the active interfaces acting as a HA.
Router# **show ipv6 mobile home-agents**

show the Mobility traffic (BA accepted
Router# **show ipv6 mobile traffic**

show the HA binding cache.
Router# **show ipv6 mobile binding**

6.3.2.2 Windows CN configurations

CN
C:\> **ipv6 mipu CN**

disable the mobility configuration.
C:\> **ipv6 mipu**
show the functionality configured.
C:\> **ipv6 mip**
show the binding cache.
C:\> **ipv6 bc**

6.3.2.3 Windows MN configurations

MN
C:\> **ipv6 mipu MN**
C:\> **ipv6 hau <MN_ha> <Prefix_len> <HA_adddress>**

disable the mobility configuration.
C:\> **ipv6 mipu**
show the functionality configured.
C:\> **ipv6 mip**

show the binding cache.
C:\> **ipv6 bc**

6.3.2.4 Windows MNs configurations

MN + CN
C:\> **ipv6 mipu MN CN**
C:\> **ipv6 hau <MN_ha> <Prefix_len> <HA_adddress>**
* If just a MN functionality is specified, the MN don't act as a CN (not allowing the MN - MN scenario.)

disable the mobility configuration.
C:\> **ipv6 mipu**
show the functionality configured.
C:\> **ipv6 mip**
show the binding cache.
C:\> **ipv6 bc**

6.3.2.5 Linux CN configurations

network-mip6.conf

```
# MIPL Mobile IPv6 Configuration file

# Should this node act as a home agent (ha), mobile node (mn) or
# correspondent node (cn). HA and MN both have CN functionality
# embedded. Default: cn.
FUNCTIONALITY=cn

# In error situations it may be desired to get more detailed
# information what is happening. Increase this value to get more
# messages from the module (default: 0).
DEBUGLEVEL=1

# Should unicasts to node's site-local address be tunnelled when mobile
# node is not in its home network (default: yes).
#TUNNEL_SITELOCAL=yes

# Number of tunnel devices MIPL can use. Limits the number of simultaneous
# bindings between different HAs and MNs.
TUNNEL_NR=2

# Home address for mobile node with prefix length. Example:
# 3FFE:2620:6:1234:ABCD::1/48 (Don't use the example value!)
#HOMEADDRESS=3ffe:3103:0:8:260:8ff:fe46:46e7/64

# Home agent's address for mobile node.
#HOMEAGENT=3ffe:3103:0:8::3/64

# Path for a file containing list of IPv6 addresses (and other
# information) of mobile nodes for which this node is allowed to act
# as a home agent.
MOBILENODEFILE=/etc/mipv6_acl.conf

#Uncomment this to use the new authentication mechanism from draft 15.
#Otherwise the signalling is authenticated only if you compiled AH support
#AUTHENTICATION=draft15
```

```
# Configuration file for security associations used for authentication of
# MIPv6 signalling.
SAFILE=/etc/mipv6_sas.conf

# Mobile nodes are allowed to send Router Solicitation messages at a
# shorter minimum interval than normal IPv6 nodes (default: 4
# seconds). This value controls the mobile nodes minimum Router
# Solicitation interval in seconds (default: 1).
#RTR_SOLICITATION_INTERVAL=1

# After sending MAX_RTR_SOLICITATIONS (defined by IPv6, default: 3)
# mobile node reduces Router Solicitation send rate with binary
# exponential back-off mechanism until the maximum send time limit is
# reached. This option sets the maximum send time in seconds.
#RTR_SOLICITATION_MAX_SENDTIME=5
```

6.3.2.6 Linux MN configurations

network-mip6.conf

```
# MIPL Mobile IPv6 Configuration file

# Should this node act as a home agent (ha), mobile node (mn) or
# correspondent node (cn). HA and MN both have CN functionality
# embedded. Default: cn.
FUNCTIONALITY=mn

# In error situations it may be desired to get more detailed
# information what is happening. Increase this value to get more
# messages from the module (default: 0).
DEBUGLEVEL=1

# Should unicasts to node's site-local address be tunnelled when mobile
# node is not in its home network (default: yes).
#TUNNEL_SITELOCAL=yes

# Number of tunnel devices MIPL can use. Limits the number of simultaneous
# bindings between different HAs and MNs.
TUNNEL_NR=2

# Home address for mobile node with prefix length. Example:
# 3FFE:2620:6:1234:ABCD::1/48 (Don't use the example value!)
HOMEADDRESS=3ffe:3103:0:8:201:2ff:fe2a:93b9/64

# Home agent's address for mobile node.
HOMEAGENT=3ffe:3103:0:8::3/64

# Path for a file containing list of IPv6 addresses (and other
# information) of mobile nodes for which this node is allowed to act
# as a home agent.
MOBILENODEFILE=/etc/mipv6_acl.conf

#Uncomment this to use the new authentication mechanism from draft 15.
#Otherwise the siganling is authenticated only if you compiled AH support
#AUTHENTICATION=draft15

# Configuration file for security associations used for authentication of
# MIPv6 signaling.
SAFILE=/etc/mipv6_sas.conf

# Mobile nodes are allowed to send Router Solicitation messages at a
# shorter minimum interval than normal IPv6 nodes (default: 4
# seconds). This value controls the mobile nodes minimum Router
```

```
# Solicitation interval in seconds (default: 1).
RTR_SOLICITATION_INTERVAL=1

# After sending MAX_RTR_SOLICITATIONS (defined by IPv6, default: 3)
# mobile node reduces Router Solicitation send rate with binary
# exponential back-off mechanism until the maximum send time limit is
# reached. This option sets the maximum send time in seconds.
RTR_SOLICITATION_MAX_SENDTIME=5
```

6.3.2.7 Linux HA configurations

network-mip6.conf

```
# MIPL Mobile IPv6 Configuration file

# Should this node act as a home agent (ha), mobile node (mn) or
# correspondent node (cn). HA and MN both have CN functionality
# embedded. Default: cn.
FUNCTIONALITY=ha

# In error situations it may be desired to get more detailed
# information what is happening. Increase this value to get more
# messages from the module (default: 0).
DEBUGLEVEL=1

# Should unicasts to node's site-local address be tunnelled when mobile
# node is not in its home network (default: yes).
#TUNNEL_SITELOCAL=yes

# Number of tunnel devices MIPL can use. Limits the number of simultaneous
# bindings between different HAs and MNs.
TUNNEL_NR=5

# Home address for mobile node with prefix length. Example:
# 3FFE:2620:6:1234:ABCD::1/48 (Don't use the example value!)
#HOMEADDRESS=3ffe:3103:0:8:260:8ff:fe46:46e7/64

# Home agent's address for mobile node.
#HOMEAGENT=3ffe:3103:0:8::3/64

# Path for a file containing list of IPv6 addresses (and other
# information) of mobile nodes for which this node is allowed to act
# as a home agent.
MOBILENODEFILE=/etc/mipv6_acl.conf

#Uncomment this to use the new authentication mechanism from draft 15.
#Otherwise the siganling is authenticated only if you compiled AH support
#AUTHENTICATION=draft15

# Configuration file for security associations used for authentication of
# MIPv6 signaling.
SAFILE=/etc/mipv6_sas.conf

# Mobile nodes are allowed to send Router Solicitation messages at a
# shorter minimum interval than normal IPv6 nodes (default: 4
# seconds). This value controls the mobile nodes minimum Router
# Solicitation interval in seconds (default: 1).
#RTR_SOLICITATION_INTERVAL=1

# After sending MAX_RTR_SOLICITATIONS (defined by IPv6, default: 3)
# mobile node reduces Router Solicitation send rate with binary
# exponential back-off mechanism until the maximum send time limit is
# reached. This option sets the maximum send time in seconds.
#RTR_SOLICITATION_MAX_SENDTIME=5
```

6.4 Appendix D: Detailed analysis of MIPv6 implementations in TILAB

6.4.1 Linux MIPL

6.4.1.1 Introduction

This document has, as its main goal, to analyze the performances in terms of throughput of a Mobile Node (MN) equipped with MIPL Mobile IPv6 implementation for Linux.

One of the main mechanisms inside a Mobile IPv6 network is the Route Optimization (RO) that can significantly modify the througput measured on a Mobile Node. Infact when the Route Optimization mechanism is enabled packets coming from the CN to the MN are directly forwarded to the Mobile Node and no intermediate node will be involved in the forwarding.

This analysis will regard, first of all, the measure of the throughput both in a scenario with RO and in a scenario without RO (with different CN operatying systems) and then we will execute some tests by changing the traffic direction (not only from CN to MN, but also from MN to CN).

The tests confirm that Route Optimization mechanism improve the performances, as we expect from the theoretical study.

MIPv6 implementation up to now realized are still experimental versions that need to be improved. Infact MIPv6 specifications are not definitive and consequently it is obvoius that frequent changes and modifications will be made to make them conform to the new draft versions.

Although some implementations have already a good maturity level (like KAME and MIPL), performance measurements are not still available in terms of, for example, handover latency. By now, mostly conformance or interoperability tests have been carried out.

Test we made regard mostly the performances (in terms of throughput) with and without *Route Optimization mechanism*. We thought of using MIPL implementation since is one of the most stable MIPv6 implementations already available.

Moreover, since in a mobility scenario main nodes are the HA and the MN, we chose to use MIPL for HA and MN, while we tested CN both with Linux and Windows.

To verify the improvement of the performances due to the *Route Optimization* mechanism, we measured basically the *throughput* between (in both directions) MN and CN, by changing the network conditions.

Unfortunately our test bed did not permit us to modify the hops number between CN and MN so we simulated a more realistic situation inserting a delay in the packet forwarding between *home network* and *foreign network*.

Since, at the being time, there is no tool to create this delay over the IPv6 traffic, our solution was to set-up an IPv6 in IPv4 tunnel between the HA and the MN, and to insert a node that can simulate delays, packet losses and bandwidth limitations over the IPv4 traffic as Figure 6-28.

6.4.1.1.1 Testbed description

The *testbed* we used is shown in Figure 6-28 where CN is placed inside the same *foreign network* as Mobile Node, and consequently we are in the worst case because, with no *Route Optimization* mechanism enabled, traffic from CN to MN gets through the HA, even if CN and MN are on the same link.

We measured the *throughput* between CN and MN, with changing the delay and the bandwidth on the route between *home network* and *foreign network*. As we already said, HA and MN support Linux operatying system and use MIPL MIPv6 implementation, while in CN we used:

- Linux with MIPL support and *Route Optimization* enabled;
- Linux with MIPL support and *Route Optimization* disabled;
- Windows 2000 with Windows Tech Preview, and without *Route Optimization* mechanism.

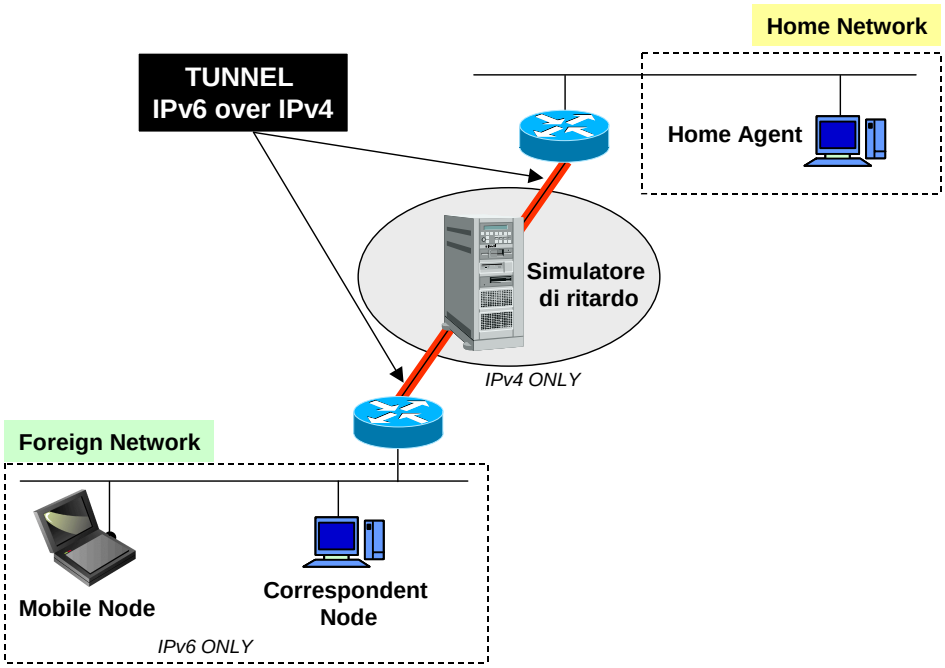


Figure 6-28: Test bed with delay simulator

6.4.1.1.2 Measurements

There are better tools than ttcp to measure performances inside the IPv6 networks.

However we still used ttcp to measure the throughput, because it is the only tool supported both by Linux and Windows and so that is able to interoperate between two operatying systems. We know that with ttcp, a node acts as transmitter and the other one as receiver: all measurements have been done using the CN both as transmitter and as receiver.

Since we want to evaluate the performances of Route Optimization mechanism, we repeated five times these tests.

First of all, we measured the throughput between MN and CN, with varying the delay introduced between home network and foreign network, and for different bandwidth limitations, with a packet loss at 2%. Moreover with the delay simulator NistNet we fixed the standard deviation at 10% of average value.

We remind once again that the delay inserted with NistNet represents a variation of the hop numbers between home network and foreign network.

Although in our testbed delay and bandwidth change independently with each other, in a real scenario, packet delivery time depends on the bandwidth of the route of the packets. We simulate a scenario where the route between CN and MN has both an high bandwidth and several intermediate hops.

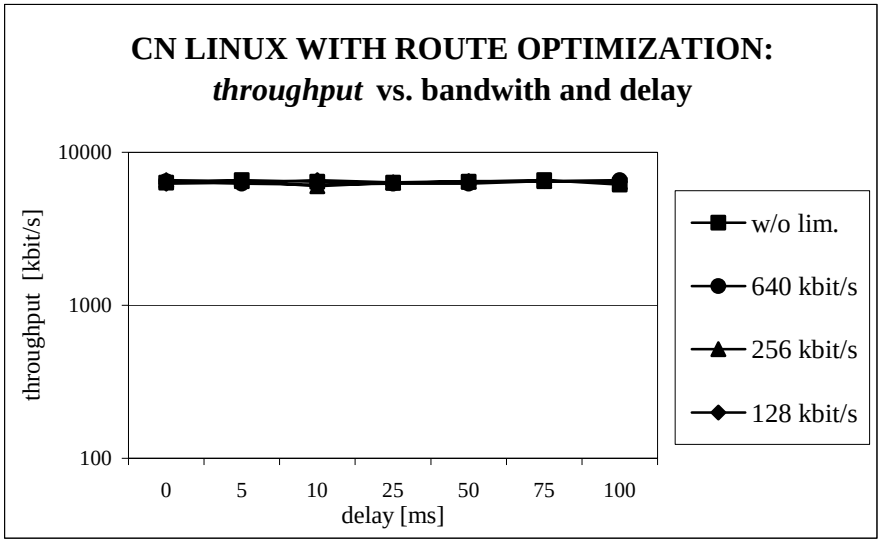
6.4.1.2 Results

In the following pictures we see the throughput between MN and CN, with varying the delay introduced by IPv4 node and using respectively the Linux Correspondent Node with Route Optimization (a), and CN Linux without Route Optimization (b) and Windows Correspondent Node without Routing Optimization (c).

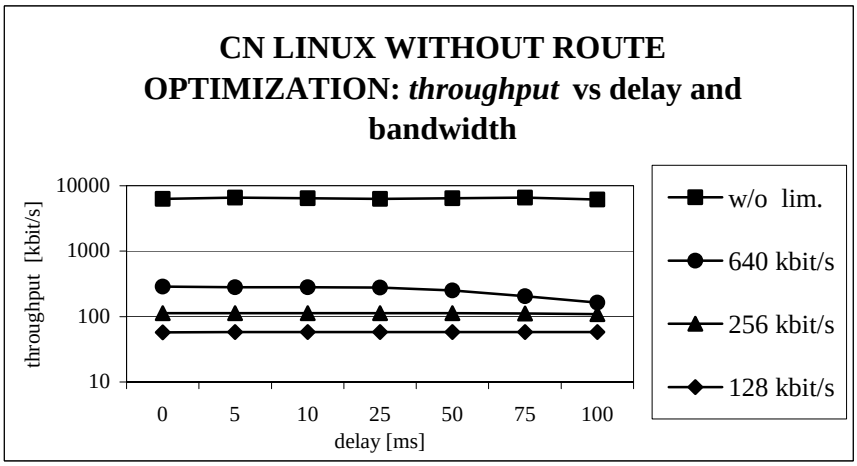
In these graphs, there are also represented throughputs obtained for different bandwidth limitations. We use logarithmic scales because of sometimes the results can vary a lot.

We note immediately that with Route Optimization (a), throughput is not conditioned by bandwidth limitations and delays, since almost all the traffic exchanged between MN and CN goes directly over the link. On the other hand, without Route Optimization, bandwidth limitations influence the result, also if only Windows CN shows an higher dependence on increasing delays.

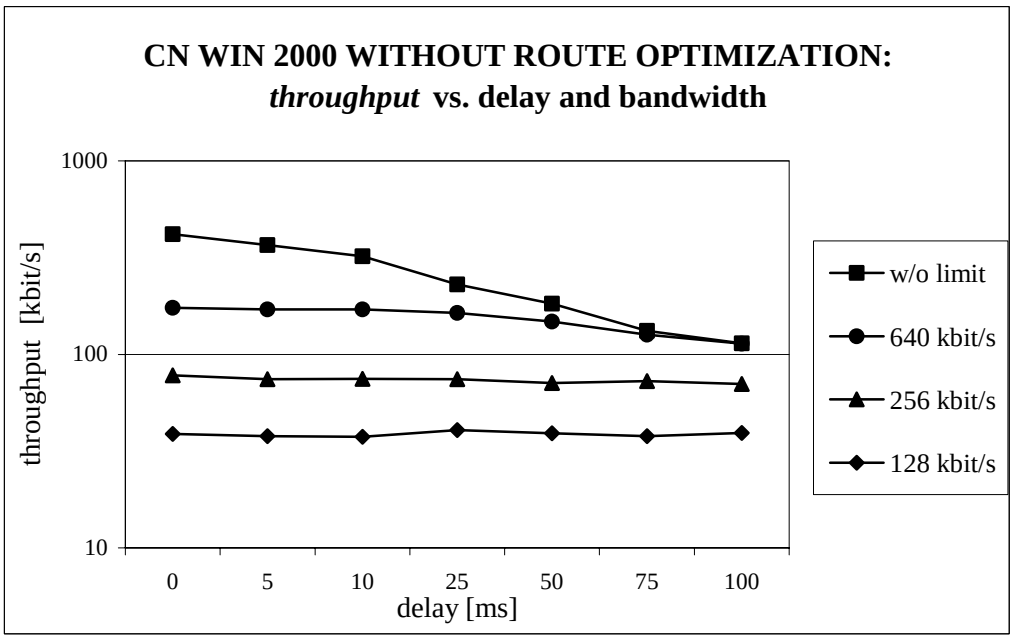
The comparison between graphs b) and c) shows that Windows CN has worse performances than Linux CN.



a) Linux CN and RO



b) Linux CN without RO



c) Windows CN without RO

Figure 6-29: *Throughput* between CN and MN

Next figure shows a comparison between CN performances. Here we can observe the throughputs with and without *Route Optimization*, with varying the bandwidth parameters and for a fixed delay.

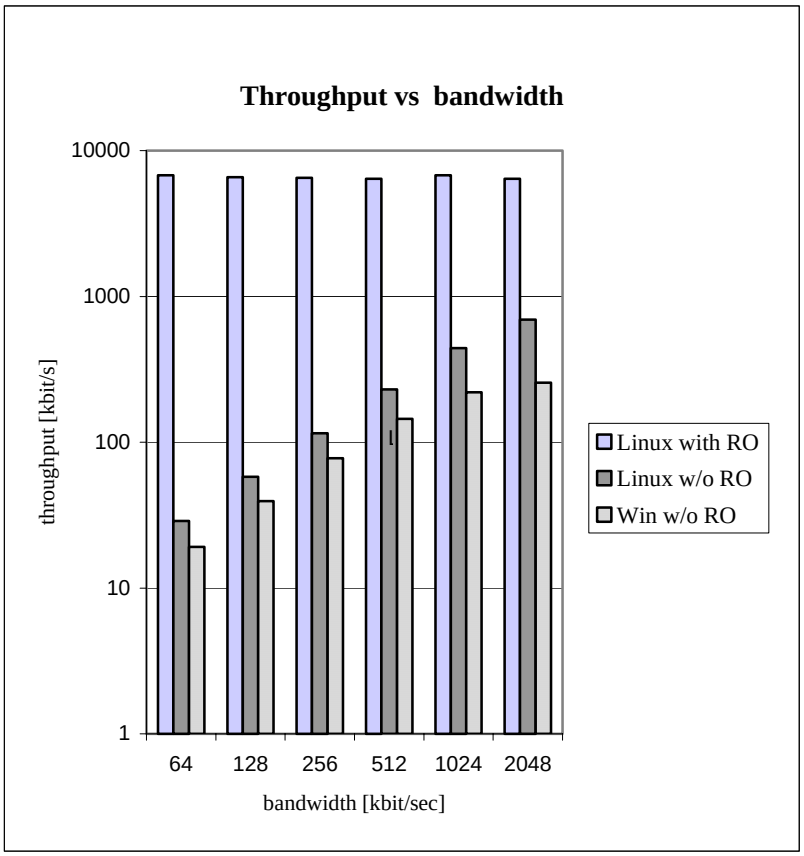


Figure 6-30: comparison table between different throughputs

The previous results refer to scenarios where CN acts as transmitter and MN as receiver, since it is the most probable situation. Note that, in this case, only the packets sent from Correspondent Node are subjected to the delays and bandwidth limitations, whereas the packets sent from MN are sent directly on the link. Consequently these effects (delay and bandwidth limitations) are more relevant when CN transmits and MN receives.

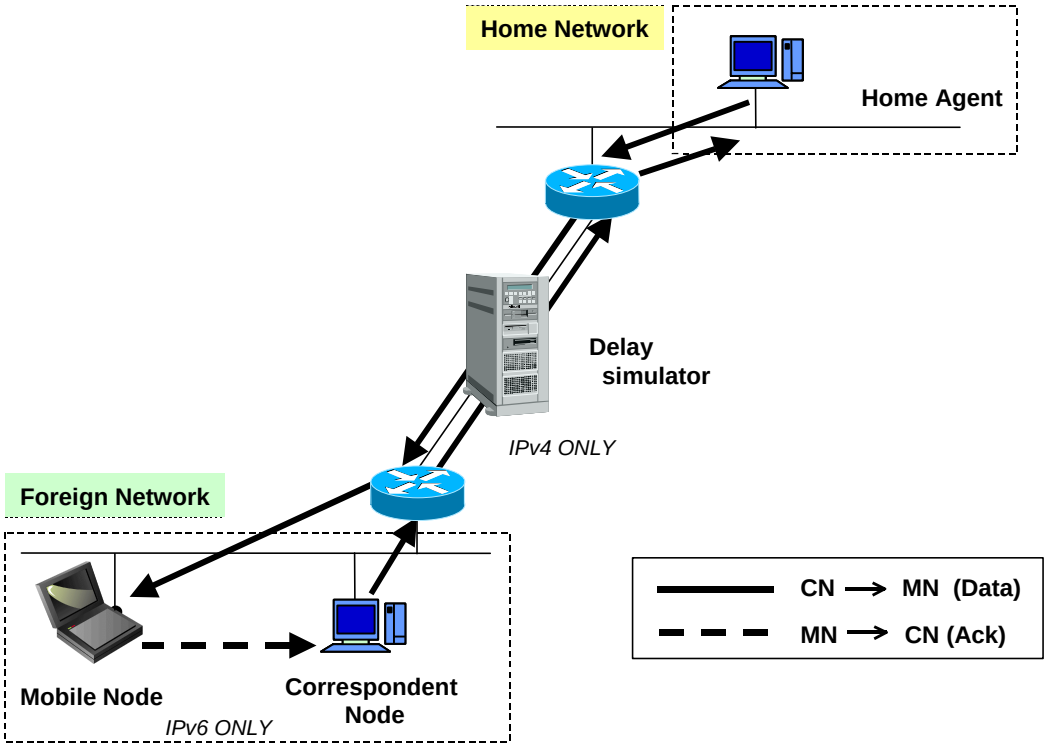
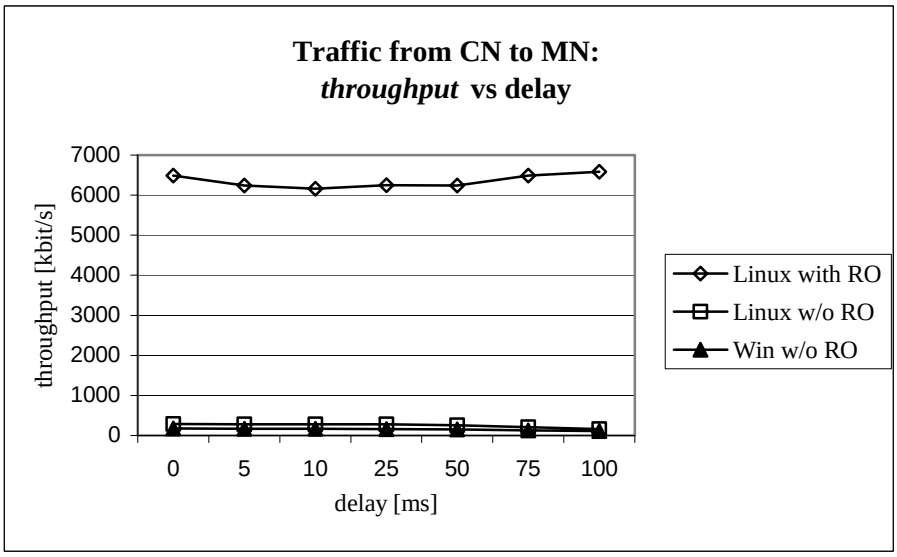
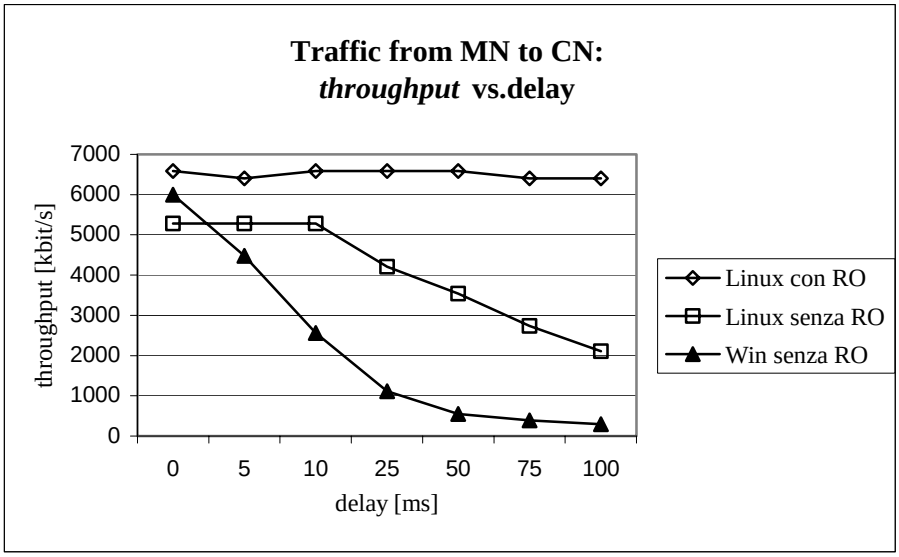


Figure 6-31: Test bed scenario for transmission in both directions

Next picture shows the effects of delay and bandwidth limitation, depending of the traffic direction, without *Route Optimization*. These graphs show that, without *Route Optimization*, when the Correspondent Node transmits, the performances worsen significantly due to delay and bandwidth limitation than the scenario when MN transmits. Infact, if MN transmits, data flow pass twice through the delay simulator and consequently this is the reason why the throughput is significantly less than the scenario where MN is the transmitter. Infact in this last case, only the *acknowledgement* are delayed and hence the throughput is less conditioned.



a)



b)

Figure 6-32: T*hroughput* between CN and MN, depending on traffic direction

6.4.2 Windows 2000

6.4.2.1 Introduction

This document aims to provide an analysis about Windows 2000 Mobile IPv6 implementations that could be used inside the Euro6IX network with reference to the tests carried out in the TILab IPv6 laboratories inside the A4.1 sub-activity.

In this document there are basically two aspects that are taken into account. First one regards mainly the analysis of the implementation, from a more general point of view, where the main features and limitations are described and the second one regards the testing of the implementation to verify the conformance to the standards.

Particularly the experimentation regarded, first of all, mobility tests at network-layer to verify the degree of conformance of the implementations to the MIPv6 draft, then transport layer tests have been carried out (to understand if the transport layer protocols, like TCP and UDP, are transparent to the mobility), and, at the end, application-layer tests have been carried out to analyze what happens to the applications already active in an IPv6 mobility environment when a mobile node moves from a subnet to another one.

This implementation has been analyzed from many points of view. The goal was infact both to understand if the implementation was effectively conform to the MIPV6 standards, both also to verify that it was application transparent. Infact it is important in the same way that the applications were always active although the mobility of node inside the network.

In a mobility scenario the entities that have to support the MIPv6 protocol are the Mobile Node, the Correspondent Node and the Home Agent. Generally speaking, the operating systems that can be used are Windows 2000, FreeBSD and Linux. Our choice, during this experimentation, was to take into account the LandMARC implementation for Windows 2000

The LandMARC implementation is the first experimental version developed by Microsoft Research, and although it is not as mature as some other one already developed for the other operating systems, we decided to test it why Windows is now the most popular end-user operating system all over the world and it could be useful to verify the stability and the functionalities already supported.

The guiding criterion of the tests was to demonstrate, not only the conformance to the MIPv6 draft, but also that, moving a mobile node from a home network to a foreign network, the transport layer and application layer connections were always active.

Hence, the test scenario is characterized by an home network (i.e. a link where Mobile Node and its HA belong), a foreign network (where the mobile node arrives, when it leaves the Home Network), a third network where Correspondent Node (CN for short) is placed (to use as foreign network, when testing the movement of Mobile Node from a foreign network to an other foreign network).

All the tests have been carried out using wired link and Ethernet interfaces, consequently the movements of the mobile node have been realized disconnecting by hand the laptop and connecting it to a different network. Infact, to verify the conformance to the implementation, is not important to use wired or wireless link.

6.4.2.2 Overview on LandMARC implementation for W2000

Microsoft Research, in partnership with a research group of Lancaster University, released an implementation of Mobile IPv6 for Windows2000, available with both source and executable code. Essentially it was added the mobility support to the stack MSR IPv6-ver-1.4, adding MIPv6 features described in the version 12 of protocol (draft-ietf-mobileip-ipv6-12). This implementation can be tested using the IPv6 applications provided with stack IPv6-ver-1.4 (ping6, tracert6, ttcp) and should be interoperable with the Windows2000 Technology Preview for IPv6 and all related applications (http client, ftp client, telnet client/server). It wasn’t tested under any other Windows platform (such as Windows NT).

In the following, we report main features, limitations and bugs of this implementation as indicated by Microsoft Research

6.4.2.2.1 Main Features

- Mobile Node (MN), Correspondent Node (CN) and Home Agent (HA) features.
- Binding Cache on HA e CN and Binding Update List on MN.
- Binding Update and Binding Acknowledgement handling, (Binding Request handling not yet tested)
- Route Optimization mechanism support.
- The Movement Detection mechanism is based on Router Advertisement messages and on NDIS (Network Driver Interface Specification)[1] media connect signaling.
- Interoperability and integration with the existent IPSec features of stack MSR IPv6, to protect from an improper use of signaling control.
- Non-volatile storing of MIPv6 configuration.
- Multiple home addresses configuration capabilities.
- It offers the MIPv6 Configuration Service, a mechanism to configure and specify automatically the home address and the home agent where the user can register among those IPv6 routers announced on the link through Router Advertisement.
- Transparent mobility support for upper layer protocols.

6.4.2.2.2 Limitations

- The main limitations are here reported:
- Handover is too dependent from network adapters and drivers behaviour.
- It doesn’t support site and local home addresses;
- It doesn’t support multi-homed MN
- It doesn’t support the Dynamic Home Agent Address Discovery procedure
- It doesn’t support IKE;
- It doesn’t support forwarding from a previous COA

6.4.2.2.3 Known Bugs

- Refusing of Binding Update (due to a wrong format) exchanged between MNs in the following cases:
- Both are not in the home network
- The packets are fragmented
- The Authentication Header is used
- Route Optimization slow when the handover occurs during a TCP/UDP connection with high traffic (due to the delay of router when transmitting Router Advertisements).

7.1

[1] NDIS are the Windows specifications about interfacing of protocol programs (TCP/IP) with network device drivers

- On some devices the Autoconfiguration Service must be started manually the first time after installation.
- Handover slows when MN and CN are on the HN and when the MN moves elsewhere. The router discards packets sent from CN to MN for a while after confirmed Binding Update
- The MN refuses packets arriving from an IPv4 serial PPP link since DAD (Duplicate Address Discovery) has not been terminated. Solution: UDP, TCP and ICMP must be interested about results of DAD only on the interfaces implementing this mechanism. Some patches to fix this problem are available.
- RTT very high when IPSec is enabled.
- MN Crashes when using a V4-compatible home address on an IPv4 PPP serial line connection.
- Some patches are available as source files (*.c o *.h) to recompile and fix the kernel stack.

6.4.2.3 Testing Activity

In the following, the testing scenario will be shown. The entities involved in the experimentation are three and for each of these information regarding the software installed will be given:

Mobile Node:

Windows2000 Service Pack 1

IPv6 Technology Preview

MSR Mobile IPv6 stack.

Microsoft Internet Explorer 5.5 S.P.1

Correspondent Node:

Windows2000 Service Pack 1;

Microsoft Internet Explorer 5.5 S.P.1

MSR Mobile IPv6 stack;

Fnord! 1.5 http server;

Home Agent:

Windows 2000 S.P.1

MSR IPv6 Mobility Protocol

Note that in our testbed the Home Agent is a node with only one network interface, connected to home network and without any routing functionality. The routers that effectively manage the Router Advertisement and on the Home Network and on the Foreign Network are here in the following indicated:

Home Network Access Router (faito)

NetBSD 1.5

Zebra Routing Daemon v-0.89a

Foreign Network Access Router (`tevere`)

FreeBSD 4.3

Zebra Routing Daemon v-0.92a

Next picture shows the test bed used to execute our testing activity.

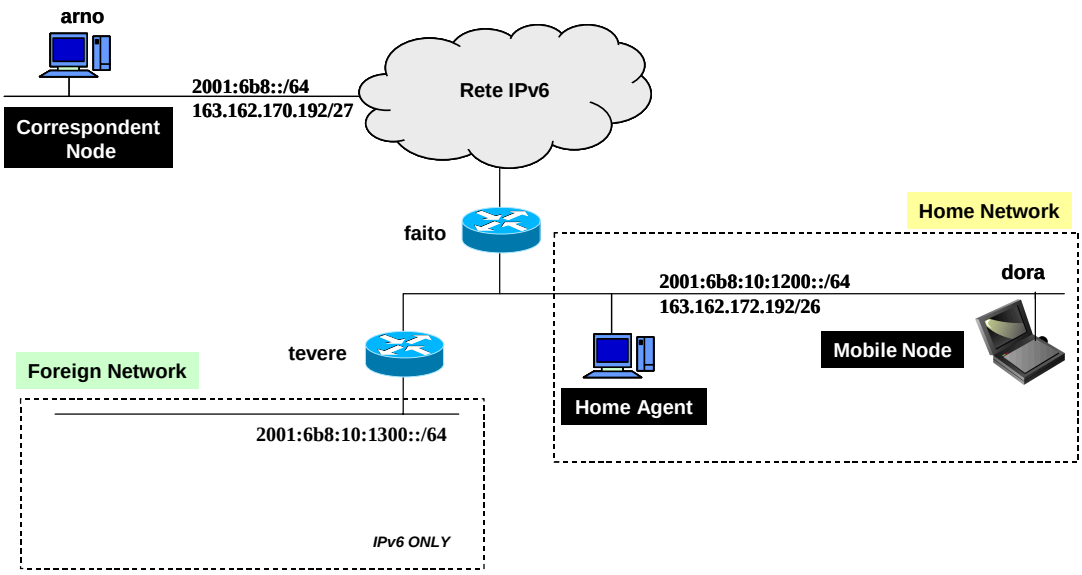


Figure 6-33: Test Bed used for the testing of LandMARC implementation

Note that to let the Mobility IPv6 support work, it could have been enough to install Windows 2000 and subsequently the MIPv6 stack on HA, CN and MN. But, since to test the mobility support we need some further applications for IPv6, available in the Windows 2000 Technology Preview, we had to follow a different procedure on MN and CN.

For example, in Mobile Node, we had to install the packets in a defined order:

- Windows2000 free OS;
- Windows2000 Service Pack 1;
- IE5.5 Service Pack 1;
- Windows2000 Tech Preview;
- MSR Mobile IPv6 *stack*.

Main aim of the experimentation was to verify that the implementation was conform to the protocol draft, and from this point of view, the goal was both to guarantee the mobile node reachability after various movements both to understand the impact of the mobility over transport layer protocols and applications. Delay measurements won't be done, but it will be done a qualitative analysis in order to verify, for example, the correct use of binding cache or the mechanism of Route Optimization.

6.4.2.3.1 Network Layer Mobility Test

Mobility support has been tested, first of all, at network layer. When a mobile node gets access to a network that is not its home network, it has to configure a new IPv6 address (Care-Of Address) and register it to Home Agent in order to set up properly the routing of the packets addresses to the MN. So the mobile node should be able to send and to receive packets towards any other node.

In order to test this feature, a ping6 has been used so to verify the IPv6 reachability between two nodes through the Echo Reply and Echo Request messages.

Moreover we verified that ping6 is active also if mobile node moves to another foreign network. In this case we noted that, after a while, the message exchange starts again properly. Note also that the test have been carried out in both of the directions. In other words, ping6 has been executed both from the mobile node to the Correspondent Node and vice-versa. We observe that results are not the same for both the directions.

Description	**W2000**

The Mobile Node is inside the *foreign network* while a `ping6` from CN to the MN, (and vice-versa) is active.	OK[2]
The MN is inside the home *network*. While a `ping6` from CN to the MN (and vice-versa) is active, the mobile node moves to foreign network.	Some problems[3]
MN is inside the *foreign network*. A `ping6` from CN to MN (and vice-versa) is executed, and during the ICMPv6 messages exchange, MN is disconnected from *foreign network* and come back on the *home network*.	OK[4]
MN is on the *home network*. During a `ping6` between MN and CN, mobile node moves to *foreign network1*, and subsequently on the *foreign network2*	OK[5]

6.4.2.3.2 Transport Layer Test

Since one of the most important goals to achieve, in mobile environment, is that the mobility does not have influence on transport layer protocols. In order to do this we use ttcp6 application that permits to verify the TCP connectivity between two nodes. One node is the transmitter and the other one is the sender. For every received packet, the receiver sends acknowledgement packets to the sender. Moreover the acknowledgment packets can be sent also for more that one single packet. If, after some attempts, the session does not go up, TCP session breaks and goes down.

Description	**W2000**

7.1

[2] OK: *Route Optimization* mechanism works properly for both implementations

[3] HA does not implement properly *Proxy Neighbor Discovery procedure*, infact HA starts to receive the packets going to the MN, forwarding them to the CoA via *tunneling*, with some delay referring to the Binding Update (from MN) reception time. *Route Optimization* mechanism works fine.
If a ping6 is executed from MN to CN, we note a second problem. Infact, following to the movement of MN, MN sends Echo Request on packets that have as source address the *home address*, instead of COA

[4] Following to this movement, there is a temporary loss of connectivity at network layer, but subsequently the exchanges of Echo Request and Echo Reply messages work again properly

[5] `ping6` is active during both movements, and in both cases *Route Optimization mechanism works propoerly*.

MN is inside *foreign network.* A `ttcp6` is executed from CN to the MN.	OK[6]
A `ttcp6` from CN to the MN is executed, while MN moves from *home network* to the *foreign network.*	Some problems[7]
A `ttcp6` from CN to MN (and vice-versa) is executed, while mobile node goes from *foreign network* to the *home network.*	OK

6.4.2.3.3 Application Layer Test

This testing session has been executed in order to verify the impact of mobility over applications.

First test we made was with HTTP. For these tests we used an ad-hoc http server suggested by Microsoft installed on CN (its name is Fnord!). To do these tests we used Internet Explorer 5.5 Service Pack1 (as client with IPv6 support browsing) and in order to monitor continuously the connectivity we used two html pages, each one redirecting to the other one every two seconds.

With the same configuration also two tests have been made: download via http of a file stored inside the CN and the MP3 audio streaming using FreeAmp.

In both cases tests have been made while the MN moving from home network to foreign network and vice-versa.

Description	**Result**

7.1 _____

[6] Routing optimisation mechanism works properly.
[7] When time between the transmission of two consecutive TCP packets is not defined, session usually breaks. When this time increasing, TCP session remain active and mobile node receives all the packets transmitted from CN. *Route Optimization* mechanism has some problems due to the fact that MN inserts, in the packet sent to *foreign network*, its *home address* as source address, instead of Care-Of Address.

HTTP: in this test, on the mobile node, an HTML page is opened (with an automatical redirection to another page) while MN moves from *home network* to the *foreign network*, and then came back to the *home network*.	Some problems[8]
HTTP: in this test, we try to download a file in the MN from CN, while MN moves from home *network* to *foreign network*, and vice-versa.	NO[9]
HTTP: audio streaming via http of a MP3 file stored inside the CN, while mobile node mobile moves from *home network* to the *foreign network*.	NO[10].
TELNET: MN is the telnet *server* and CN is the *client*. While MN is in the home network, a telnet session is opened. Then the mobile node moves to *foreign network*, and at the end comes back to *home network*.	NO[11]

6.4.2.4 Results

The trials demonstrate that this implementation has more problems and bugs than declared from Microsoft Research. It seems that are mainly the functionalities that work wrong:

7.1

[8] Following to the various movements application is always opened and delays are well-accepted. In other words, Internet Explorer continues to show a page, and to ask for the other one to the server, up to connectivity is not recovered and the other page is shown. *Route Optimisation mechanism does not work.*
[9] The application is not usually tolerant to the delays for recovering the transport layer connectivity and it stops. When application has remained active, following that movement of mobile node , it has been noted that the exchange of the packets is similar to that one observed with html pages download.
[10] See the previous test
[11]This application usually is not tolerant to the delays to recover the connectivity at transport layer and it stops (*unsolicited* Neighbor Advertisement is not sent by HA).

- HA, after receiving a Binding Update from MN, moving from HN to a FN, doesn't send an *unsolicited* Neighbor Advertisement in charge of MN to all nodes on the *link*;
- MN, coming back to Home Network and receiving a Binding Acknowledgement from Home Agent, doesn't send an unsolicited Neighbor Advertisement to all nodes on the link;
- MN ignores the messages Redirect sent from the router;
- Immediately after the move of MN to the *foreign network*, there is a transient time during it uses the *Home Address* as source address of packets transmitted, even if it has correctly configured its new COA and sent the Binding Update to the HA

Probably with the testbed suggested by Microsoft, where MN and CN are on different subnets linked to the same router which works as HA too, and the movements are only on these two links, the implementation works better. In fact, that scenario doesn't have the first two problematic items.

6.5 Appendix E: Detailed MIPv6 tests in Consulintel

6.5.1 Mobility tests

6.5.1.1 Introduction

IPv6 includes many features for streamlining mobility support that are missing in IP version 4, including Stateless Address Autoconfiguration and Neighbor Discovery [rfc2460], [rfc2460] and [rfc2462].

Mobility support in IPv6, as proposed by the Mobile IP working group, follows the design for Mobile IPv4. It retains the ideas of a home network, home agent, and the use of encapsulation to deliver packets from the home network to the mobile node's current point of attachment. While discovery of a care-of address is still required, a mobile node can configure its care-of address by using Stateless Address Autoconfiguration and Neighbor Discovery. Thus, foreign agents are not required to support mobility in IPv6. IPv6-within-IPv6 tunnelling is also already specified.

All IPv6 nodes are expected to implement strong authentication and encryption features to improve Internet security. This affords a major simplification for IPv6 mobility support, since all authentication procedures can be assumed to exist when needed and do not have to be specified in the Mobile IPv6 protocol. Even with the security features in IPv6, however, the current working group draft for IPv6 mobility support specifies the use of authentication procedures as infrequently as possible. The reasons for this are twofold. First, good authentication comes at the cost of performance and so should be required only occasionally. Second, questions about the availability of Internet-wide key management are far from resolved at this time.

Mobility in IPv6 (MIPv6) is a "work in progress" field, so all the documents about this topic must be considered just like a snap-shoot of that's moment status [1]. Revisions and updates of mobility's documents are essential.

With this in mind, we have to establish an actual status either of the protocols (drafts) and the implementations (that follows a concrete version of the drafts).

There is a lot of literature covering the basic concepts involved in Mobility over IP, and over IPv6. Further information in [ietf-mip] and [rfc2002].

6.5.1.2 Objectives

MIPv6 is not a closed issue but a "work in progress" field as it was said in previous lines and this field was traditionally bound to developing environments like Linux. HUT [mipl] is an excellent and well known implementation example of MIPv6 on Linux.

Other platforms like Windows did not have any implementation until now because the source of this operating system is not liberated. However more and more advances and new releases are developed every day, some of them based on Windows.

MIPv6 operating on Linux is well known thanks to previous works made by Consulintel, UMU, UPM and PTIN and documented on the Deliverable D4.1 Annex I. However, there is not any kind of experience neither on this field on Windows nor interoperability between MIPv6 Linux and MIPv6 Windows.

So, it seems very interesting to make any kind of trials on this field in order to check the "state of the art" nowadays and get a lot of experience that will be useful to us in the next works.

So, the main objectives Consulintel has proposed on this contribution are the following:

- To know the mobility over IPv6 implementations on Windows there are nowadays.
- To evaluate and to test this MIPv6 implementations.
- To test the interoperability between MIPv6 implementation on Windows environment and MIPv6 implementation on Linux environment.

6.5.1.3 Test Bed

6.5.1.3.1 MIPv6 implementation on Windows

Consulintel mentioned on its lastest contribution on mobility, there were a MIPv6 implementation based on Windows 2000, which is called LandMARC [landmarc] and located at http://research.microsoft.com/programs/europe/projects/MIPv6Dev.asp web site. This implementation was developed by the University of Lancaster under a frame agreement with Microsoft and it were based on draft 12 of the Mobile IPv6 Internet draft. We proposed there in the Future Works section to try that release in order to gain experience with it.

Unfortunately, at the moment we started to prepare our tests (October-November 2002) this release were not available anymore. The reason was that some incompatibilities with Technological Preview stack for Windows 2000 were starting to appear and thus Microsoft decided to drop out the develop of this release and not to offer support to this product. Microsoft prefers focus their efforts with MIPv6 on new operating systems like Windows XP, Windows .NET and Windows CE.

Our idea to this contribution were use this implementation to test it in order to know the working of it. However, LandMARC implementation can not be downloaded anymore due to all already mentioned and so it does not make sense to try it.

Because of all this, Consulintel had to redefine its initial idea to this contribution. We had looked for other MIPv6 implementation on Windows in order to do the proper work in this field.

Fortunately, we have found an experimental IPv6 stack for Windows 2000[icripv6] developed by the Institute for Communications Research (an organism in Singapore) which has mobility support and located at http://www.cwc.nus.edu.sg/ipv6. Although this is an experimental stack, it is completely functional and we could test it.

The main features of this stack are mentioned next.

6.5.1.3.2 ICR IPv6 Stack Features

ICR IPv6 stack can be downloaded from

http://www.icr.a-star.edu.sg/ipv6/dnloadfiles/ICR-IPv6Ver1820.zip

and it presents the following features:

· Total implementation of the IPv6 stack (Neighbour discovery, autoconfiguration, etc).

· The stack includes several DOS tools to configure manually the network interfaces and "pinging"

· The stack comes with an application to easily configure network addresses, to show the route table and to make pings and traceroutes.

No "dll" for browsers is developed, so it is not possible browsing to IPv6 Internet with IExplorer.

· Mobility support with the main following features

- Based on the Mobility IPv6 Internet draft 13
- Only Mobile Node functionality is supported
- No security of any kind on communications between Mobile Node-Home Agent and between Mobile Node-Correspondent Node is supported.
- Discovery Home Agent Address Mechanism is supported

6.5.1.3.3 Installing and configuring the ICR IPv6 Stack

After download the zip file containing all files needed to the stack work fine, it is necessary "unzip" it from a directory and the following files will be extracted:

- cping6.exe	- mip.exe
- cwcip6.inf	- nd.exe
- ifconfig.exe	- readme.txt
- ip6perf.dll	- version.txt

- ipv6man.exe
- wship6.dll
- tcpip6.sys

Then, it is necessary following the next steps:

1.- Go to Start → Configuration → Network Connections → LAN → Properties and the next window will be shown.

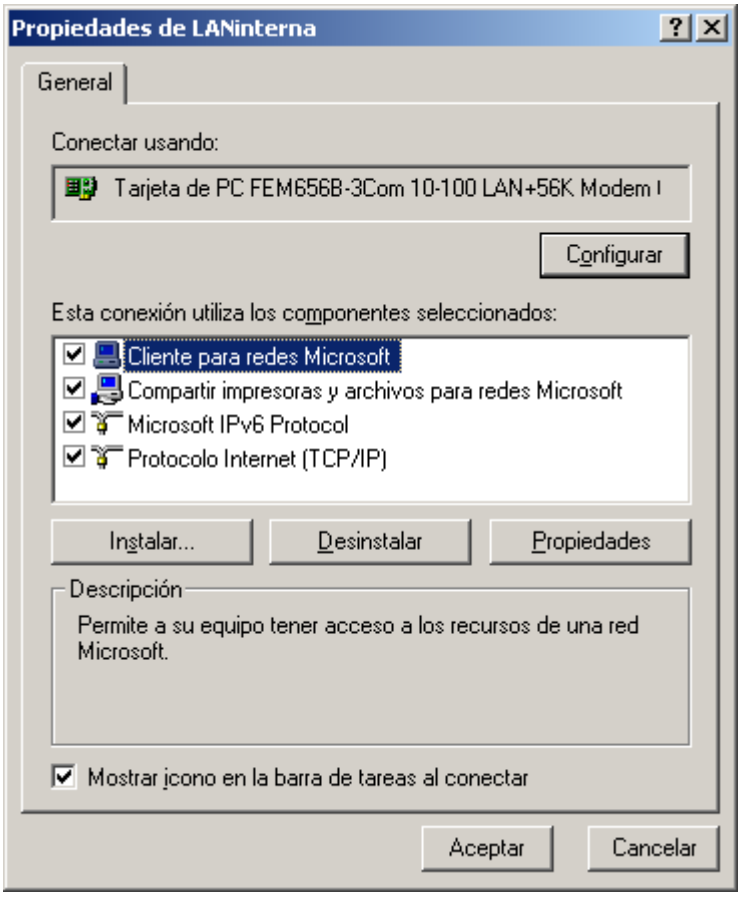


Figure 6-34: LAN properties.

2.- Click the Install button and select "Protocol" in the next window, as it is shown in the next figure. Then click the Add button.

Figure 6-35: Selecting protocol.

3.- In the next window click the "Have Disk" button and then give the path for the ICR stack where all files were "unzipped", for example "C:\ICR-IPv6". IPv6 should now install itself.

4.- The above install procedure will copy the files from the install kit into the appropriate places and add entries to the system registry for IPv6 configuration. Therefore, it is not necessary uninstall any other IPv6 stack which were in your system. It will be automatically uninstall.

5.- After boot time, link-local and global addresses will be added to the network interface because of the stateless address autoconfiguration is supported. However, if a manual configuration will be necessary, the DOS tool called mip, can be used as follow:

mip –start <interface> <home address> <prefix length>

6.- Home Agent's address can be supplied selecting the ICR stack in the next window and then clicking the properties button.

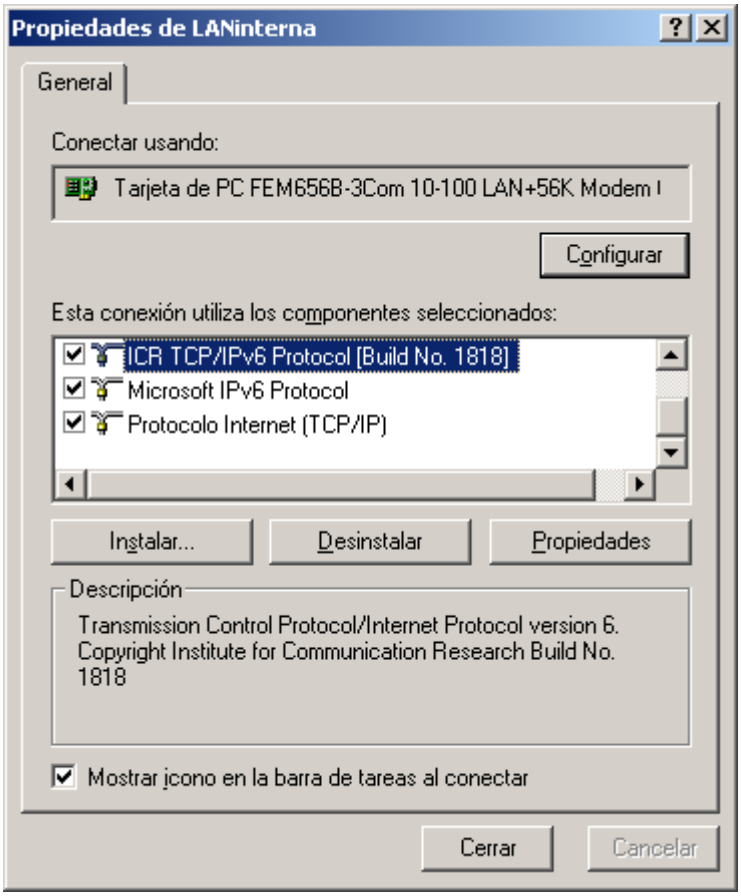


Figure 6-36: LAN properties.

7.- The next window is shown and we can write the address where the Mobile Node will try to register in order to receive the packets sending to their home address.

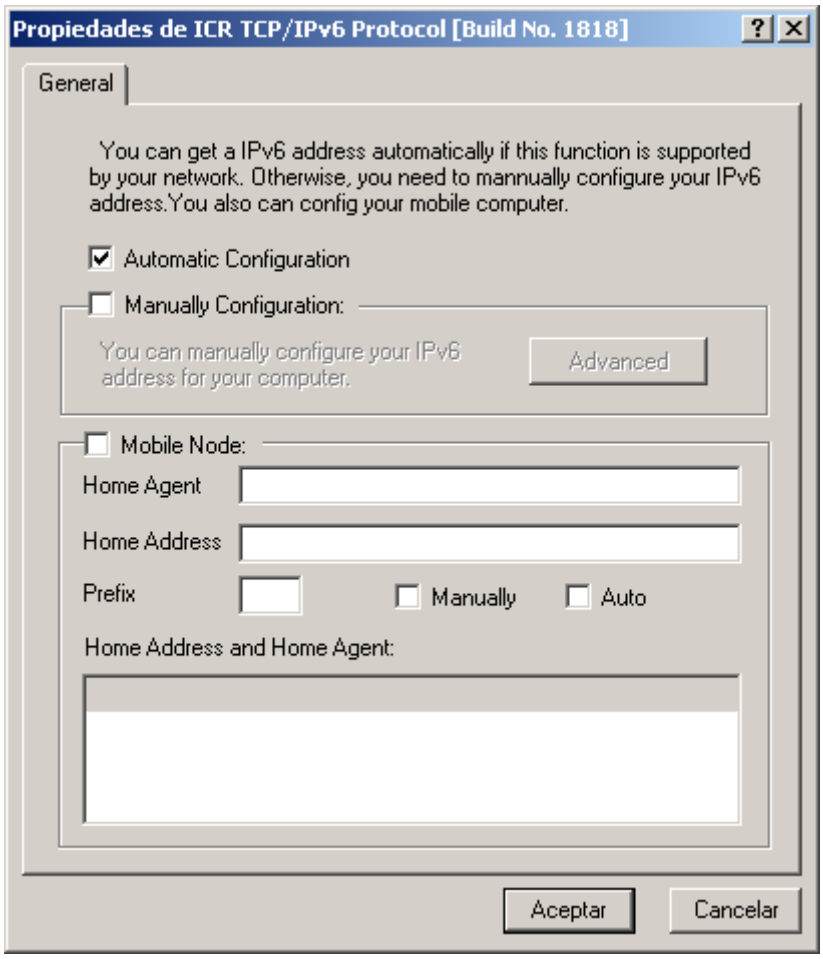


Figure 6-37: ICR IPv6 stack properties

8.- However, the above two steps are not necessaries because of MIPv6 configuration can be done by mean the "ipv6man" application which brings us the possibility to give the mobile node's home address and to start mobility, as it is shown in the next figure:

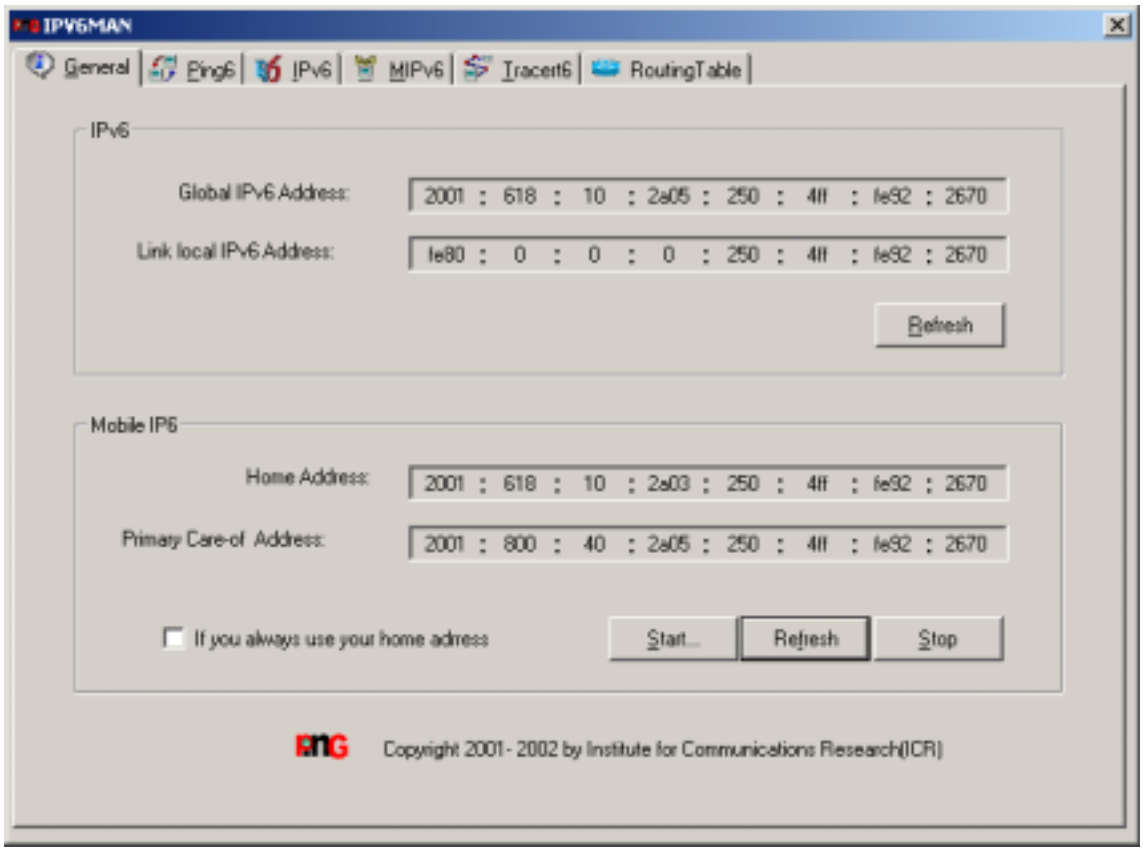


Figure 6-38: IPv6man application

In this case, the Mobile Node will use the Discovery Home Agent Address Mechanism to look for the Home Agent.

6.5.1.4 Designing the test bed

For the purpose of trials and test-beds in MIPv6 in this contribution, we only have considered basic Mobility.

Basic mobility allows a MN to move from one network (home) to other (foreign) without loosing the possibility of being accessible by a CN that knows the home network address of the MN. Also involves the mechanisms for Route Optimization using binding updates.

Unfortunately **security issues** related to mobility could not be tested in this contribution due to the limitations in this field of the ICR IPv6 stack.

Our basic test-bed had three different networks, connected by mean of an IPv6 “cloud” (See next figure). With this scenario several functionalities worked.

The CN (Correspondent Node) is placed in one of them, the CNN (Correspondent Node Network). The MN (Mobile Node) has its own HN (Home Network) with his HA (Home Agent) on it. The third network, FN (Foreign Network), should be used to test the movement of the MN to it, and also the return to HN, also the movement of the MN to the CNN from HN or from the FN.

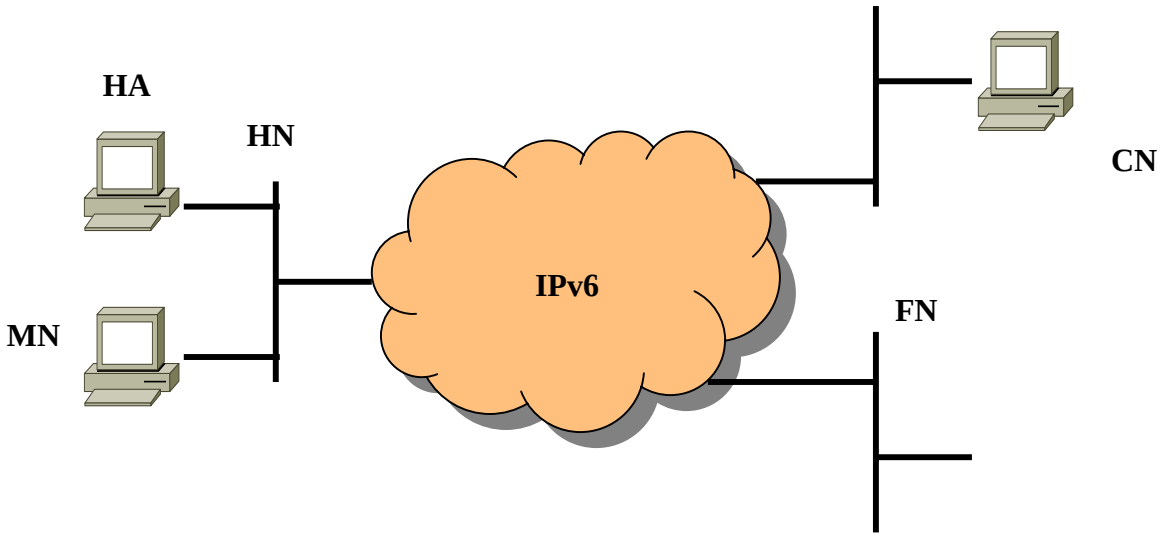


Figure 6-39: Basic Mobility Test-bed

This contribution wants to be a survey about MIPv6 on Windows, so all above functionalities should be implemented on Windows. However, as were already said there is not a complete MIPv6 on Windows that supports all functionalities.

Until now, only MN and CN have been developed on Windows as follow:

- MN. Entity supported on the ICR IPv6 stack. Also Windows CE supports it but it has not been tested in this contribution.
- CN. Entity supported on the new operating systems Windows XP and Windows .NET.
- HA. Entity implemented on Linux.

Therefore, in order to test MIPv6 under Windows it is necessary to make an interesting mixture of different implementations which let us check interoperability with them.

Finally, the next implementations have been chosen in order to do MIPv6 tests on Windows:

- MN: ICR IPv6 stack.
- CN: Windows .NET.
- HA: HUT Linux, version 0.9.4

and the test-bed diagram remains as follows:

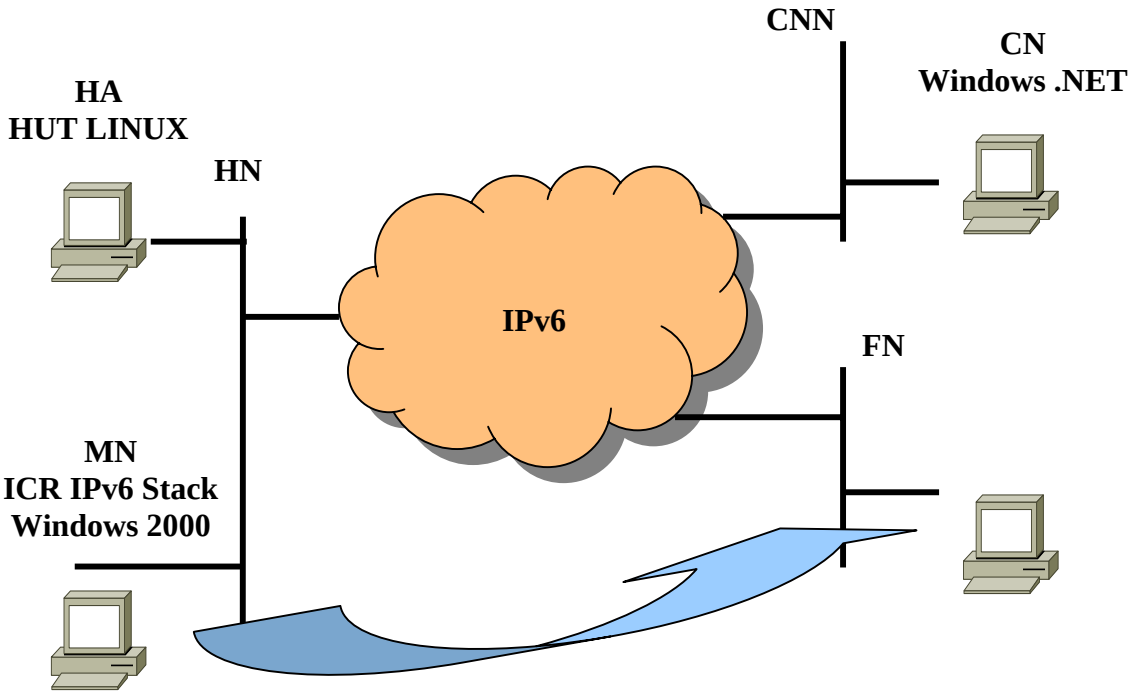


Figure 6-40: Basic Mobility Windows Test-bed

6.5.1.4.1 Designing principles

The network architecture implemented as test-bed is shown in the figure and the following details corresponding to each node are important in order to achieve the total integration.

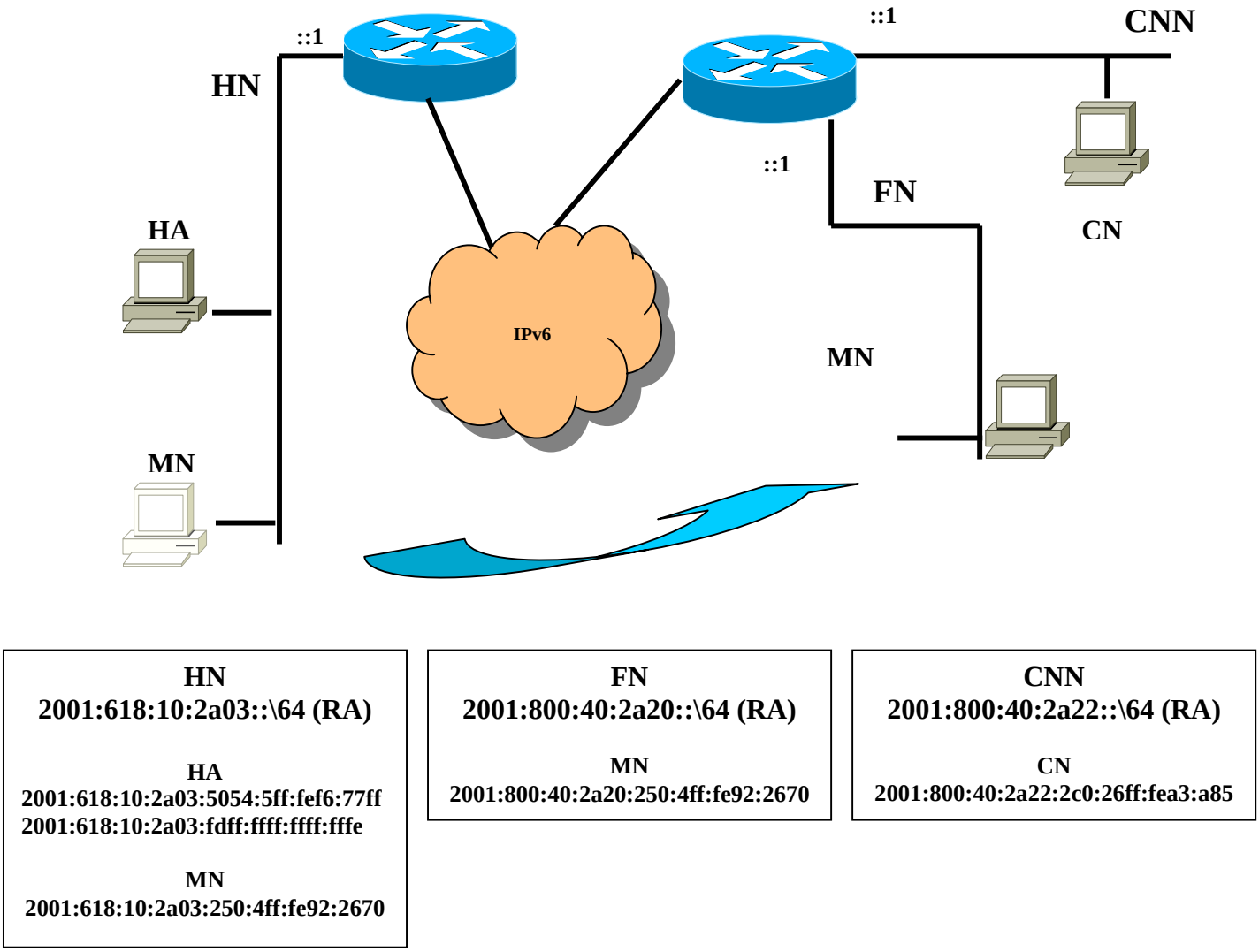


Figure 6-41: Consulintel's network architecture test-bed

In the test-bed Mobile Node moves from Home Network to Foreign Network and from there communicates to Home Agent to do a binding update.

6.5.1.4.1.1 Home Agent

Home Agent was running over a Linux host with Red Hat 7.3 release. The implementation chosen to the test-bed was HUT version 0.9.4 under Linux, which is based on the draft 15 of the Mobile IPv6 Internet draft. No details about installation and configuration are shown here because they were explained in depth in the Deliverable D4.1 Annex I. However, it is important to note that in this case no support to security is needed, thus the **`MIPv6: AH Support`** option in the kernel configuration procedure must be chosen with the '**N**' value before compiling the kernel, and the **`AUTHENTICATION`** and **`SAFILE`** options in the `/etc/sysconfig/network-mip6.conf` file must be commented.

There is only one Home Agent in the Home Network and in order to allow the Discovery Home Agent Address Mechanism works fine it is necessary to configure it as router. Doing this, the Home Agent sends RA packets with the HA bit set to the Home Network which are listened by itself. When the Home Agent realizes that the HA bit is set, it updates their Home Agent List with your own address.

If the Home Agent is not set as router, it will have always their Home Agent List empty and no communication with the Mobile Node will be achieved because HA will not respond to the request address messages from Mobile Node.

To configure the Home Agent as router it is necessary to follow the next steps:

1.- Be sure the HA has forwarding capacity modifying if necessary the `/etc/sysconfig/network` file as follows:

```
# --- IPv6 ---
IPV6FORWARDING=yes
```

2.- When this option is enabled, the Ethernet interface do not listen to any router advertisement so they do not autoconfigure themselves. To give a static IPv6 address to an interface, you must modify the `/etc/syscofig/network-scripts/ifcfg-eth0` file and add next lines where the `IPV6ADDR` option will be particular in each case.

```
# --- IPv6 ---
IPV6INIT=yes
IPV6AUTOCONF=no
IPV6ADDR=2001:618:10:2a03:5054:5ff:fef6:77df/64
```

3.- Furthermore, you must add all routes to the table route as necessary in order to let the HA reaches all networks that it wants to communicate. This is done adding one line with the command route per route to the `/etc/rc.local` file and so this routes are applied at boot time. Also, you must add a new IPv6 address to the HA so that it can receive the messages requesting HA address when a Mobile Node starts their registration process. This new address is a reserved anycast address consisted the Home Network prefix plus a reserved interface identifier.

```
#The next lines add a route to the Foreign Network and a default route
route -A inet6 add 2001:800:40:2a20::/64 gw 2001:618:10:2a03::1 dev
eth0
route -A inet6 add ::/0 gw 2001:618:10:2a03::1 dev eth0

#The next line add a new IPv6 address to the HA in order to listen
# Home Agent Address  Discover Messages (HAAD)
ifconfig eth0 add 2001:618:10:2a03:fdff:ffff:ffff:fffe/64
```

4.- Finally, you must start the **radvd** daemon in order to send router advertisements through network interface to advert their own network. In `/etc/radvd.conf` file you must specify the configuration as follows:

```
interface eth0
{
        AdvSendAdvert on;

# These settings cause advertisements to be sent every 3-10 seconds.
# This range is good for 6to4 with a dynamic IPv4 address, but can be
# greatly increased when not using 6to4 prefixes.

        MinRtrAdvInterval 3;
        MaxRtrAdvInterval 10;

#
# Enable Mobile IPv6 support
#
        AdvHomeAgentFlag on;

#
# Set up of a standard prefix
#
# It is very important add the complete Home Agent addres as prefix
# in order to let the Discover Home Agent Address Mechanism works
# fine
        prefix 2001:618:10:2a03:5054:5ff:fef6:77df/64
        {
                AdvOnLink on;
                AdvAutonomous on;
                AdvRouterAddr on;
        };

};
```

All parameters in this example can be found in the man pages in **radvd** package.

Relevant network parameters for the Home Agent are the following:

```
eth0 Link encap:Ethernet
          HWaddr 52:54:05:F6:77:DF
     inet6 addr: 2001:618:10:2a03:fdff:ffff:ffff:fffe/64 Scope:Global
     inet6 addr: 2001:618:10:2a03:5054:5ff:fef6:77df/64 Scope:Global
     inet6 addr: fe80::5054:5ff:fef6:77df/10 Scope:Link
     UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
     RX packets:24215 errors:0 dropped:0 overruns:0 frame:0
     TX packets:17576 errors:0 dropped:0 overruns:0 carrier:0
     collisions:114 txqueuelen:100
     RX bytes:18356235 (17.5 Mb)  TX bytes:17516136 (16.7 Mb)
     Interrupt:10 Base address:0xe800

ip6tnl1
     Link encap:UNSPEC
          HWaddr 00-00-00-00-00-00-00-00-00-00-00-00-00-00-00-00
     UP RUNNING NOARP  MTU:1460  Metric:1
     RX packets:0 errors:0 dropped:0 overruns:0 frame:0
     TX packets:0 errors:0 dropped:0 overruns:0 carrier:0
     collisions:0 txqueuelen:0
     RX bytes:0 (0.0 b)  TX bytes:0 (0.0 b)
```

6.5.1.4.1.2 Correspondent Node

Windows .NET Server [9] provides a complete server platform for quickly building connected solutions and will be released in the next months. However, you can join to the Preview Customer Program and download the lastest Release Candidate of this operating system at http://www.microsoft.com/windows.netserver/preview/default.mspx Filling in a form you will receive a login and password to get a product key and download the release.

Windows .NET was the operating system chosen to implement the Correspondent Node functionality because of two reasons:

1.- The main objective in this contribution is to use as Window functionalities as possible in order to gain experience in this field.

2.- Using Windows .NET instead of Windows XP (the other Windows implementations that supply Correspondent Node functionality) has as main advantage to get experience in the lastest release of Windows family, which is important in a field as dynamic as mobility.

Once Windows .NET is installed, only two things are necessary to configure the node as Correspondent Node: to install the IPv6 stack and to activate the Correspondent Node functionality.

6.5.1.4.1.2.1 Installing the IPv6 Stack

1.- First of all you must install IPv6 following the next steps

2.- Open the Network Connections Window from Start → Configurations

3.- Click the mouse's right button and then select the **Properties** option

4.- In the **Select Network Component Type** dialog box, click **Protocol** and then click **Add**

5.- In the **Select Network Protocol** dialog box, click **Microsoft TCP/IP Version 6** and click **OK**

6.- Click Close to save changes to your network connection

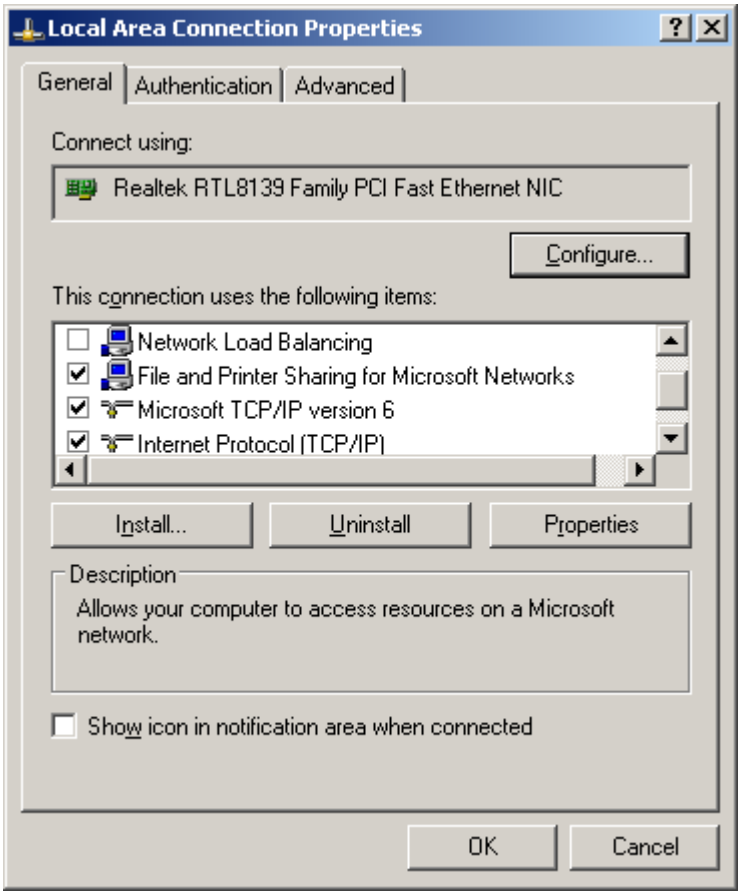


Figure 6-42: Installing IPv6 Protocol

7.- If you need to configure IPv6 with manual addresses you must open the Command Prompt window and type **netsh** command

8.- Next, type **interface ipv6** and press ENTER

9.- Type **add address** [**interface**=]*String* [**address**=]*IPv6Address*

Where:

[**interface**=]*String* specifies the name for the interface

[**address**=]*IPv6Address* specifies theIPv6 address

6.5.1.4.1.2.2 Configuring mobility

1.- From the **netsh ipv6** context type:

set mobility security=disabled correspondentnode=enabled

to enable Correspondent Node functionality works without any kind of security in the bindings procedures with either Home Agent and Mobile Node.

Relevant network parameters for the Correspondent Node are the following:

```
Windows IP Configuration

Ethernet adapter Local Area Connection:
   Connection-specific DNS Suffix  . :
   IP Address. . . . . . . . . . . . :
                              2001:800:40:2a22:4ce7:891d:bb69:8479
   IP Address. . . . . . . . . . . . :
                              2001:800:40:2a22:2c0:26ff:fea3:a850
   IP Address. . . . . . . . . . . . :
                              fe80::2c0:26ff:fea3:a850%4
   Default Gateway . . . . . . . . . :
                              fe80::200:87ff:fe28:10ba%4

Tunnel adapter Teredo Tunneling Pseudo-Interface:
   Connection-specific DNS Suffix  . :
   IP Address. . . . . . . . . . . . : fe80::30a3:fd6a:cf5c:5aa%5
   Default Gateway . . . . . . . . . :

Tunnel adapter Automatic Tunneling Pseudo-Interface:
   Connection-specific DNS Suffix  . :
   IP Address. . . . . . . . . . . . : fe80::5efe:192.168.11.130%2
   Default Gateway . . . . . . . . . :
```

6.5.1.4.1.3 Mobile Node

As was mentioned in previous sections, ICR IPv6 stack was chosen to achieve Mobility Node functionality because it is the only stack that supports this functionality on Windows.

All necessary to install the MN's software was mentioned in ***Installing and configuring the ICR IPv6 Stack*** section so in this section only the steps to start mobility function will be described.

1. From a command prompt window and from the directory where the ICR IPv6 Stack is saved, type the ipv6man command to start the application that let us start mobility

2. Click Start button at the Mobility section and then type the Home Network IPv6 address.

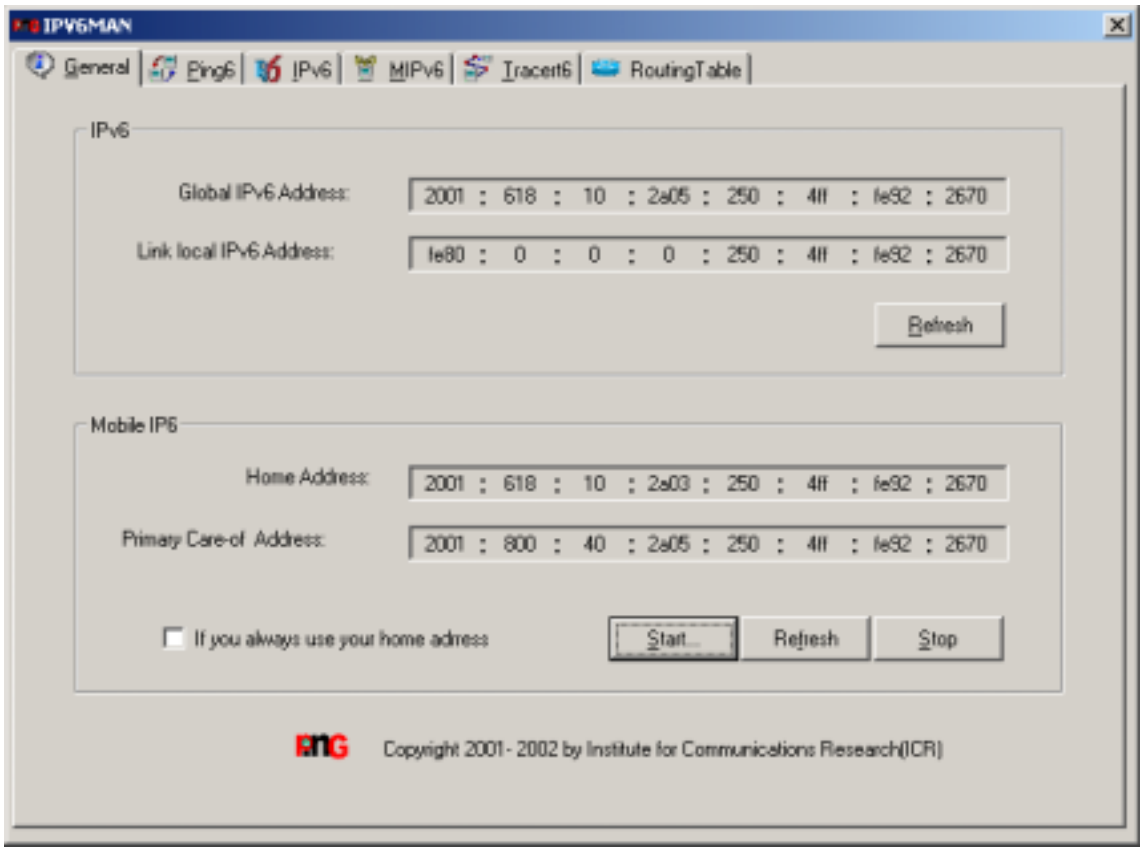


Figure 6-43: Starting mobility from Mobile Node

Relevant network parameters for the Mobile Node are the following:

```
netDevice name:            eth0
  link layer infomantion:    Ethernet
  index:                     1
  flag:                      0xa0
  link layer header size:    14
  link MTU:                  1500
  link layer address length: 6
  link layer address:

  address:            2001:800:40:2a20:250:4ff:fe92:2670
  prefix length:      64
  preferred lifeTime: 0x49d40
  valid lifeTime:     0x13c680
  flag:               0xc83
  state:              4

  address:            ff02::1:ff92:2670
  prefix length:      64
  preferred lifeTime: 0xffffffff
  valid lifeTime:     0xffffffff
  flag:               0x80000e11
  state:              4

  address:            fe80::250:4ff:fe92:2670
  prefix length:      64
  preferred lifeTime: 0xffffffff
  valid lifeTime:     0xffffffff
  flag:               0xc91
  state:              4
```

6.5.1.5 Running the test-bed

Once we have configured everything and the Mobile Node is in the Foreign Network, we can run the mobile support in Mobile Node using the **ipv6man** application. See last figure.

All actions happened in the Home Agent while mobility is running are captured by mean of the **tail** application, which let us see log messages in the `/var/log/messages` file with **–f** parameter as follows:

```
tail –f  /var/log/messages
```

All packets sent to the network from different hosts are registered by mean of two applications: **tcpdump** with either **ip6** or **icmp6** option filter on Linux environment and **ethereal** with the same option filter on Windows.

Next figure shows the scenario when the Mobile Node go away from Home Network and there the next actions happened:

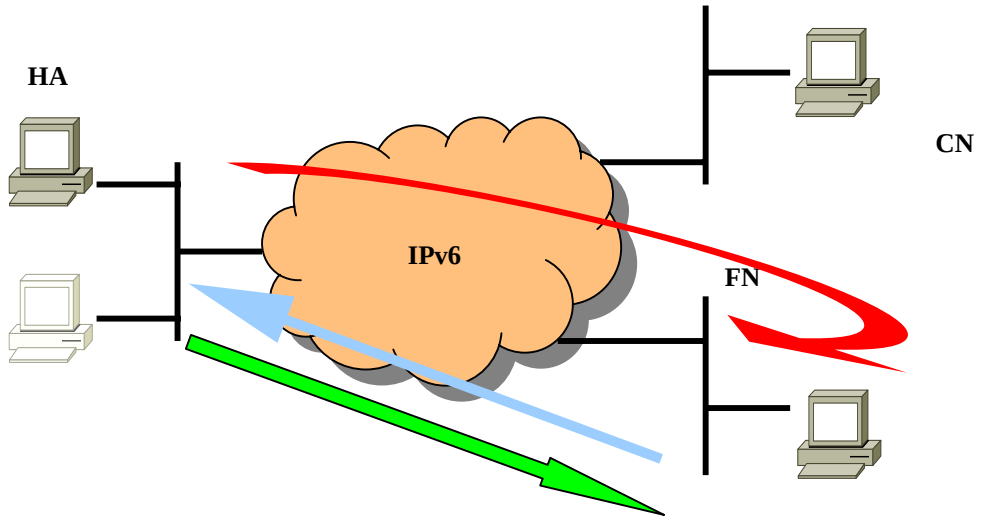


Figure 6-44: Running test-bed

1. In Home Agent the mobility module starts and begins to initialise the Binding Cache and Home Agent List database and initialises the tunnel that will be used to forward packets to Mobile Node. Then it adds the access rules on the mipv6_acl.conf file. Home Agent sends Routing Advertisement packets with HA bit set to the network, which are listened by the itself and then update their Home Agent List database

2. When Mobile Node detects it is away, it makes a new IPv6 address (care-of-address) listening Router Advertisements from the new network.

3. Then it tries to discover the Home Agent Address in the Home Network because it was configure without this data. So, it sends a ICMPv6 *Home Agent Address Discovery Request Message* (**HAAD_Rqt**) to the IPv6 anycast address `2001:618:10:2a03:fdff:ffff:ffff:fffe`, where `2001:618:10:2a03` is the Home Network prefix.

4. Home Agent receives the above **HAAD_Rqt** messages from Mobile Node and prepares and sends a ICMPv6 *Home Agent Address Discovery Reply Message* (**HAAD_Rpy**) with a list of preferred Home Agent Address. However, only one Home Agent is in the Home Network, so the list is empty.

5. Mobile Node receives the **HAAD_Rpy** with the empty list but it get the Home Agent Address of the IPv6 packet Source Address Field.

6. Next, Mobile Node tries to register its new care-of-address in the Home Agent and sends to it a IPv6 packet with the **Binding Update** destination option, which contains its care-of-address and the **A** bit set requesting an acknowledgement message. No security is supported in the ICR IPv6 stack, so this message is not ciphered. Furthermore, this message is based on Mobility Internet draft 13 because of this stack is based on it. The format of this destination option is:

```
 0                   1                   2                   3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
                                +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
                                |  Option Type  | Option Length |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|A|H|R|D|Reservd| Prefix Length |        Sequence Number        |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                            Lifetime                           |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|   Sub-Options...
+-+-+-+-+-+-+-+-+-+-+-+-
```

It is important to note that the size of the Sequence Number is 16 bits.

7. **Binding Update** is received by Home Agent which interprets the packet as a draft 15 Binding Update message because of it is based on this draft version. Format that Home Agent thinks it is receiving is as follows:

```
 0                   1                   2                   3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
                                +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
                                |  Option Type  | Option Length |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|A|H|S|D|               Reserved               |   Sequence #  |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                            Lifetime                           |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|   Sub-Options...
+-+-+-+-+-+-+-+-+-+-+-+-
```

Where the size of the **Sequence Number** field is now 8 bits, so the highest byte of this field is lose.

HA updates its Binding Cache and sends to Mobile Node a draft 15 Binding Acknowledgement message. The format of this new message is:

```
 0                   1                   2                   3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
                                                +-+-+-+-+-+-+-+-+
                                                |  Option Type  |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
| Option Length |    Status     |   Reserved    |   Sequence #  |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                            Lifetime                           |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                            Refresh                            |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|   Sub-Options...
+-+-+-+-+-+-+-+-+-+-+-+-
```

Where again the **Sequence Number** field is 8 bits size and it is filled with lowest byte of the Binding Update Sequence Number field received.

8. This message is received by the Mobile Node, but it thinks it is a draft 13 message with the following format:

```
 0                   1                   2                   3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
                                                +-+-+-+-+-+-+-+-+
                                                |  Option Type  |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
| Option Length |    Status     |        Sequence Number        |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                           Lifetime                            |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                           Refresh                             |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|   Sub-Options...
+-+-+-+-+-+-+-+-+-+-+-+-+-
```

Where the **Sequence Number** field is now 16 bits size. So, only the lowest bits matches the Binding Update Sequence Number field sent previously and Mobile Node do not recognize this acknowledgement.

9. When the acknowledgement timer is out, Mobile Node again sends a Binding Update with a new **Sequence Number** waiting for a draft 13 acknowledgement.

10. Home Agent receives the new Binding Update and sends a corresponding draft 15 acknowledgment. This loop is repeated with no end.

11. Until this point we can summarize the scenario as follows:

- **Home Agent** knows Mobile Node is away and its care-of-address was updated.

- **Mobile Node** knows it is away from the Home Network; it knows the Home Agent Address by mean of Discovery Home Agent Address Messages mechanism; it does not know the Binding Update Messages sent were accepted by the Home Agent and it is continuously sending Binding Updates.

CN communicating to MN

If Corresponding Node does not know Mobile Node is away from Home Network, it sends packets to Mobile Node's Home Address. Now, the new scenario is shown in next figure and the following actions happened when Correspondent Node "ping" to the Mobile Node:

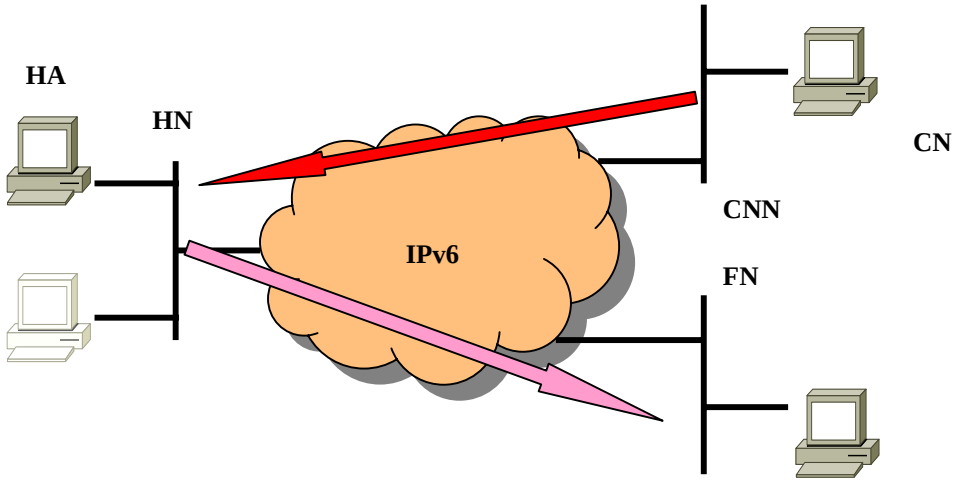


Figure 6-45: CN communicating to MN

1. Home Agent captures packets coming from CN and it tunnelizes them through a IPv6 tunnel to the care-of-address Mobile Node.

2. Tunnelized packets are discarded in Mobile Node because it did not receive yet a valid Binding Acknowledgement. So, it can not send a Binding Update to the Correspondent Node for Route Optimization and does not reply to ping's echo requests.

3. Correspondent Node continues sending pings without replies to the Mobile Node's Home Address.

In this point no more tests can be done because Mobile Node is always in the **Binding Update** state and do not do any transition to other state. Thus, no communication to it can be achieved from either Mobile Node or Correspondent Node.

6.5.1.6 Conclusions

There are several issues that must be considered in order to summarize this work:

1. Only one Home Agent implementation (HUT's MIPL) for Linux platform has been tried.

2. Only one Mobile Node implementation (ICR IPv6 Stack) for Windows platform has been tried.

3. Only one Correspondent Node implementation (Windows .NET) for Windows environment has been tested. So this is a limited test-bed.

4. No security tests/analysis were made due to limitations existing in ICR IPv6 Stack.

The main conclusion in this contribution is clear: **There is no compatibility between ICR IPv6 Stack for Windows 2000 and HUT implementation for Linux** because first one is based on Mobility Internet draft 13 whereas last one is based on draft 15. Different formats in Binding Update and Binding Acknowledgment messages in both drafts makes care-of-address updates between Home Agent and Mobile Node impossible.

Furthermore, a strange behavior in the ICR IPv6 Stack was detected when mobility functionality is started. After a few seconds the stack seems losing connectivity and no packets are sent/received from/to Mobile Node and the global care-of-address becomes to value `0:0:0:0:0:0:0:0` as it is shown in next figure:

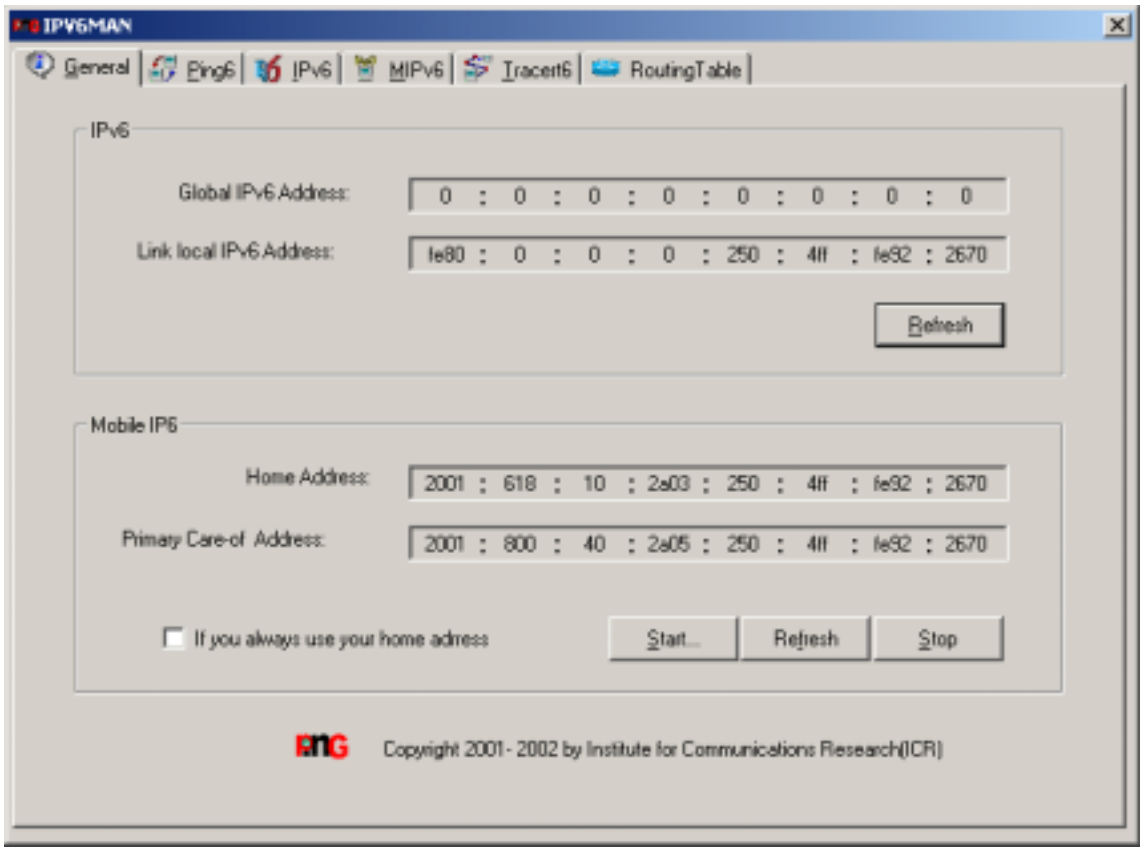


Figure 6-46: Mobile Node losing connectivity

The "feeling" after the tests is that the mobility support in Windows is not a maturity product but experimental in most cases. At present, there is not any implementation with all mobility functionalities but only implementations with Correspondent Node or Mobile Node and any with Home Agent.

Likely the reason is this field is not closed yet but a work in progress and new releases are expected in the future.

6.5.1.7 Future work

There are some things to be done in the future:

1. Test completely Windows .NET Correspondent Node in scenarios.

2. Test other Windows Mobile Node like Windows CE in wireless tests.
3. Perform security tests.
4. Make performance tests.

6.5.1.8 Glossary

MN: Mobile Node

HN: Home Network

HoA: Home Address

HA: Home Agent

CN: Correspondant Node

FN: Foreign Network

CoA: Care-of Address

7. REFERENCES

[mipl] HUT's MIPv6 implementation http://www.mipl.mediapoli.com/

[6wind-ha] Home Agent in 6wind 6200 series. http://www.6wind.com/6200.html

[mgen6] MGEN 6 http://matrix.it.uc3m.es/~long/software/mgen6/

[ipv6stack] INRIA's IPv6 stack. ftp://ftp.inria.fr/network/ipv6/freebsd/

[stackdoc] Documentation INRIA's IPv6 stack. ftp://ftp.inria.fr/network/ipv6/

[freebsd34] FreeBSD 3.4 RELEASE. http://www.freebsdmirrors.org/FBSDsites.php3?release=3.4-RELEASE

[fbsdHbook] FreeBSD HandBook. http://www.freebsd.org/doc/en_US.ISO8859-1/books/handbook/

[hmipv6inria] INRIA's HMIPv6 implementation http://www.inrialpes.fr/planete/people/bellier/hmip.html

[d-mipv6-12] Johnson and Perkins, Mobility Support in IPv6 http://www.monarch.cs.rice.edu/internet-drafts/draft-ietf-mobileip-ipv6-12.txt

[d-hmipv6-07] Soliman, Castellucia Hierarchical Mobile IPv6 mobility management (HMIPv6) http://www.ietf.org/internet-drafts/draft-ietf-mobileip-hmipv6-07.txt

[kame] KAME IPv6 stack. http://www.kame.net

[ietf-mip] IETF Mobile IP Working Group, D. Johnson, C. Perkins, J. Arkko, Mobility Support in IPv6 (draft mobileip-ipv6 version 15), http://www.ietf.org/internet-drafts/draft-ietf-mobileip-ipv6-15.txt, May, 2002.

[ietf-mip-19] IETF Mobile IP Working Group, D. Johnson, C. Perkins, J. Arkko, Mobility Support in IPv6 (draft mobileip-ipv6 version 19), http://www.ietf.org/internet-drafts/draft-ietf-mobileip-ipv6-19.txt, Oct, 2002

[rfc2002] "IP Mobility Support," C. Perkins, ed., IETF RFC 2002, Oct. 1996. http://www.ietf.org/rfc/rfc2002.txt

[rfc2460] S. Deering and R. Hinden, "Internet Protocol, Version 6 (IPv6) Specification," IETF RFC 2460, Dec. 1998. http://www.ietf.org/rfc/`rfc2460.txt

[rfc2461] T. Narten, E. Nordmark and W. Simpson "Neighbor Discovery for IP Version 6 (IPv6)". IETF RFC 2461, Dec. 1998. http://www.ietf.org/rfc/rfc2461.txt

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

[landmarc] The LandMARC Mobile IPv6 implementation for MSRIPv6 1.4. Release Candidate 4.1b http://www.landmarc.net/research/mobility.htm

[icripv6] ICR IPv6 Stack for Windows 2000 http://www.cwc.nus.edu.sg/ipv6

[.net] Windows .NET Server http://www.microsoft.com/net/

[preview] Preview Customer Program http://www.microsoft.com/windows.netserver/preview/default.mspx