Testbed for MIND project on IPv6

Diego Chaparro
dchaparro@acm.org
Raúl Rodríguez
rrodrigu@gsyc.escet.urjc.es

Spanish version: http://../

24th, January, 2002

Índice

1. Introduction

This document explains how to configure a testbed to test a macromobility protocol (*Mobile IPv6*) and a micromobility protocol (*Cellular IPv6*), using *Internet Protocol version 6* (*IPv6*).

The testbed looks like the following figure:

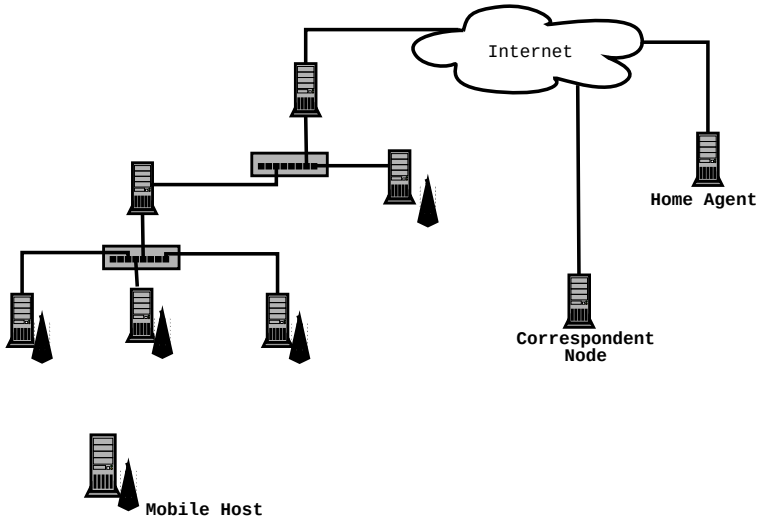


Figure 1

This testbed consists in a set of PCs and a *Mobile Host* that moves between these PCs.

The *Mobile Host* has a communication with the *Correspondent Node* and while the *Mobile Host* is moving from one *Base Station* to another *Base Station* (in the figure 1, PCs with a small antenna), performing *handoffs*, the communication with the *Correspondent Node* is held completely transparent for the user.

The communication between the *Mobile Host* and the *Base Stations* is made using *802.11*, a *Wireless Technology*.

First, we will explain how to prepare the testbed to work with *IPv6* and then we will show how to install *Mobile IPv6* and *Cellular IPv6*.

2. Configuring the testbed with *IPv6*

This testbed is composed of 7 PCs and a laptop. The laptop will be the *Mobile Host*. The operative system used is *Debian GNU/Linux*, kernel version 2.4.7.

The *Mobile Host* and the *Base Stations* have *802.11b wireless cards*. All of them are *PCMCIA cards*, and the *Base Stations* have a *PCI-PCMCIA bridge* to be able use them.

2.1. Configuring the kernel to support *IPv6*, *Mobile IPv6* and *Cellular IPv6*

From the version 2.2.19, the *Linux* kernel supports *IPv6*. In the series 2.4, new features have been incorporated to reach the current state of the *IPv6* module, in experimental state.

Some features in this *IPv6* module are:

- Address space higher
- Authentication and privacy mechanisms
- *IPv4* interaction

The first thing you have to do is to download the kernel sources. You can use the typical way to install packages in *Linux*, using `dselect` o *dpkg* if your distribution is *Debian* or from the `.rpm` package (like in *Red Hat*). You can also dowload them from http://kernel.org/[1].

The default directory where the sources are installed is `/usr/src/kernel-source-2.x.x/` (if you use *Debian GNU/Linux*).

Next, you need to connect to the Helsinki University of Technology[2] and download the *Mobile IPv6* version according to our kernel version. We use kernel version 2.4.7, so we download mipv6-0.9-v2.4.7.tar.gz[3].

A typical directory to unzip and untar the file is `/usr/local/`.

```
host:/usr/local# tar xvfz mipv6-0.9-v2.4.7.tar.gz
```

This command will create the `/usr/local/mipv6-0.9-v2.4.7` directory. In this directory, there is the kernel patch we have to apply (`mipv6-0.9-v2.4.7.patch`) to the kernel sources. This patch works from kernel versions 2.4.4 to 2.4.7.

After untar, copy this `mipv6-0.9-v2.4.7.patch` file to the directory where you untar the kernel sources and apply the patch:

```
host:/usr/local/mipv6-0.9-v2.4.7#: cp mipv6-0.9-v2.4.7.patch ../../src/kernel-source-2.4.7
host:/usr/local/mipv6-0.9-v2.4.7#: cd /usr/src/kernel-source-2.4.7
host:/usr/src/kernel-source-2.4.7#: patch -p1 < mipv6-0.9-v2.4.7.patch
```

[1] `http://kernel.org/`
[2] `http://www.mipl.mediapoli.com/`
[3] `http://www.mipl.mediapoli.com/download/ mipv6-0.9-v2.4.7.tar.gz`

This command modifies the kernel sources to be able to support *Mobile IPv6*. Now, you have to recompile the kernel. Go to the kernel source directory and run one of the tools that allow you recompile it:

- `make menuconfig`
- `make oldconfig`
- `make xconfig`

We prefer `make xconfig`. In the `Networking Options` section are placed the options for *IPv6* and *Mobile IPv6* support. The options you should select are the options that you can find in the `README` file included in the `/usr/local/mipv6-0.9-v2.4.7/` directory. We use this testbed to test *Mobile IPv4* too, so we select more options than the options specified in that `README` file.

The options are the following:

- `Packet socket (Y)`
- `Kernel/User netlink socket (Y)`
- `Routing messages (Y)`
- `Networking packet filtering (replace ipchains) (Y)`
- `Socket filtering (Y)`
- `Unix domain sockets (Y)`
- `TCP/IP networking (Y)`
- `IP: multicasting (Y)`
- `IP: advanced router (Y)`
- `IP: policy routing (Y)`
- `IP: tunneling (Y)`
- `The IPv6 protocolo (EXPERIMENTAL) (m)`
- `IPv6: Mobility Support (EXPERIMENTAL) (m)`
- `MIPv6: Debug Messages (m)`

You should select the options like modules and not included into the kernel to prevent failures the first times you boot the new kernel.

The `MIPv6: AH Support` option offers support for *IPSec Authentication Header* (security mechanism in *Mobile IPv6*). If you select this option, PCs that have been compiled with the `MIPv6: AH Support` option can't interoperate with PCs that haven't been compiled with this option.

Finally, you have to save changes and compile the new kernel in the habitual way:

- `make dep`
- `make clean`
- `make bzImage`
- `make modules`
- `make modules_install`

At this point, your kernel is ready to work with *IPv6* and *Mobile IPv6*. If you are running *Debian GNU/Linux*, you have to update *LILO*: copy the new kernel to the directory that contains the kernel images (normally, in the `/boot` directory):

```
host:/usr/src/kernel-soruce-2.4.7/#: cp arch/i386/boot/bzImage /boot/vmlinuz-2.4.7
```

and add an entry to the `/etc/lilo.conf` file like this:

```
...
image=/boot/vmlinuz-2.4.7
label=Linux-2.4.7
readonly
...
```

If you correctly finished all these steps, you only need to run `lilo` from a terminal and reboot your PC. When *LILO* appers, select *Linux-2.4.7* to boot the kernel with *IPv6* and *Mobile IPv6* support.

Before to reboot the PC, you have to run the next command to complete the *Mobile IPv6* configuration:

```
host:~/: mknod /dev/mipv6_dev c 0xf9 0
```

2.2. Configuring the *PCMCIA 802.11b* cards

We use in our testbed *Lucent Technologies* and *Compaq* cards. All the cards have the *Lucent* chip.

In order to configure the *wireless* cards, you can configure the cards in `Managed` mode or in `Ad Hoc` mode. The last mode is the mode we use for communicate the *Mobile Host* with the *Base Stations*.

To configure the *wireless* cards we need to install the `PCMCIA` package. You can get it from this web page[4]. You should use `make config`, `make all` and `make install` to install the package in your PC.

This package contains some drivers to use with the *Lucent* and the *Compaq* cards (like `wvlan` and `orinoco`). We use the `orinoco` driver because it is newer.

[4] `http://sourceforge.net/projects/pcmcia-cs/`

You should also install the wireless tools[5] package. This package offers some tools to configure the cards easily.

To configure the cards in the `Ad-Hoc` mode, we need to edit the `/etc/pcmcia/wireless.opts` file. This file includes several sections for configuring your *wireless* card. You need to look at the *MAC* address in your *wireless cards* and associate an entry for this *MAC* address with the appropiated driver. For example, our *wireless* cards have *MAC* address like `*:*:*:00:60:1D:*`, `*:*:*:00:02:2D:*` and `*:*:*:00:02:A5:*`, so in the `/etc/pcmcia/wireless.opts` file there is an entry like this:

```
# Lucent Wavelan IEEE
# Note : wvlan_cs driver only, and version 1.0.4+ for encryption support
*,*,*,00:60:1D:*|*,*,*,00:02:2D:*|*,*,*,00:02:A5:*)
    INFO="Wavelan IEEE example (Lucent default settings)"
    ESSID="MINIRED"
    MODE="Ad-Hoc"
    RATE="auto"
```

The `ESSID` field specifies the network identifier. All the cards with this identifier can establish an *Ad Hoc* network when they are configured in `Ad Hoc` mode.

To load the driver you have to restart the *PCMCIA* services with:

```
/etc/init.d/pcmcia restart
```

2.3. Testbed design

To build the testbed we need to define the subnetworks we are going to use, network prefix to be used, *IPv6* address of every subnetwork ...

This testbed has six different subnetworks:

Figure 2

The *IPv6* addresses for every subnetwork are:

- **Subnetwork 1:** fec0:0:0:2::/64
- **Subnetwork 2:** fec0:0:0:3::/64
- **Subnetwork 3:** fec0:0:0:4::/64
- **Subnetwork 4:** fec0:0:0:5::/64
- **Subnetwork 5:** fec0:0:0:7::/64
- **Subnetwork 6:** fec0:0:0:6::/64

[5] `http://www.hpl.hp.com/personal/Jean_Tourrilhes/Linux/Tools.html`

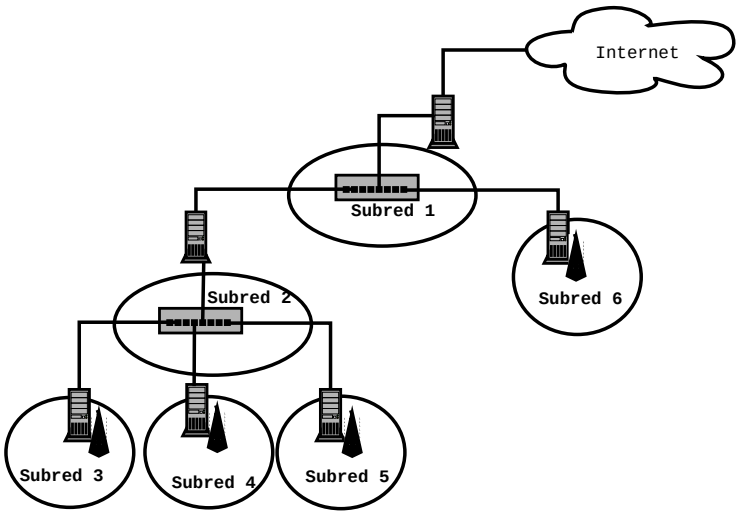
2.4. Configuring *IPv6* addresses

Every PC in the testbed (except the *Mobile Host*) has a pair of interfaces: one interface is connected to the upper part of the tree and the other to the lower PCs:

- Upper interface (to the root of the tree): It automatically configures an *IPv6* address from a `radvd` (*Router Advertisement daemon*).
- Lower interface: It has a fixed *IPv6* address and has a `radvd` to announce the prefix subnetwork to the *Mobile Hosts* that come to this subnetwork (if they are *Base Stations*).

The `radvd` (*Router Advertisement daemon*) sends periodically packets informing about its subnetwork prefix. A *Mobile Host* that receives these messages can automatically configure its *IPv6* address and attach to this subnetwork. An example message sent by a `radvd` is:

```
Router advertisement from fe80::250:4ff:fe47:d29a (hoplimit 255)
        AdvCurHopLimit: 64
        AdvManagedFlag: off
        AdvOtherConfigFlag: off
        AdvHomeAgentFlag: off
        AdvReachableTime: 0
        AdvRetransTimer: 0
        Prefix fec0:0:0:1::/64
                AdvValidLifetime: 2592000
                AdvPreferredLifetime: 604800
                AdvOnLink: on
                AdvAutonomous: on
                AdvRouterAddr: off
        AdvSourceLLAddress: 00 50 04 47 D2 9A
```



This message means that it has been sent by a host with `fe80::250:4ff:fe47:d29a` *IPv6* address and it advertises that the subnetwork prefix is `fec0:0:0:1::/64`.

The configuration file (in `/etc/radvd.conf`) for this message is the next:

```
interface eth0
        {
                AdvSendAdvert on;
                MaxRtrAdvInterval 10;
                #AdvSourceLLAddress off;
                prefix fec0:0:0:1::/64
                {
                        AdvOnLink on;
                        AdvAutonomous on;
                };
        };
```

When a *Mobile Host* receives a *Router Advertisement*, it automatically configures an adequated *IPv6* address for this subnetwork. It takes the subnetwork prefix and adds its *MAC* address. For example, a host with *MAC* address `00:50:DA:4F:A7:87` and the previous *Router Advertisement* obtains the *IPv6* address `fec0::1:250:daff:fe4f:a787`.

To configured the fixed *IPv6* address you have to specify them in the `/etc/network/interfaces` file. You sould add lines like these (for example, for the subnetwork 4):

```
iface eth2 inet6 static
        address fec0::5:202:a5ff:fe6e:5209
        netmask 64
```

2.5. Configuring routes in *IPv6*

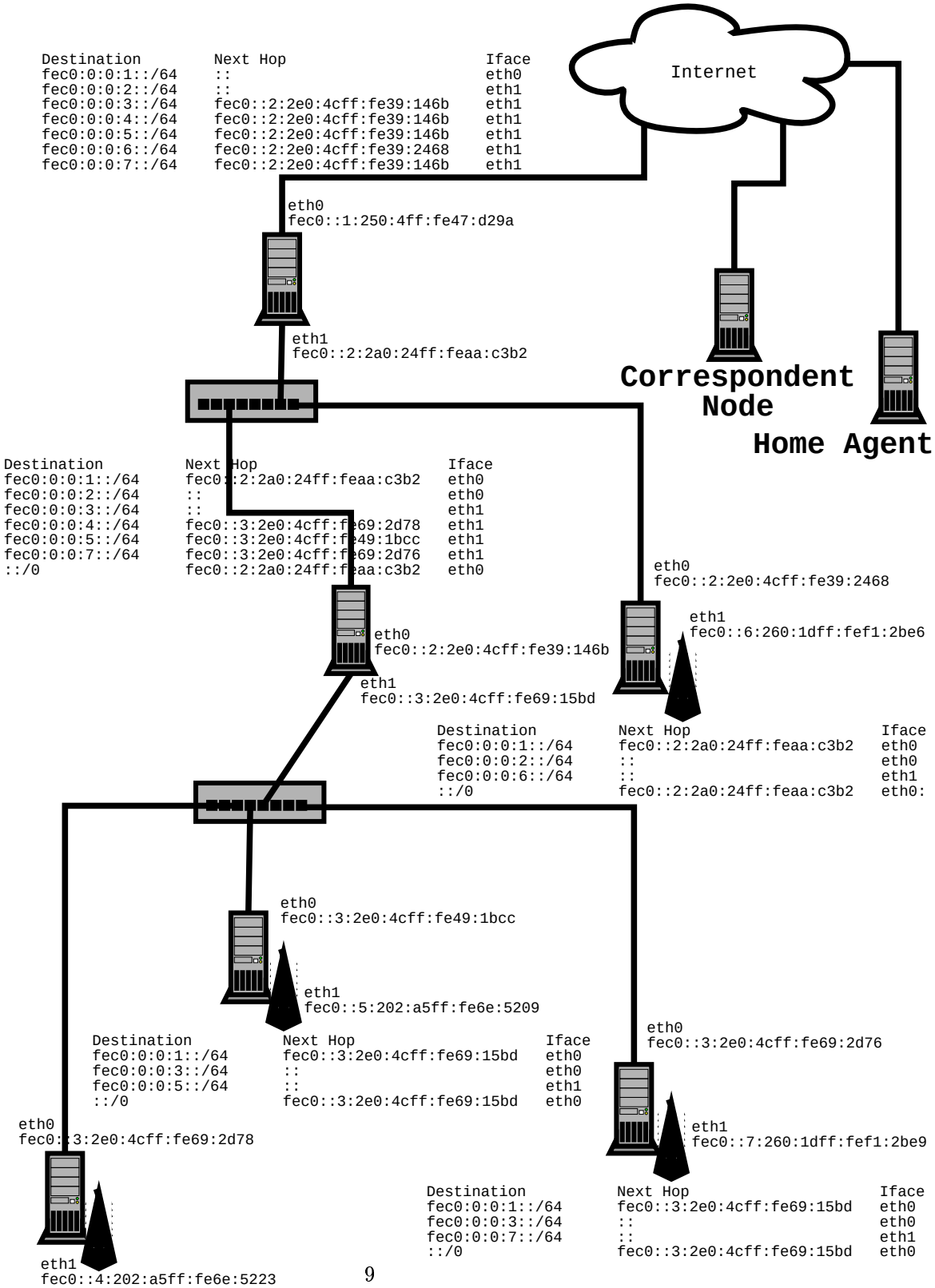
We use a script executed when the PC is booted and establishes the correct routes for the testbed.

You can establish routes with the `route` command:

```
route add -A inet6 fec0:0:0:3::/64 gw fec0::2:2e0:4cff:fe39:146b
dev eth1
```

2.6. Detailed testbed

Next we show a detailed figure with the testbed, showing the *IPv6* address for every PC and the main routes:


Internet
Destination Next Hop Iface
fec0:0:0:1::/64 :: eth0
fec0:0:0:2::/64 :: eth1
fec0:0:0:3::/64 fec0::2:2e0:4cff:fe39:146b eth1
fec0:0:0:4::/64 fec0::2:2e0:4cff:fe39:146b eth1
fec0:0:0:5::/64 fec0::2:2e0:4cff:fe39:146b eth1
fec0:0:0:6::/64 fec0::2:2e0:4cff:fe39:2468 eth1
fec0:0:0:7::/64 fec0::2:2e0:4cff:fe39:146b eth1
eth0
fec0::1:250:4ff:fe47:d29a
eth1
fec0::2:2a0:24ff:feaa:c3b2
Correspondent Node
Home Agent
Destination Next Hop Iface
fec0:0:0:1::/64 fec0::2:2a0:24ff:feaa:c3b2 eth0
fec0:0:0:2::/64 :: eth0
fec0:0:0:3::/64 :: eth1
fec0:0:0:4::/64 fec0::3:2e0:4cff:fe69:2d78 eth1
fec0:0:0:5::/64 fec0::3:2e0:4cff:fe49:1bcc eth1
fec0:0:0:7::/64 fec0::3:2e0:4cff:fe69:2d76 eth1
::/0 fec0::2:2a0:24ff:feaa:c3b2 eth0
eth0
fec0::2:2e0:4cff:fe39:2468
eth1
fec0::6:260:1dff:fef1:2be6
eth0
fec0::2:2e0:4cff:fe39:146b
eth1
fec0::3:2e0:4cff:fe69:15bd
Destination Next Hop Iface
fec0:0:0:1::/64 fec0::2:2a0:24ff:feaa:c3b2 eth0
fec0:0:0:2::/64 :: eth0
fec0:0:0:6::/64 :: eth1
::/0 fec0::2:2a0:24ff:feaa:c3b2 eth0:
eth0
fec0::3:2e0:4cff:fe49:1bcc
eth1
fec0::5:202:a5ff:fe6e:5209
Destination Next Hop Iface
fec0:0:0:1::/64 fec0::3:2e0:4cff:fe69:15bd eth0
fec0:0:0:3::/64 :: eth0
fec0:0:0:5::/64 :: eth1
::/0 fec0::3:2e0:4cff:fe69:15bd eth0
eth0
fec0::3:2e0:4cff:fe69:2d76
eth0
fec0::3:2e0:4cff:fe69:2d78
eth1
fec0::7:260:1dff:fef1:2be9
Destination Next Hop Iface
fec0:0:0:1::/64 fec0::3:2e0:4cff:fe69:15bd eth0
fec0:0:0:3::/64 :: eth0
fec0:0:0:7::/64 :: eth1
::/0 fec0::3:2e0:4cff:fe69:15bd eth0
eth1
fec0::4:202:a5ff:fe6e:5223
Destination Next Hop Iface
fec0:0:0:1::/64 fec0::3:2e0:4cff:fe69:15bd eth0
fec0:0:0:3::/64 :: eth0
fec0:0:0:4::/64 :: eth1
::/0 fec0::3:2e0:4cff:fe69:15bd eth0

3. *Mobile IPv6*

3.1. *Mobile IPv6* basis

Mobile IPv6 specifies how the mobil devices should act using *IPv6*. Every mobil device is identified by its *Home Address*, independently where it is situated in Internet.

When the *Mobile Host* is away from its *Home Network*, it acquires a new *care-of address*. The *IPv6* packets sent to the *Home Address* are routed to its new *care-of address*.

Main differences between *Mobile IPv4* and *Mobile IPv6*:

- The *Route Optimization* concept in *Mobile IPv4*, in *Mobile IPv6* is part of the own protocol.

  This means that packets sent from the *Correspondent Node* to the *Mobile Node* directly go to the *Mobile Node* without going first to the *Home Agent* and then from the *Home Agent* to the *Mobile Node*.

- In *Mobile IPv6*, the packets sent by the *Mobile Node*, in their *source address* is specified the *care-of address*, and in the optional headers is specified the *Home Address*. This makes the packets are routed transparently to routers and to the upper layers.

- Using the *Care-of address* as source address makes easy to route multicast packets sent by the *Mobile Node*.

  In *Mobile IPv4*, the *Mobile Node* establishes a tunnel with its *Home Agent* to be able to use its *Home Address* transparently as source address of its multicast packets.

  In *Mobile IPv6*, using the optional *IPv6* Rounting Header, makes the packet routing compatible with multicast routing.

- In *Mobile IPv6*, the *Foreign Agents* are not necessary anymore. The *Mobile Node* uses the *IPv6* features like *address autoconfiguration* and *Neighbour Discovery*.

- The packets sent by the *Mobile Node* when it is in the *Foreign Network* don't need be encapsulated (like in *Mobile IPv4*). They use the *IPv6 Routing Header* so the number of bytes transmited are reduced.

- When the *Mobile Node* isn't in the *Home Network*, the *Home Agent* intercepts every packet sent to the *MobileNode* using *IPv6 Neighbour Discovery* rather than *ARP* (*Address Resolution Protocol*) like in *Mobile IPv4*. This makes easy the *Mobile IPv6* implementation because it is independent from the link layer.

- The *dynamic Home Agent address*' discovery mechanism in *Mobile IPv6* uses *IPv6 anycast* and returns only one reply to the *Mobile Node*. *Mobile IPv4* uses broadcast messages and a reply for every *Home Agent*.

3.2. Configuring nodes

We need the following nodes to use *Mobile IPv6*:

- A ***Mobile Node***.

  This is the mobil device that "moves" from one subnetwork to a different subnetwork. It has a *Home Network* in which it is situated its *Home Agent*.

  It has an *IPv6* address belonging to that subnetwork: the *Home Address*. The *Mobile Node* is always accesible by this address.

- A ***Home Agent***.

  The *Home Agent* intercepts every packet sent to the *Mobile Node* and forwards them to its current *Care-of address* when it is in the *Foreign Network*.

- A ***Correspondent Node***.

  It is a normal host in Internet. The *Correspondent Node* communicates with the *Mobile Node* (and vice versa).

Every node in the testbed should have correctly installed the *Mobile IPv6* module as we describe in the 2.1 section.

To configure this module, there are three configuration files:

- `/etc/sysconfig/network-mip6.conf`. Main configuration file.
- `/etc/mipv6_acl.conf`. File to specify the *Mobile Node*'s *Access Control List*.
- `/etc/mipv6_sas.conf`. *Mobile IPv6* security aspects.

The configuration for each node would be:

- ***Mobile Node***.

  The `/etc/sysconfig/network-mip6.conf` file contains the following:

```
# MIPL Mobile IPv6 Configuration file

FUNCTIONALITY=mn
DEBUGLEVEL=7
# TUNNEL_SITELOCAL=yes
HOMEADDRESS=fec0::1:260:1dff:fef1:2be8/64
HOMEAGENT=fec0::1:250:4ff:fe47:d29a
# MOBILENODEFILE=/etc/mipv6_acl.conf
RTR_SOLICITATION_INTERVAL=1
RTR_SOLICITATION_MAX_SENDTIME=5
```

- ***Home Agent***.

  The `/etc/sysconfig/network-mip6.conf` contains the next:

```
# MIPL Mobile IPv6 Configuration file

FUNCTIONALITY=ha
DEBUGLEVEL=1
TUNNEL_SITELOCAL=yes
# HOMEADDRESS=fec0::1:260:1dff:fef1:2be8
# HOMEAGENT=fec0::1:250:4ff:fe47:d29a
MOBILENODEFILE=/etc/mipv6_acl.conf
# RTR_SOLICITATION_INTERVAL=1
# RTR_SOLICITATION_MAX_SENDTIME=5
```

And its `/etc/sysconfig/mipv6_acl.conf` file:

```
ALLOW fec0::1:260:1dff:fef1:2be8/64
```

- ***Correspondent Node.***

  The `/etc/sysconfig/network-mip6.conf` contains:

```
# MIPL Mobile IPv6 Configuration file

FUNCTIONALITY=cn
DEBUGLEVEL=2
# TUNNEL_SITELOCAL=yes
# Home address for mobile node with prefix length. Example:
# HOMEADDRESS=3ffe:b00:c18:1fff:0:0:0:bd5
# HOMEAGENT=3ffe:b00:c18:1fff:0:0:0:3cb
# MOBILENODEFILE=/etc/mipv6_acl.conf
# MD5KEY=
# SHA1KEY=
# RTR_SOLICITATION_INTERVAL=1
# RTR_SOLICITATION_MAX_SENDTIME=5
```

We also need that routers in every foreign subnetwork are running a `radvd`, as we explain in the 2.4 section.

3.3. Starting up

Once we have configured all nodes, we only have to enable the *Mobile IPv6* modules. To do this, in every node (*Mobile Node*, *Home Agent* and *Correspondent Node*) we have to executed the next:

```
/etc/init.d/mobile-ip6 start
```

This command enables *Mobile IPv6* in every node. If the *Mobile Node* is in the *Home Network*, it works normally, sending and receiving packets.

To run the `radvd`, you should execute the following:

```
/etc/init.d/radvd start
```

When the *Mobile Node* isn't in the *Home Network*, it needs to acquire a new *IPv6* address from the *Foreign Network* (its *Care-of address*). Then, the *Mobile Node* sends to its *Home Agent* a *Binding Registration* packet to inform the *Home Agent* about its new location (*Binging Update* optional header).

This packet is received by the *Home Agent* and replies it with a *Binding Acknowledgement* packet. Now, the *Mobile Node*'s *IPv6* address will be the *primary care-of address*, and the *Home Agent* will intercept every packet sent to the *Mobile Node* using *Neighbour Discovery* and forward them to the *Mobile Node* using *IPv6 encapsultaion.*

Every time the *Mobile Node* changes its location, will send a *Binding Update* to inform to its *Home Agent.*

You can test *Mobile IPv6* connecting the *Mobile Node* to different subnetworks, seeing how it automatically configures the new *care-of addresses* and checking it goes on in communication with the *Correspondent Node.*

In this page[6] you can view applications with *IPv6* support in *Debian GNU/-Liunx.*

4. *Cellular IPv6*

4.1. *Cellular IPv6* basis

Although *Mobile IPv6* is a powerful Internet mobility protocol, it presents some weaknesses for frequently migrating hosts. Specifically, after each host migration, a local temporary address must be obtained and communicated to a possibly distant *Home Agent.*

This significantly disturbs *TCP* connections while causing network signaling overload. The simplest way to alleviate this weakness is to introduce hierarchies into the *IP* mobility infrastructure. A hierarchical *IP* mobility management scheme specifies that host mobility should be handled where it originates, namely in the access network.

Cellular IPv6 is such an approach, which combines the efficiency and scalability of *IP* with inherent features found in cellular networks, such as seamless handoff support, passive connectivity and paging.

Thus, *Cellular IPv6* is a *Mobile IPv6* protocol extension and not a replacement. A *Cellular IPv6* network is comprised of a *Gateway* router that connects the network to the Internet as well as a set of nodes that are responsible for routing packets to *Mobile Hosts* connected to the network via wireless access points called *Base Stations.*

The *Cellular IPv6* desing and implementation is based on the original implementation of the Columbia University[7]. The main design issues for *Cellular IPv6* are:

- the use of *IPv6* extension headers to carry control information

[6] `http://people.debian.org/~csmall/ipv6/packages.html`
[7] `http://comet.ctr.columbia.edu/cellularip/`

- authentication transactions based on *IPv6* authentication headers
- deployment of IPv6 stateless address autoconfiguration to obtain a *care-of address*, and
- the use of *IPv6 care-of address* to identify *Mobile Hosts*.

Mobile-IPv6-capable hosts, use their *IPv6 care-of address* as the source of every packet they send and carry their permanent *Home IPv6 address* into a *Home address* destination options header. In order to be in line with *Mobile IPv6* specification, the *Cellular IPv6* control packets (`route-updates` and `paging-updates`) are sent uplink with source address the *Mobile Host*'s *IPv6 care-of address*.

On the reverse direction, *IPv6* packets destined to a *Mobile Host* reach the *Cellular IPv6 Gateway* in two alternative structures:

- *IPv6-encapsulated.*

  This means that the sender is not aware of the recipient *Mobile host*'s current *care-of address*, and sends the packet with destination its *Home IPv6 address*.

  This packet is normally routed to the *Mobile Host*'s *Home Network*, where it is intercepted by the local *Home Agent* which next encapsulates and sends the packet to the *Mobile host*'s current *care-of address*.

- Carrying an *IPv6* routing header.

  This means that the sender has a fresh binding for the recipient *Mobile Host* and sends the packet directly to its current *care-of address*. In this case, the sender maps the *Mobile Host*'s *Home IPv6 address* as the last entry in the routing header, while the *Mobile Host*'s current *care-of address* is mapped as second-to-last.

Packets addressed to a *Mobile Host* will be routed towards the *Cellular IPv6 Gateway/Router* using prefix-based routing. Next, *Cellular IPv6* host-based routing into the *Cellular IPv6 Access Network* will forward packets to the *Mobile Host*, through the *Base Station* that it is currently attached to.

You can get this information and more in this page[8].

4.2. Configuring nodes

A *Cellular IPv6* network is composed of different kinds of nodes. The nodes can be *Gateways* (top of the *Cellular IPv6* network), *Base Stations* (leaves of the *Cellular IPv6* tree) or *intermediate nodes*.

The configuration is shown in the next figure:

Figure 3

[8] `http://cipv6.intranet.gr/`

First you need to download the sources from this site[9] and compile them with `make`. A typical directory might be `/usr/local/`.

You also have to install the `Libpcap with LSF (Linux Socket Filtering)` and the `iproute2` packages in the PCs.

To configure the *Gateway*, the *Base Stations* and the *intermediate nodes*, you have to edit the `/usr/local/cipv6-1.1/cipnode6/cipnode6.conf` file. This file is well documented and you only have to specify the *IPv6 address* for each node.

The *Gateway* configuration file in our testbed looks like this:

```
esy
```

A *Base Station* configuration file is:

```
esy
```

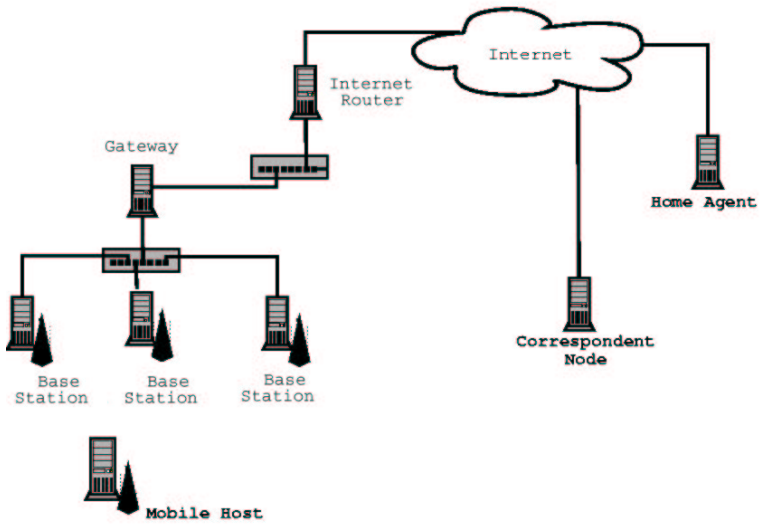
To configure the *Mobile host*, you have to edit the `/usr/local/cipv6-1.1/cipmobile6/cipmobile6.con`
For example, in our testbed:

```
esy
```

All the nodes (*Gateway* and *Base Stations*) in the *Cellular IPv6* network belong to the same subnetwork. Their subnetwork prefix is `fec0:0:0:2`.

You should check your kernel IP routing table is correctly configured.

[9] `http://cipv6.intranet.gr/`



4.3. Starting up

Once we have configured all nodes (*Gateway*, *Base Station* and *Mobile Host*), we only have to execute the daemons. In nodes which aren't the *Mobile Host*:

`/usr/local/cipv6-1.1/cipnode6/cipnode6`

In the *Mobile Host*:

`/usr/local/cipv6-1.1/cipnode6/cip6`

When the *Mobile Host* is registered in the *Cellular IPv6* network, it automatically configures its *Care-of adress*. While it is migrating from one *Base Station* to other *Base Station*, the routing caches in the *Cellular IPv6* network nodes are updated.