National Institute of Standards and Technology

The Network Simulator NS-2 NIST add-on

IEEE 802.21 model (based on IEEE P802.21/D03.00)

January 2007

1 Table of Content

2 Glossary

AP – Access Point
BS – Base Station
CN – Correspondent Node
ND – Neighbor Discovery
MIH – Media Independent Handover
RA – Router Advertisement
RS – Router Solicitation
SAP – Service Access Point

3 Overview

This document presents an MIH implementation for NS-2 developed for the Seamless Mobility project. It also provides an overview of the mobility extensions made to the standard release of NS-2.

Note: Due to the limitations of NS-2 in the evaluation of mobility scenarios, this release contains numerous modifications of the standard release. Workarounds were found to be able to simulate layer 2 and layer 3 handovers. We consider that the reader must be familiar with the details of NS-2 especially in the following areas:

- **Hierarchical addressing**
- **Understanding of the design and implementation of wireless nodes.**
- **Knowledge of wireless networks and their technologies (IEEE 802.11, IEEE 802.16) and mobility protocols**.

4 MIH implementation

The modified version of ns-2.29 contains an implementation of MIHF based on the draft 3 of IEEE 802.21 specifications. It is a platform to evaluate the performances and find problems that could arise due to the definition of the primitives. It also serves to evaluate different handover decision engines.

4.1 Architecture and internals

4.1.1 Design overview

Figure 1 represents a high level view of the MIHF interaction with the different components of the node. The MIHF is implemented as an Agent and therefore can send layer 3 packets to remote MIHF. The MIHF contains the list of local interfaces to get their status and control their behavior. The MIH User is also implemented as an Agent and registers with the MIHF to receive events from local and remote interfaces.

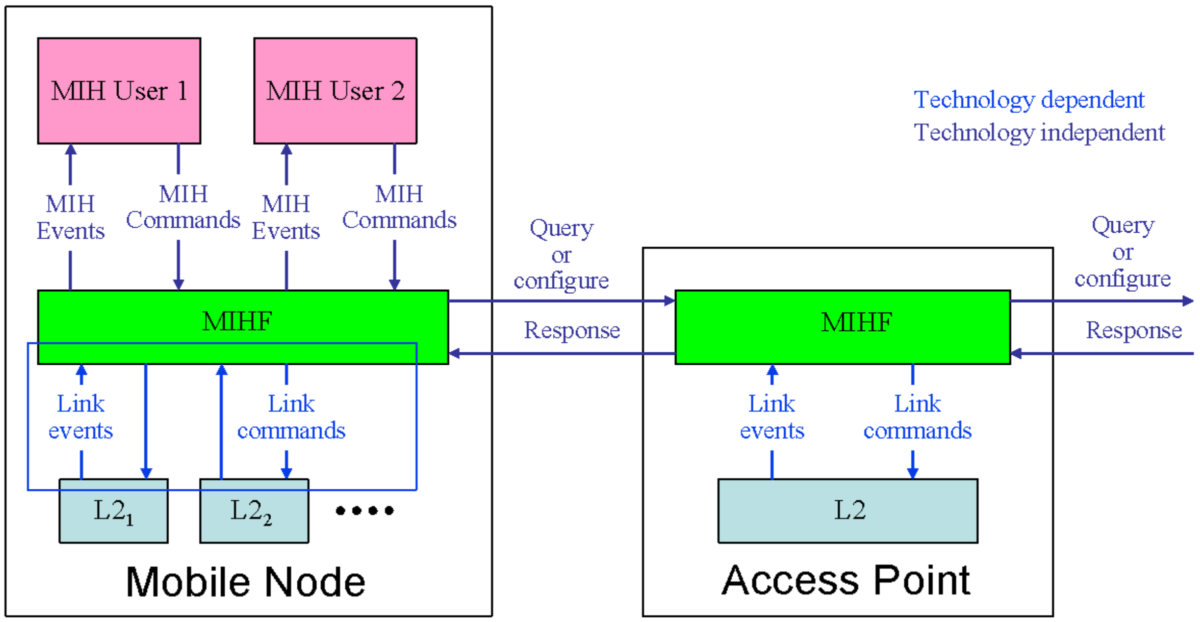

Figure 1: MIH design overview

The cross layer information exchange has been added to the NS-2 by modifying the MAC layer and linking the MIHF to the MAC layers via TCL.

4.1.2 MIH Function

As mentioned earlier, the MIHF extends the class Agent defined in NS-2. This allows each instance to send and receive packets at layer 3. The files related to the MIH implementation are located in the hsntg subdirectory of the distribution. The MIHAgent is at the center of the implementation. It communicates with both the lower layers (i.e. MAC) and the higher layers (i.e. MIH Users).
The class handles the list of MIH Users and their registration information. It also takes care of handling the communication with remotes MIHFs. Finally it provides the mapping between the media independent interface (MIH_SAP) and the media dependent interface (MIH_LINK_SAP and media specific primitives). Figure 2 shows the key components of the MIHF implementation.

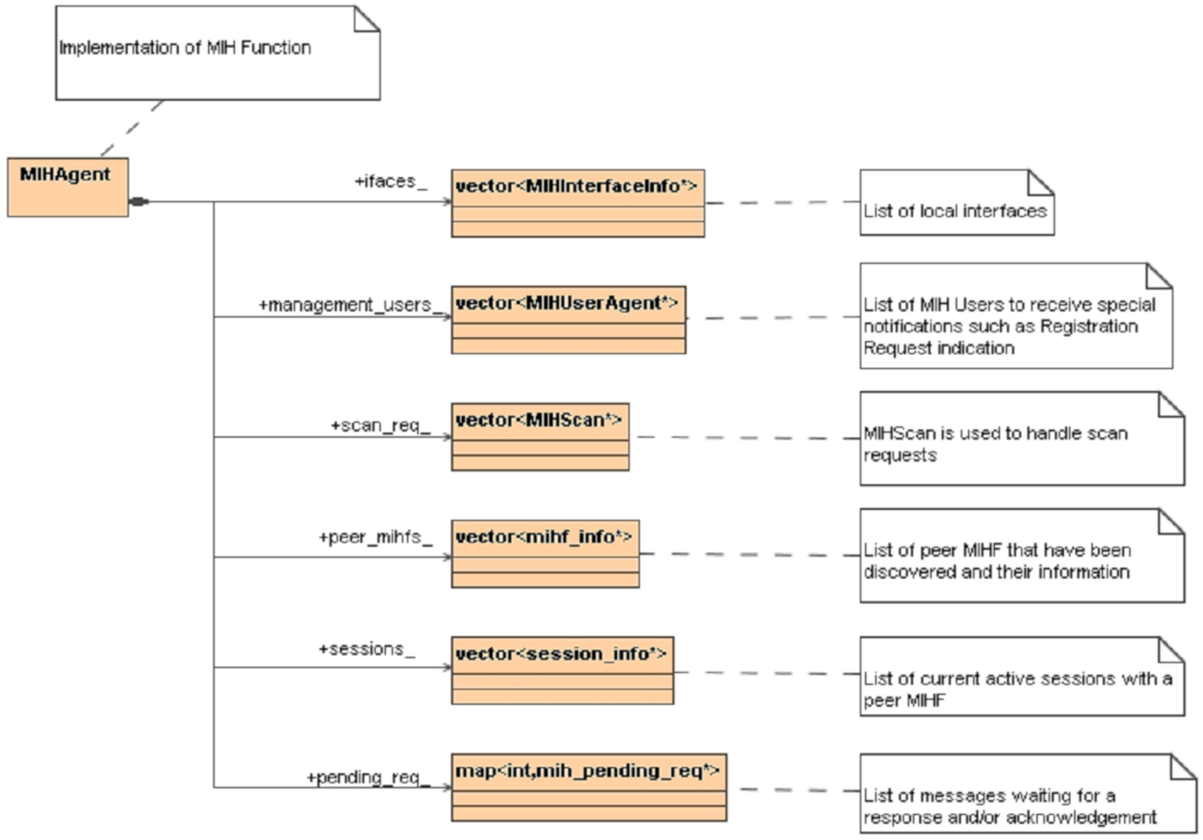
Implementation of MIH Function
MIHAgent
+ifaces_
vector<MIHInterfaceInfo*>
List of local interfaces
+management_users_
vector<MIHUserAgent*>
List of MIH Users to receive special notifications such as Registration Request indication
+scan_req_
vector<MIHScan*>
MIHScan is used to handle scan requests
+peer_mihfs_
vector<mihf_info*>
List of peer MIHF that have been discovered and their information
+sessions_
vector<session_info*>
List of current active sessions with a peer MIHF
+pending_req_
map<int,mih_pending_req*>
List of messages waiting for a response and/or acknowledgement

Figure 2: MIHF design

4.1.3 MIH User

MIH Users are entities that make use of the MIHF functionalities in order to enhance user performances by optimizing handovers. Since there are an infinite number of implementations depending on the user preference or network policies, the implementation provides an abstract class MIHUser that can be easily extended.

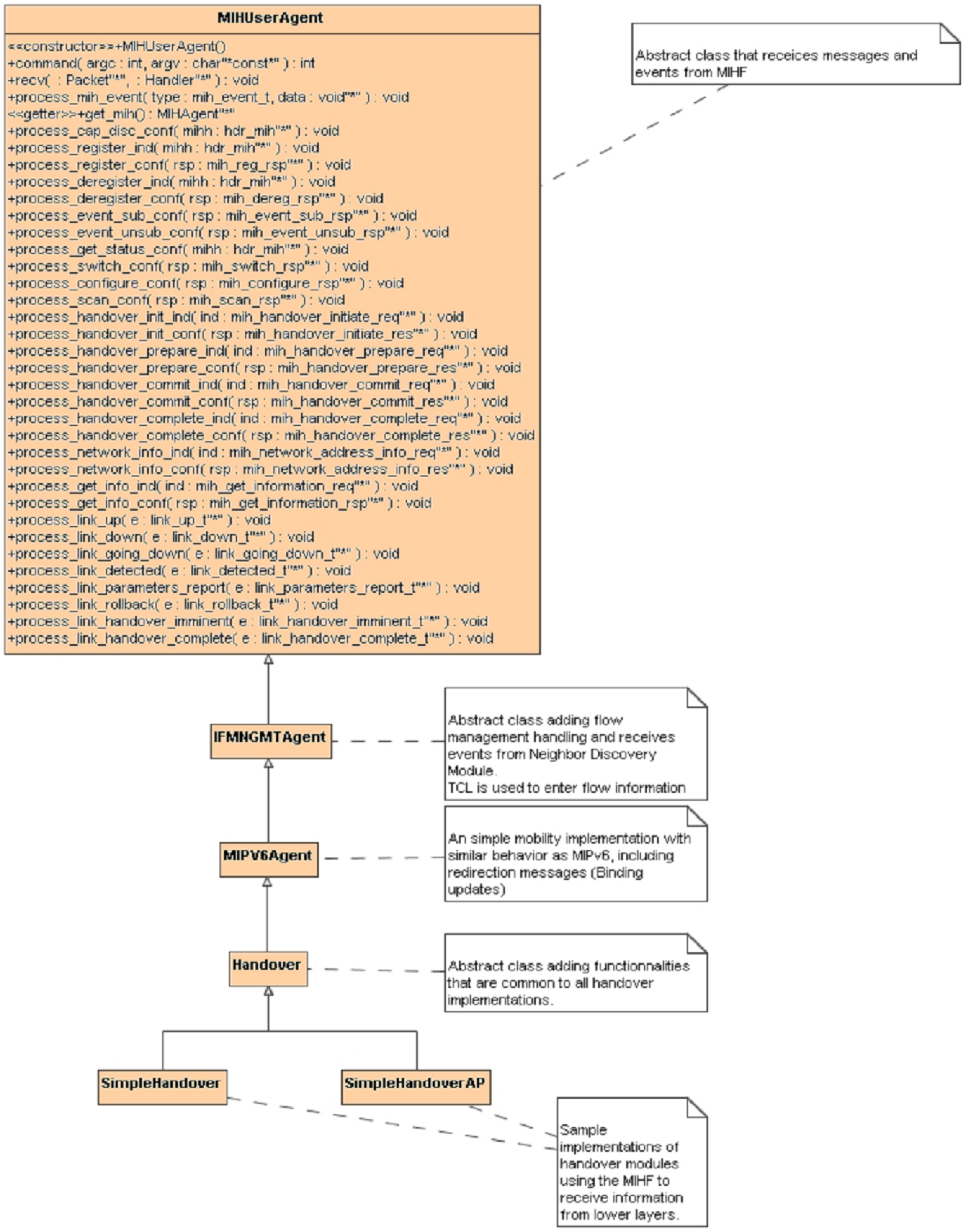
MIHUserAgent
<<constructor>>+MIHUserAgent()
+command(argc : int, argv : char"*const*") : int
+recv(: Packet"*", : Handler"*") : void
+process_mih_event(type : mih_event_t, data : void"*") : void
<<getter>>+get_mih() : MIHAgent"*"
+process_cap_disc_conf(mihh : hdr_mih"*") : void
+process_register_ind(mihh : hdr_mih"*") : void
+process_register_conf(rsp : mih_reg_rsp"*") : void
+process_deregister_ind(mihh : hdr_mih"*") : void
+process_deregister_conf(rsp : mih_dereg_rsp"*") : void
+process_event_sub_conf(rsp : mih_event_sub_rsp"*") : void
+process_event_unsub_conf(rsp : mih_event_unsub_rsp"*") : void
+process_get_status_conf(mihh : hdr_mih"*") : void
+process_switch_conf(rsp : mih_switch_rsp"*") : void
+process_configure_conf(rsp : mih_configure_rsp"*") : void
+process_scan_conf(rsp : mih_scan_rsp"*") : void
+process_handover_init_ind(ind : mih_handover_initiate_req"*") : void
+process_handover_init_conf(rsp : mih_handover_initiate_res"*") : void
+process_handover_prepare_ind(ind : mih_handover_prepare_req"*") : void
+process_handover_prepare_conf(rsp : mih_handover_prepare_res"*") : void
+process_handover_commit_ind(ind : mih_handover_commit_req"*") : void
+process_handover_commit_conf(rsp : mih_handover_commit_res"*") : void
+process_handover_complete_ind(ind : mih_handover_complete_req"*") : void
+process_handover_complete_conf(rsp : mih_handover_complete_res"*") : void
+process_network_info_ind(ind : mih_network_address_info_req"*") : void
+process_network_info_conf(rsp : mih_network_address_info_res"*") : void
+process_get_info_ind(ind : mih_get_information_req"*") : void
+process_get_info_conf(rsp : mih_get_information_rsp"*") : void
+process_link_up(e : link_up_t"*") : void
+process_link_down(e : link_down_t"*") : void
+process_link_going_down(e : link_going_down_t"*") : void
+process_link_detected(e : link_detected_t"*") : void
+process_link_parameters_report(e : link_parameters_report_t"*") : void
+process_link_rollback(e : link_rollback_t"*") : void
+process_link_handover_imminent(e : link_handover_imminent_t"*") : void
+process_link_handover_complete(e : link_handover_complete_t"*") : void
Abstract class that receices messages and events from MIHF
IFMNGMTAgent
Abstract class adding flow management handling and receives events from Neighbor Discovery Module. TCL is used to enter flow information
MIPV6Agent
An simple mobility implementation with similar behavior as MIPv6, including redirection messages (Binding updates)
Handover
Abstract class adding functionnalities that are common to all handover implementations.
SimpleHandover
SimpleHandoverAP
Sample implementations of handover modules using the MIHF to receive information from lower layers.

Figure 3: MIH User class hierarchy

As show inFigure 3, the MIHUser sends commands to and receive events/messages from the MIHF. To enhance usability, the implementation also provides a series of abstract classes that contain commonly used functionality. The IFMNGMT provides flow management functions. Using the TCL, the user can register the flows that are being used in the node. This facilitates the handover module in finding the flows that needs to be redirected. It also receives events from the ND agent when a new prefix is detected or when it expires. The MIPV6Agent adds the redirection capability to the MIH User. When a flow needs to be redirected, a message can be sent to the source node to inform him of the new address or interface to use. Finally, the Handover class provides a template for handover modules and computing a new address after successful handover.

Samples of handover modules are located in the hsntg/ and hsntg/802_21 subdirectories.

4.2 Configuration

Adding the MIHF to a node

```
set mihf [$node install-mih]
```

Linking the MIHF to the lower layers

```
set mac [$node set mac_(0)] ;#get the MAC element
$mac mih $mih ;#link the mih to the MAC layer
$mih add-mac $mac ;#add the MAC layer to the MIH
```

Adding a handover module

```
set handover [new Agent/MIHUser/IFMNGMT/MIPV6/Handover/Simple]
$node install-ifmanager $handover
$handover connect-mih $mih
```

Linking the ND and MAC modules to handover module

```
set nd [$node install-nd] ;# create ND agent
$handover nd_mac $nd $mac ;# inform the IFMNGMT agent about the
        relationship between the ND module and the MAC layer.
```

To support flow redirection, the CN (i.e. the node with which the MN is communicating) must be able to receive messages from the MIPV6. The following command must be including in all CNs.

```
$cn install-default-ifmanager
```

Default values for the MIH parameters are located in the file tcl/lib/ns-hsntg-nist.tcl.

5 MAC layer support for MIH

The MAC layers have been modified to include the MIH_LINK_SAP functions and to handle trigger generation. This section explains the modification to the classes.

5.1 Class Mac

The Mac class is the superclass of all the MAC implementations. The MIH_LINK_SAP has been added to this class so that the MIH handles Mac objects as much as possible.

5.2 Class Mac802_11

This class has been enhanced to support management frames.

5.2.1 Supported events

It supports the following events:

- Link UP
- Link Down
- Link Going Down
- Link Detected
- Link Event Rollback
- Link Parameters Report
- Link Handover Imminent
- Link Handover Complete

Therefore the bitmap mask for the MAC 802.11 is 0x1BF.

5.2.2 Supported commands

The following link commands are supported:

- Link Event Subscribe
- Link Event Unsubscribe
- Link Configure Threshold
- Link Get Parameters

The bitmask for the command list is 0xF

5.3 *Class Mac802_16*

5.3.1 Supported events

- Link UP
- Link Down
- Link Going Down
- Link Detected
- Link Event Rollback
- Link Parameters Report
- Link Handover Imminent
- Link Handover Complete

Therefore the bitmap mask for the MAC 802.16 is 0x1BF.

5.3.2 Supported commands

The following link commands are supported:

- Link Event Subscribe
- Link Event Unsubscribe
- Link Configure Threshold
- Link Get Parameters

The bitmask for the command list is 0xF

5.4 *Class Mac802_3*

5.4.1 Supported events

- Link UP
- Link Down

Therefore the bitmap mask for the MAC 802.16 is 0x3.

5.4.2 Supported commands

The following link commands are supported:

- Link Event Subscribe
- Link Event Unsubscribe

The bitmask for the command list is 0x3

5.5 Class UMTS

5.5.1 Supported events

- Link UP
- Link Down

Therefore the bitmap mask for the MAC 802.16 is 0x3.

5.5.2 Supported commands

The following link commands are supported:

- Link Event Subscribe
- Link Event Unsubscribe

The bitmask for the command list is 0x3

6 Mobility extensions for NS-2

In addition to the MIH implementation, the mobility model includes enhancements to support handovers in NS-2. This includes:

- Integration of multiple technologies (UMTS, Bluetooth, 802.16) to allow for heterogeneous handovers.
- Modification of default implementation (802.11) to support handovers.
- Define a generic design for nodes with multiple interfaces.
- Support for subnet discovery and change of address.

6.1 Support for multiple interfaces

In order to evaluate handover in heterogeneous environment, we integrated multiple packages providing additional technologies. The following is a list of technologies added to the package:

- UMTS: the source code is based on the EURANE code (http://www.ti-wmc.nl/eurane/). The modification includes support for hierarchical addressing.
- Bluetooth: package from the University of Cincinnati (http://www.ececs.uc.edu/~cdmc/ucbt/) based on version ucbt-0.9.8.2a.
- IEEE 802.16: developed internally and focusing on the mobility aspects of the technology (802.16e).
-

The difficulty encountered is that support for multiple interfaces is not intuitive in NS-2. Furthermore, external packages do not necessarily follow the same node structure as the one defined in the basic model. For example the routing algorithms are different. For this reason, a workaround has been found allowing each technology to work independently from the others.

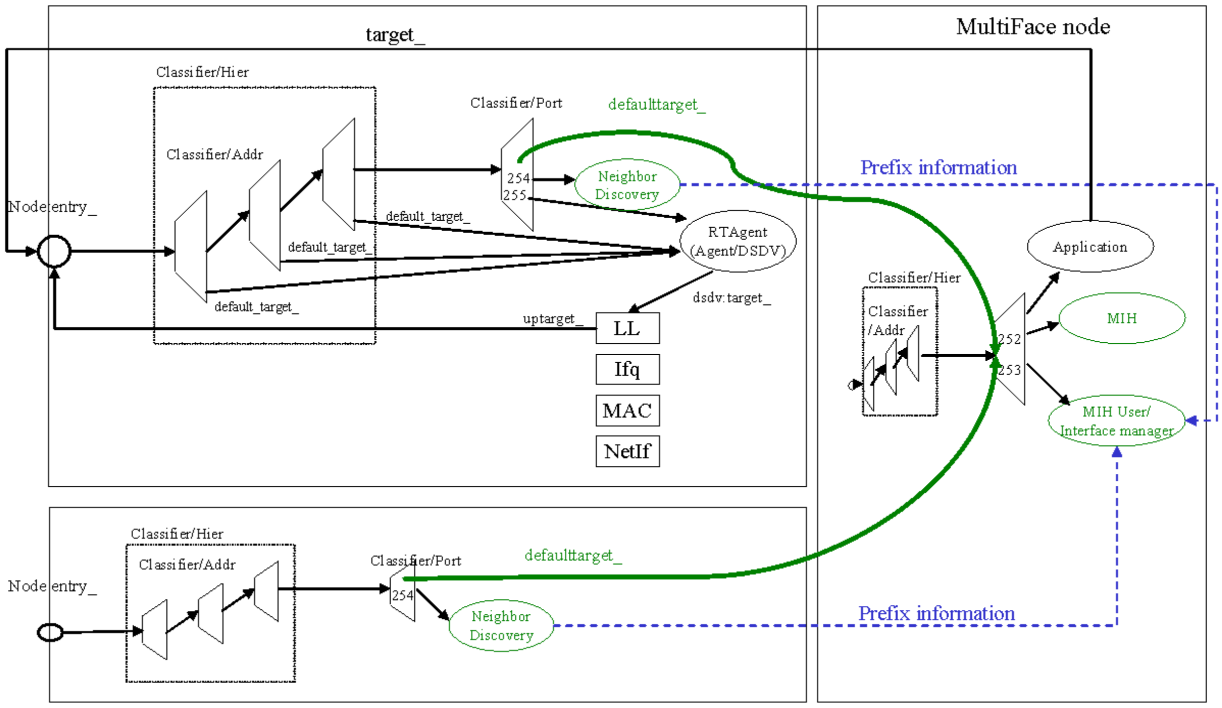

Figure 4: Multiple interfaces node design

As show on **Error! Reference source not found.**, the multiFace node is a virtual node linking nodes of similar or different technologies. The other nodes are considered interfaces for the multiFace node. Let's define them as interface nodes. An ND agent located in each node allows for layer 3 movement detection (new and expired prefix) and notifications are sent to the interface manager (IFMNGMT). The MIH located in the MultiFace node is linked with each MAC object of the interface node. The application target_ object is dynamically assigned to the entry of the node chosen to send the traffic.
With this design, we have been able to simulate handovers between IEEE 802.11, IEEE 802.16 and UMTS.

6.2 Layer 2 mobility

Layer 2 mobility is provided by the lower layer. In the model provided, IEEE 802.11 and IEEE 802.16 allows for switching AP/BS. Note that in NS-2, each AP/BS is on a different subnet (i.e. domain or cluster) and therefore will require a

Layer 3 handover. For information about Layer 2 mobility, refer to the documentation specific to each media.

6.3 Layer 3 mobility

6.3.1 Neighbor Discovery

This section contains a short overview of the functionalities provided by the ND module. For more information refer to the Neighbor_Discovery documentation manual.
The ND module is used to provide Layer 3 movement detection. In the network, APs and BSs periodically send RAs to inform the MNs about the network prefix. In our scenarios, the prefix is the address of the AP/BS. The RA is sent using broadcast messages when supported (802.11, 802.16) or unicast otherwise (ethernet). The ND agent located in the MN receives these RAs and determine if the message contains a new prefix and inform the interface manager. A timer is associated with the lifetime of the prefix. When the MN losses its connection with the AP/BS, the prefix expired and a notification is sent to the interface manager. The implementation also supports RS to enable a MN to discover new AP/BS after a handover.

By convention, we assign the address of the AP/BS with a node address of 0. For example 3.0.0 and 4.0.0 are valid addresses for these nodes. MNs are assigned addresses with node IDs other than 0 (i.e 3.0.1, 4.0.12).

6.3.2 Routing algorithms and address update

In NS-2 the routing is done in different ways depending on the type of nodes. In a wired network, the routes are computed at the initialization and the classifiers are updated. In a wireless network, a routing agent takes care of routing the packets in a more dynamic way since the topology and routing can change. The project does not focus on ad-hoc networks; therefore, a routing algorithm supporting infrastructure more is required. The first solution was to modify the DSDV routing to behave differently and not send too many packets. The second solution was to use the NOAH protocol (http://icapeople.epfl.ch/widmer/uwb/ns-2/noah/). We modified it to handle node address change.

During a simulation, a MN can change network. When switching AP/BS, the node needs to be reachable via a new address. This functionality is not included in NS-2.29 and we propose the following workaround. After the execution of a layer 2 handover, the ND protocol is used to receive the new prefix information. When receiving a new prefix, we compute the new node address with the following formula:

```
        new_address = (old_address & 0x7FF)|(prefix & 0xFFFFF800);
```

We can note that the node ID will be the same in the new network. Only the domain and cluster is updated.

To ensure proper behavior, the following rules must be followed:

- Determine the maximum number of nodes that can be within a single network. For example if at the beginning of the simulation there are 2 wireless networks (defined by 2.0.0 and 3.0.0) with 2 MNs each, it is possible that two nodes move to the other network, and therefore the maximum number of nodes in the network is 5 (4MNs + AP).
- Assign the address with unique node ID for each nodes. Using the example above, a possible configuration is 2.0.1, 2.0.2, 3.0.3, and 3.0.4.

Figure 5 illustrates the change of address

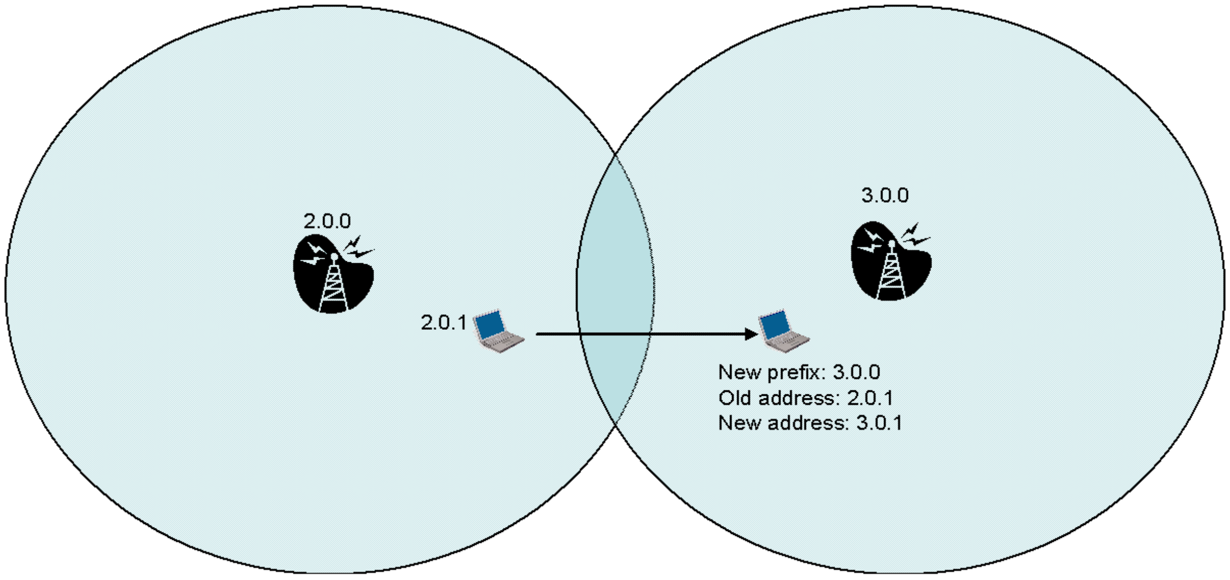

Figure 5: Example of node address change

After computing the new address, additional changes are required:

- Update the address of the node
- Since each agent located in the node has a cache of the node address they also need to be updated. This is especially valid for the routing algorithm.
- Update the base station information in the routing protocol.

Using the functionalities provided by MIPV6Agent, the MN can send a redirect message to the CN to inform the node about its new address.

Examples can be found in the tcl/hsntg subdirectory.