# A Comprehensive MIPv6 Based Mobility Management Simulation Engine for the Next Generation Network

Faqir Zarrar Yousaf, Christian Müller and Christian Wietfeld
Communications Networks Institute (CNI)
Dortmund University of Technology, Germany
{faqir.yousaf, christian5.mueller, christian.wietfeld}@tu-dortmund.de

## ABSTRACT

Mobility management is one of the core requirements of the IPv6 based NGN to provide seamless handover services to mobile entities. In this regard, IETF has proposed protocols like Mobile IPv6 (MIPv6), Fast MIPv6 (FMIPv6) and Hierarchical MIPv6 (HMIPv6) to provide efficient mobility services. Due to the lack of deployment, there is not enough empirical data available to determine the efficacy of these protocols in the context of NGN. For this purpose, there is a need to develop an *accurate* and *reliable* simulation framework with the help of which the users can quickly test the validity of such protocols under a variety of network and load conditions. The simulation framework should also allow the user to *rapidly* develop and test *prototype* solutions and algorithms for attaining optimum performance goals.

In this paper we present the logic, design and performance results of an *OMNeT++* based Mobility Management Simulation Engine for IPv6 networks, which enables the user to accurately test the performance of protocols such as MIPv6, FMIPv6 and HMIPv6, and their possible combination on a single unified platform.

## Categories and Subject Descriptors

I.6.4 [**Simulation and Modeling**]: Model Validation and Analysis; I.6.5 [**Simulation and Modeling**]: Model Development—*Modeling methodologies*; I.6.6 [**Simulation and Modeling**]: Simulation Output Analysis

## General Terms

Measurement, Performance, Design, Reliability, Experimentation, Verification

## Keywords

Protocol Simulation, MIPv6, FMIPv6, HMIPv6, CARD, OMNeT++, C++ Discrete Event Simulation.

## 1. INTRODUCTION

The Next Generation Networks (NGN) is expected to provide ubiquitous communication services to mobile entities moving in an IPv6 based heterogeneous wireless network environment. The provisioning of efficient mobility management services to mobile entities with varied QoS requirements, and undergoing frequent handovers in such a technologically diverse environment is an imposing challenge. This calls upon designing efficient mobility management solutions that would provide accurate location updates (location management) and seamless handover services (handover management).

To address such a scenario, IETF has specified MIPv6 protocol [2] that provides location management and handover management services to single interface Mobile Nodes (MN). However, it incurs a high handover delay and signaling load making it unsuitable for fulfilling the operational and functional requirement of NGN.

In order to circumvent the performance deficiencies of MIPv6, IETF has proposed protocols such as HMIPv6 [6] and FMIPv6 [3] to reduce the signaling overhead and handover delay respectively. Both HMIPv6 and FMIPv6 are extended versions of the base MIPv6 protocol, and rely on the basic protocol constructs defined by it.

As NGN is still in its evolutionary stage, there are no commercial deployments of MIPv6 or its derivative protocols (i.e., FMIPv6 and HMIPv6), thereby making it difficult to assess their performance boundaries, and hence applicability within the NGN framework. It thus becomes necessary to develop a simulation environment, where the users can accurately and reliably test and analyze the performance of these protocols under a variety of network and load conditions, and hence draw tangible conclusions.

We present in this paper the logic, design and performance results of a *Mobility Management Simulation Engine for IPv6 (MMSEv6)*, which is a MIPv6 based simulation framework developed in OMNeT++ [5] and integrated into the INET framework. The MMSEv6 framework is composed of several protocol implementations, and provides a single comprehensive simulation platform enabling the user to simulate MIPv6, FMIPv6 and HMIPv6 protocols respectively. The conceptual design of the MMSEv6 is shown in Figure 1. The MMSEv6 has been developed by extending our previous simulation model called Extensible MIPv6 (xMIPv6) [9], which was developed and tested to accurately simulate the performance of MIPv6 protocol only. The xMIPv6 simulation model has already been released to the public [8] and is being used widely.

An important consideration while designing the MMSEv6 framework was to enable researchers and practitioners to not only accurately simulate and analyze the MIPv6, FMIPv6 or HMIPv6 protocols under varying network and load conditions, but also enable them to rapidly develop prototype solutions and/or algorithms for further optimizing the MIPv6 based mobility management solutions. The accuracy has been ensured by strictly conforming to the requisite details as prescribed in the related IETF standards


OMNeT++ 2010, March 15-19, Torremolinos, Malaga, Spain.

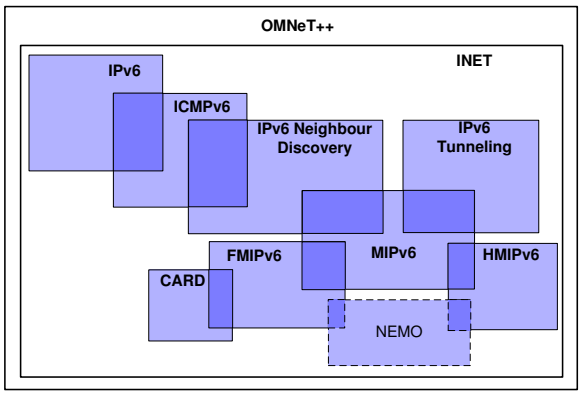



**Figure 1: MMSEv6 Design Conceptual Layout**

and testing the operational and performance accuracy of the MIPv6 model against a real MIPv6 test bed [9].

The rest of the paper is organized as follows. Section 2 gives a brief overview of the FMIPv6 and HMIPv6 protocols, whereas the implementation and design details of the MMSEv6 is discussed in Sections 3 and 4 respectively. Section 5 presents performance results of the MMSEv6 framework, and the concluding remarks are given in Section 6.

# 2. IMPLEMENTED PROTOCOLS

The MIPv6 protocol, which is based on *break-before-make* concept, is composed of several subprocesses. Each of the subprocess, such as *Movement Detection (MD) (inclusive of Layer-2 (L2) handover delay)*, *Duplicate Address Detection (DAD)* test [7], *Home Registration (HR)*, *Return Routability (RR)* test and *Correspondent Node Registration (CR)*, introduces a finite amount of delay adding to the total handover delay. The total MIPv6 handover latency ($T_{HO}$) can be expressed as:

$$T_{HO} = T_{MD} + T_{DAD} + T_{HR} + T_{RR} + T_{CR} \quad (1)$$

Of the above mentioned sub-processes, the MD and DAD subprocess contributes substantially to $T_{HO}$. DAD alone accounts for 1 second delay during which time no packets are sent. The $T_{HO}$, depending on the topological distance of the MN from its *Home Agent (HA)* and *Correspondent Node (CN)*, is typically in excess of 1.5 seconds. During the MIPv6 handover process, the MN is practically disconnected from the network resulting in packet losses. The high handover latency and the resulting packet losses makes MIPv6 unsuitable for providing seamless handover services.

## 2.1 FMIPv6 Overview

To provide seamless handover services, IETF has proposed FMIPv6 protocol [3], which is an extension of the MIPv6 protocol. FMIPv6 is designed to reduce handover latency and the resulting packet losses. The handover latency is reduced by enabling the MN to discover the *New Access Router (NAR)*[1] well in advance to the actual execution of the handover. This implies that the MN is able to perform delay incurring tasks like *movement detection* and *address-autoconfiguration* while the MN is still connected to its *Present Access Router* (PAR). The packet losses are reduced by buffering the packets till the MN completes its handover from PAR to NAR.

[1]This is also called the Router Discovery Process

FMIPv6 operation typically starts when the MN receives some Layer 2 trigger in the form of beacon signals from the in-range *Access Points (AP)*. The MN will extract the L2-IDs (i.e., MAC addresses) carried by these signals and request the PAR via a *Router Solicitation for Proxy (RtSolPr)* message to resolve the *identity* and *capabilities* of the *Candidate Access Router (CAR)* associated with the detected AP(s). The PAR will initiate the process of *Reverse Address Translation (RAT)* and *Discovery of CAR Capabilities (DCC)* and the requisite information is then conveyed to the MN via the *Proxy Router Advertisement* (PrRtAdv) message. Thereafter, the MN will select an appropriate NAR and autoconfigure [7] a prospective *New Care-of-Address (NCoA)* based on the NAR's address prefix. The MN will then inform the PAR of its choice of NAR and the NCoA by sending a *Fast Binding Update (FBU)* message. Upon receiving the FBU, PAR will send a *Handover Initiate (HI)* message to NAR carrying the MN's NCoA. The NAR, after performing the DAD test on NCoA, will reply with a *Handover Acknowledgment (HAck)* message indicating the NAR's decision to accept or reject the handover request. As soon as the PAR receives a positive HAck, a binding is created between the previous CoA and the NCoA. A bidirectional tunnel is also established between the PAR and the NAR, and all subsequent packets will get tunneled towards the NAR where they will get buffered. In the meantime, the PAR will inform the MN of the handover decision by sending a *Fast Binding Acknowledgment (FBAck)* message.

As the MN enters the NAR domain, it will inform the NAR of its presence by sending a *Unsolicited Neighbor Advertisement (UNA)* message, and soon afterwards all the buffered and subsequent packets are forwarded to the MN. The MN then starts the normal MIPv6 handover process by sending bindings to its HA and CN(s) informing them of its NCoA. After successful bindings, the handover gets completed and the tunnel gets released. FMIPv6 specifies two handover modes namely *Predictive* and *Reactive* Handover modes, of which the Predictive mode is the default mode and is implemented in the MMSEv6.

It may be noted that the FMIPv6 specifications does not specify the mechanism to perform RAT and DCC functions and instead suggests the use of external protocols to achieve this end. The IETF prescribed *Candidate Access Router Discovery* (CARD) [4] is one such protocol that enables the MN to perform RAT and DCC while it is still connected to its PAR. The MMSEv6 incorporates full implementation of the CARD protocol that is used by the FMIPv6 protocol operations.

## 2.2 HMIPv6 Overview

MIPv6 incurs a high signaling load in the sense that it has to exchange binding messages with its HA and all the CNs every time a handover takes place. In case of higher *Round Trip Time (RTT)* between the MN and the respective HA and the CN(s), the binding will take more time adding to the overall handover latency.

To address this issue, IETF has proposed HMIPv6 [6] protocol by extending the MIPv6 protocol. HMIPv6 defines a local anchor point called *Mobility Anchor Point (MAP)*, which is a router located above the ARs in the network hierarchy. The MAP router limits the exchange of mobility messages outside its local MAP domain by eliminating the need of the MN to exchange binding information with the HA and CN during every handover instance. A MN will detect that it has entered a MAP domain by the information carried by the receiving *Router Advertisements (RA)*, which contains prefix information about the advertising AR and about available MAPs in the domain. Based on this information, the MN will autoconfigure a *Local CoA (LCoA)* and a *Regional CoA (RCoA)*, and will register itself with the MAP by sending both these addresses

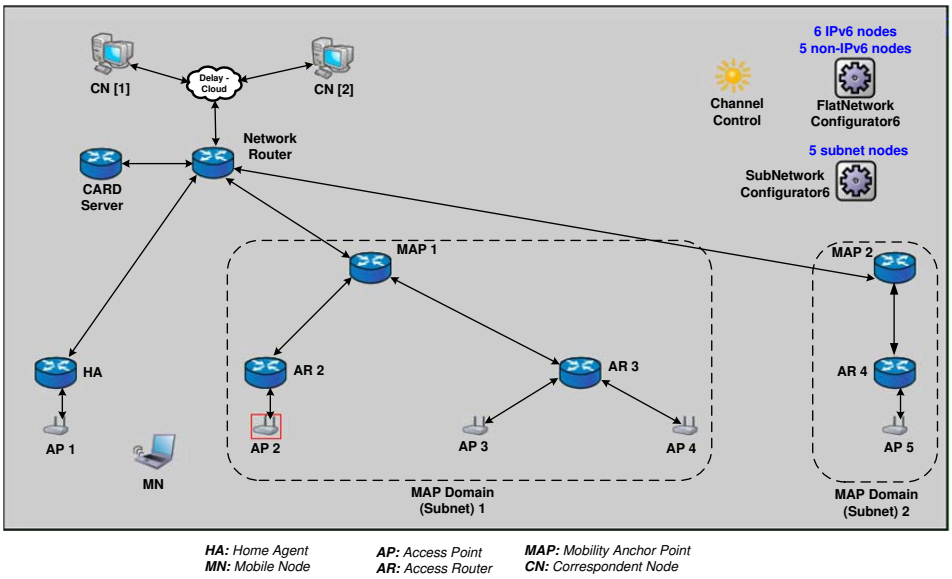



**Figure 2: Hierarchical Test Network with Multiple Subnetworks**

encapsulated into a *Local Binding Update (LBU)* message to the MAP. The MAP, after performing DAD on the RCoA, will create a binding between the LCoA and RCoA, establish a bi-directional tunnel with the MN's LCoA and send a *Local Binding Acknowledgment (LBA)* to the MN. The MN upon receiving LBA from the MAP will register its RCoA with its HA and CN as per MIPv6 protocol rules. As a result, all packets which are destined to the MN are now sent to the MAP which tunnels them to the LCoA of the MN via the MAP-MN bi-directional tunnel.

The strength of this approach can be observed when the MN changes its association within the MAP domain. As soon as the MN associates to a new AR, it only auto-configures a new LCoA and sends a LBU to the MAP with the new LCoA and the previously assigned RCoA. The MAP will update the *Binding Cache (BC)* by associating the new LCoA with the MN's RCoA, and together with this update the bi-directional tunnel to the MN. Since the RCoA of the MN remains unchanged within a MAP domain, it does not need to send any binding messages to the HA and the CN(s). In other words, the movement of the MN is transparent to the HA and the CN(s) within a MAP domain. This will account not only towards the reduction of signaling load, but will also reduce the signaling delay due to high RTT.

# 3. MMSEV6 MODEL SUMMARY

The MMSEv6 is developed by extending the xMIPv6 simulation model [9, 8], which in turn has been developed by building on top of the existing INET framework. For instance, the nodes are derived from the existing modules and the protocol implementation makes use of the existing IPv6 and IPv6 Neighbor Discovery protocol implementation, but with necessary and non-intrusive modifications. The implementation details are provided in [9].

The CARD protocol mechanics are defined inside the *CARD* simple module. This module is integrated inside the *NetworkLayer6* compound module of the MN and the AR, and connected to the *IPv6* module as shown in Figure 4. On the other hand, being an extension of the MIPv6 protocol, the operational functions of the FMIPv6 and HMIPv6 protocols are integrated inside the *xMobileIPv6* simple module and the flag *hasFMIPv6Support()* or *hasHMIPv6Support()* will determine whether the MN should negotiate the handover according to the rules of the MIPv6 protocol, the FMIPv6 or HMIPv6 protocol. Required modifications has been made in the existing *IPv6NeighbourDiscovery* module especially in terms of modeling and processing of the RA message.

Apart from the *Binding Cache* (BC) and *Binding Update List* (BUL) implemented as part of xMIPv6 model [9], each HA, MN and the AR maintains a *CAR Table* in connection to the CARD protocol. In addition to the standard MIPv6 functions, the MMSEv6 extends the xMIPv6 simulation model to incorporate the following additional functions;

- Sensing of the SNR of the received signal by the MN.
- Channel scanning for in-range APs.
- RAT and DCC functions as per the CARD protocols rules.
- Auto-configuration of a NCoA by the MN.
- Establishing of a bi-directional tunnel between the PAR and the NAR (for FMIPv6), and between the MAP and the MN (for HMIPv6).
- Packet forwarding from PAR towards NAR, and its respective buffering.
- Packet forwarding from NAR to MN after establishing IP connectivity.
- Auto-configuration of two different addresses namely LCoA and RCoA for a single interface.

## 3.1 MMSEv6 Nodes

FMIPv6 protocol uses the same nodes as defined for MIPv6, but with added functionalities and features; especially for the AR. The nodes are implemented by sub-classing from the existing modules of the INET framework with new additional modules. The boolean parameters namely *isMobileNode*, *isHomeAgent*, *isAccessRouter*,

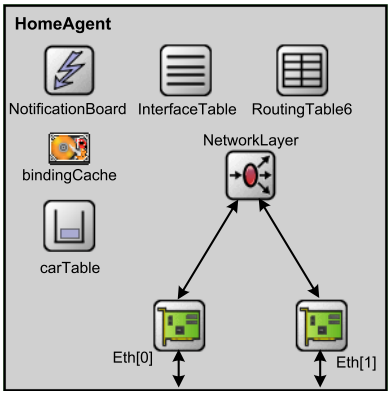



**Figure 3: Architecture of MMSEv6's Home Agent/Access Router**

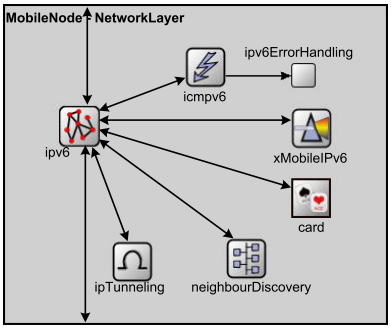



**Figure 4: Architecture of Mobile Node's Network Layer**

*isMap* and *isRouter* have been added within the simple module of *RoutingTable6* to differentiate between the definition and functions of these nodes.

The description of the relevant nodes is detailed below:

### *3.1.1 Mobile Node*

The MN is an extended version of the *WirelessHost6* defined in [9], and due to its support for FMIPv6 operation is called *FastMobileNode6*. The main difference is that the *Ieee80211Radio* module of the WLAN interface (*Ieee80211NicSTA*) has been replaced by the *SnrEval80211* and *Decider80211* simple modules. This enables the MN to sense and measure the power and SNR of the received signals, and the MN will trigger the FMIPv6 operations when the *Received Signal Strength Indicator (RSSI)* falls below a certain specified threshold value. Besides the *BUL* module, the MN will contain a single instance of a *CARTable* and *CARD* simple modules. The details of these two modules will be discussed subsequently.

MNs maintain discovered address and capability information of CARs in the *LocalL2L3Cache* cache to avoid requesting the same information repeatedly. It also enables the MN to quickly select an appropriate target AR from the list of CARs whenever a handover is imminent.

### *3.1.2 Home Agent & Access Router*

Both the HA and the AR are essentially the same as both have been extended with a *CARTable* and *CARD* module to support CARD protocol. The *isHomeAgent* and *isAccessRouter* flag defined in the *RoutingTable6* module distinguishes the operation of the HA from that of an AR.

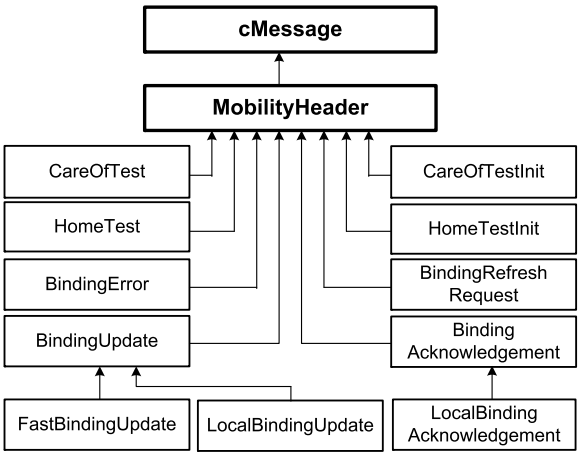



**Figure 5: FMIPv6 and HMIPv6 Mobility Messages Class Hierarchy**

### *3.1.3 CARD Server*

The CARD server is realized as a compound module called *CARDServer6* which is derived from the *Router6* module but with an additional CAR Table and CARD simple module to support CARD protocol operations [4].

### *3.1.4 Mobility Anchor Point*

Since a MAP acts like a local HA to the MN in a foreign domain, its structure and design is similar to the HA (see Section 3.1.2). The *isMAP* flag, defined in the *RoutingTable6* module, distinguishes it from the normal HA. The *BindingCache* module keeps track of the MN's movement within the MAP-domain and controls a bi-directional tunnel to the MN's current whereabout.

## 3.2 MMSEv6 Messages

In this section we will summarize the message constructs related to FMIPv6 and HMIPv6.

### *3.2.1 FMIPv6 Messages*

FMIPv6 protocol, besides using the MIPv6 specified messages, defines two additional mobility messages namely; the *FBU* and *FBAck*. As shown in Figure 5, they are derived from the *BindingUpdate* and *BindingAcknowledgement* message class defined for standard MIPv6 operation [9].

Besides, FMIPv6 also defines four new ICMPv6 messages. Two of these; namely *RtSolPr* and *PrRtrAdv* messages are classified with the Neighbor Discovery class of messages that are exchanged between the MN and PAR for RAT and DCC functions. The other two are inter-AR messages, namely *HI* and *HAck*, exchanged between PAR and NAR as introduced in Section 2.1. All of these messages are sub-classed from the *FMIPv6NDMessage* class, which in turn is sub-classed from the *ICMPv6Message* class pre-defined in the INET framework. The class hierarchy of these messages is shown in Figure 6.

### *3.2.2 CARD Messages*

As mentioned in Section 2.1, the CARD protocol is an independent protocol but is utilized by other mobility management protocols (FMIPv6 in our case) that may need to perform RAT and DCC functions. In MMSEv6, we have implemented CARD as a standalone protocol and FMIPv6 uses the relevant features of this protocol. CARD provides the RAT and DCC services by the ex-

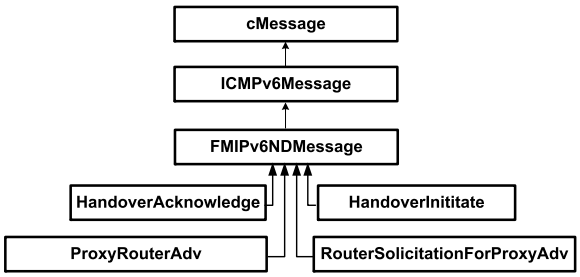



**Figure 6: FMIPv6 Neighbor Discovery Messages Class Hierarchy**

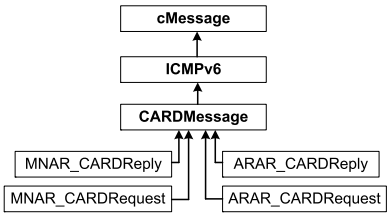



**Figure 7: CARD Messages Class Hierarchy**

change of *request* and *reply* messages between MN and PAR and between PAR and *neighboring* CARs (and/or CARD Server). The messages defined in the CARD protocol can be broadly categorized into the following two categories:

- *MN-AR Request/Reply* Messages
- *AR-AR Request/Reply* Messages

According to the CARD protocol specifications [4], the *MN-AR Request/Reply* messages are transported as *Internet Control Message Protocol* (ICMP) messages, whereas the *AR-AR Request/Reply* messages are *recommended* to be transported using the *Stream Control Transport Protocol* (SCTP), but is not mandatory. Since the implementation of SCTP is outside the scope of our work, both the MN-AR and AR-AR messages are modeled to be transported using ICMPv6, with the experimental ICMP-type header [4]. All these messages are derived from the INET's *ICMPv6Message* class. Figure 7 shows the inheritance diagram of the four main CARD messages.

| Sub Options | Interface MN-AR | Interface AR-AR |
|---|---|---|
| L2ID | yes | no |
| Address | yes | no |
| Capability Container | yes | yes |
| Preferences | yes | yes |
| Requirements | yes | no |
| Trusted Anchor | yes | no |
| Router Certificate | yes | yes |

**Table 1: Valid CARD Message Sub-options on Interfaces**

The CARD messages are appended with various sub-options, the exact type of which is determined by the type of interface the message is sent through. Table 1 shows the valid sub-options indicating their relevance to the particular interface. These sub-options are implemented as class *CARDSubOption*, which is derived from OMNeT++'s *cObject* class. *It may be mentioned that when used in conjunction with the FMIPv6 protocol, the MN-AR CARD Request/Reply messages are replaced by the FMIPv6 specified RtSolPr and PrRtrAdv messages.*

### *3.2.3 HMIPv6 Messages*

HMIPv6 protocol utilizes the MIPv6 specified messages with minor necessary modifications. For instance, the *LocalBindingUpdate* message is sub-classed from the existing *BindingUpdate* message class as shown in Figure 5. Besides the addition of the *M-field* in the *RouterAdvertisement* message class, HMIPv6 introduces a Neighbour Discovery message sub-option called MAP sub-option [6], which is appended to the *RouterAdvertisement* message carrying the available MAP(s) prefix(es) within a domain.

# 4. FUNCTIONAL SUMMARY OF MODULES IMPLEMENTING FMIPV6, HMIPV6 AND CARD

Since FMIPv6 and HMIPv6 are extended versions of the MIPv6 protocol, therefore the necessary functions and operations have been included into the *xMobileIPv6* simple module. The mechanics of the CARD protocol has been implemented by realizing two simple modules namely *CARD* and *CARTable*. A new module called *SubnetConfigurator6* has also been implemented to realize hierarchical network topologies for testing HMIPv6. The newly defined and modified modules are described below.

## 4.1 CARTable Simple Module

The *CARTable*, shown in Figure 3, is a standalone simple module implementing the CAR Table cache, which is a L2-L3 address mapping table and accessed during RAT and DCC functions. The *CARTable* module is implemented as an associative container, which maps the L2 IDs of the candidate APs with IP addresses and capabilities of the associated CARs. The *CARTable* also keeps and maintains individual CARs' capabilities information and related parameters. The CAR(s) information is retrieved by keying on the L2 ID. The conceptual arrangement of the *CARTable* is shown in Table 2.

| CAR Table | | |
|---|---|---|
| **AP ID** | **AR Information** | **Capabilities** |
| MAC Address<br>Channel No.<br>SNR (RSSI) | IP Address<br>IP Prefix<br>Prefix Length | Bandwidth<br>QoS related<br>parameters |

**Table 2: Conceptual Design of the CAR Table**

## 4.2 CARD Simple Module

The protocol functions of the CARD protocol is implemented inside the *CARD* simple module. All the relevant *MN-AR and AR-AR Request/Reply* messages and related timers are created, maintained and processed within this module as per the CARD protocol specifications [4]. As seen in Figure 4, the *CARD* module is located at the network layer and connected to the *IPv6* module. Thus the CARD messages are encapsulated as IPv6 datagrams and transported over the network.

At the time of initialization, each AR will update its local *CARTable* cache with its own [AP-ID, AR-Info] tuple information and convey it to the CARD Server via *unsolicited AR-AR CARD*

*Reply* message with relevant sub-options. In this way the CARD Server will maintain in its local CAR Table a repository of [AP-ID, AR-Info] tuple information of all the ARs within its domain. The ARs will periodically refresh the CARD Server with this information at the expiry of a predefined timer.

### *4.2.1 RAT & DCC Functions*

The WLAN interface of the MN (*Ieee80211NicSTA*) will continuously monitor and compare the strength of the received signal from the current AP as the MN moves across the network. When the signal strength becomes less than or equal to a certain specified threshold value, the MN will start the 802.11 scan process (*active or passive*) to discover the presence of in-range AP(s). During the scan process, the MN will extract the MAC address(es) of in-range AP(s) received in the *Beacon* or *Probe Response* frames and store them in the AP List defined in the *Ieee80211MgmtSTA* module of the WLAN interface. At the end of the scan process, the MN will append the discovered MAC address(es) as sub-options to the *RtSolPr* message and send it to the PAR. The PAR upon receiving the *RtSolPr* request from the MN will resolve the requested APs' L2 ID to the IP address of the associated CARs (RAT function). If specifically signaled in the request message, the PAR will also retrieve the CARs' capability parameters (DCC function) by checking its local CAR Table. In case the requisite information is not available locally, it will perform RAT and DCC by exchange of *AR-AR CARD Request/Reply* message with the CARD Server, and then with the resolved CAR(s). The PAR will refresh/update its local *CARTable* and will inform the MN by appending the identity and/or capabilities of the resolved CAR(s) to the *PrRtrAdv* message. The MN will select an appropriate NAR from amongst the discovered CARs, and store the outcome of the RAT/DCC process in a *LocalL2L3Cache* to avoid requesting the same information repeatedly, and to select an appropriate target AR from the list of CARs as quickly as possible if a handover is imminent.

## 4.3 xMobileIPv6 Simple Module

This section will provide those functions of the *xMobileIPv6* module which are related to the FMIPv6 and HMIPv6 protocol.

### *4.3.1 (Re-)Transmission Timers*

As specified in [3] *RtSolPr*, *HI* and *FBU* messages have to be sent periodically until a corresponding *PrRtrAdv*, *HAck* and *FBAck* message is received respectively. For this purpose, a so called *TransmitIfEntry* structure is defined, which contains the destination address of the message, numeric values for the current ACK timeout and the next scheduled time, a pointer to the interface over which the message is supposed to be sent and a pointer to the message itself.

Figure 8 shows the class diagram of the implemented retransmission timers. *TransmitIfEntry* is only the base class from which a *RtSolPrTransmitIfEntry* (for *RtSolPr*), *HIMsgTransmitIfEntry* (for *HI*) and *FBUTransmitIfEntry* (for *FBU*) are derived. The structure is attached as a *context pointer* to the scheduled message and therefore allows retrieving all necessary information in order to (re-)transmit the mobility message. As soon as the acknowledgment has been received, the list holding all transmission structures is traversed, the correct one deleted and the corresponding scheduled message canceled.

### *4.3.2 Packet Tunneling and Buffering*

Upon selecting an appropriate NAR, the MN will autoconfigure a prospective NCoA based upon NAR's prefix. It will then transmit a *FBU* message appended with PCoA and NCoA. A corresponding entry will be created inside the *BindingUpdateList* module of the MN and the *BindingCache* module [9] of the PAR binding the PCoA and NCoA. An *FBUTransmitIfEntry* retransmit timer will be initialized by the *createFBUTimer()* on the transmitting interface of the MN. The PAR will extract the NCoA from the *FBU* and performs the longest prefix match procedure (*doLongestPrefixMatch()*) on the NCoA to find the address of the NAR. An *HI* message will be created and sent towards NAR, and a corresponding *HIMsgTransmitIfEntry* timer will be initialized. The NAR will add the NCoA received in the *HI* message to its Neighbor Cache with a newly defined *PROXY* state. The NAR will then send a *HAck* back to PAR, which will cancel the corresponding timer, and PAR will create a tunnel towards NAR with entry point being the address of the PAR itself and the exit point being the NCoA. Only packets whose destination address matches the PCoA of the MN will be tunneled towards NAR, where they will get buffered in a newly defined *proxyBuffer* of the type *cQueue*.

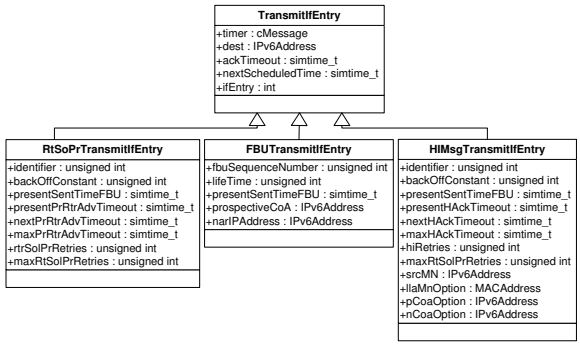



**Figure 8: Class Structure of Retransmission Timers**

### *4.3.3 Forwarding of Buffered Packets to MN*

The MN, after having established L2 connectivity with the NAR's AP, will send a *UNA* message of the type *IPv6NeighbourAdvertisement* and will perform the DAD test on the NCoA. The NAR will change the state of the NCoA in its Neighbor Cache from PROXY to STALE, and will forward all packets pending in the *proxyBuffer* towards the MN where they will be received by the *IPv6Tunneling* module and decapsulated as explained in [9].

## 4.4 A Subnet Configurator for IPv6

INET uses the *FlatNetworkConfigureator6* simple module to configure and assign IPv6 prefixes and addresses to the interfaces of the various nodes (hosts and routers) inside the system module. The limitation of this design is that all nodes are considered at the same topological level and belonging to one big subnet. Therefore, in order to realize hierarchical network topologies with distinct subnets, a *SubnetConfigurator6* simple module has been developed. The motivation is to implement HMIPv6 protocol in which the MAP router is located higher in the hierarchy than an AR. As shown in Figure 2, The MAP router defines an IPv6 subnet, with several ARs associated with it.

In contrast to the *FlatNetworkConfigurator6*, which always assigns a 64-bit unique prefix to all the interfaces, the *SubnetConfigurator6* enables the autoconfiguration and assignment of prefixes to the interfaces of the nodes (MAPs and ARs) located at different levels in the hierarchy. Each autoconfigured prefix is based on the prefix received from the router located one level up, whereas the length of the autoconfigured prefix will be extended according to the level of the hierarchy.

During simulation initialization, the *FlatNetworkConfigurator6* assigns IPv6 prefixes and addresses to all nodes except the MAP router and associated ARs. The *SubnetConfigurator6* is initialized when the MAP router receives the first *Router Advertisement (RA)* message from the network router located one level up (e.g., a core/gateway router) on its *egress* interface[2]. The RA will contain the network prefix, the size of which is determined by the position of the network router in the hierarchy. The *SubnetConfigurator6* will extend this prefix by autoconfiguring multiple unique prefixes and assign them to all the *ingress* interfaces[3] of the MAP router. These newly configured prefixes will then be periodically conveyed down to the ARs via the *RA* messages, which in turn will autoconfigure the addresses of the respective interfaces based on these prefixes.

# 5. RESULTS AND ANALYSIS

In this section we will evaluate the performance of the MMSEv6 in the context of HMIPv6 and FMIPv6 protocols. The test network environment is shown in Figure 2, in which the MN is moving across 4 ARs in a hierarchical network configuration defined by the *SubnetConfigurator6*. There are two MAP domains defined by MAP nodes namely MAP1 and MAP2. The MN undergoes *intra-domain* handovers as it enters and moves inside a particular MAP domain and *inter-domain* handover as it moves from one MAP domain into another. During handovers, the MN is communicating with CN[0] using a UDP traffic stream, and will thus establish relevant bindings with its HA and the respective CN during handovers. The UDP traffic characterizes a real-time CBR traffic. The wireless access network is based on the IEEE 802.11b WLAN standard with a free space channel model. The propagation delay between the MN, the HA and the CNs is negligible.

Since FMIPv6 and HMIPv6 are *functional extension* of the base MIPv6 protocol, therefore the performance accuracy and reliability of MMSEv6's MIPv6 protocol will lend itself to the both the protocols. The performance accuracy and reliability of the MMSEv6's MIPv6 protocol implementation has been validated against a real MIPv6 test bed and the results have already been presented in [9].

## 5.1 HMIPv6 Performance Results

As explained in section 2.2, HMIPv6 is designed to reduce the signaling overhead of the MIPv6 protocol by localizing the exchange of binding messages (inclusive of the test messages (*HoT/CoT* and *HoTI/CoTI*) for *Return Routability* test) within a single MAP domain. The bindings within a MAP domain are handled by the MAP router, and the MN does not need to send binding messages to HA and CN as long as it moves inside a MAP domain. This would reduce the number of binding messages being sent by the MN to the HA and the CN. This is depicted in Figure 9, which compares the performance of HMIPv6 with MIPv6 in terms of the number of binding messages exchanged between the MN and the respective HA and the CNs. The signaling load is measured by comparing the number of individual messages and the total number of binding messages transmitted when the MN undergoes inter/intra domain handovers. As evident from Figure 9, when the MN enters the domain of MAP1 (or MAP2), it induces a higher total signaling load due to the exchange of two additional binding messages (i.e., *LBU* and *LBA*) with the MAP1 (or MAP2); besides the regular exchange of binding messages with the HA and CN. Once inside the MAP1 (or MAP2) domain, the MN only exchanges a single pair of LBU/LBA message pair with the MAP1 (or MAP2) router during every handover instance occurring within the respective MAP domain without needing to perform binding with the HA and the CNs respectively. In other words, HMIPv6 incurs a *constant minimum signaling overhead* as opposed to MIPv6, in which the number of binding messages exchanged is proportional to the number of the CNs communicating with the MN. This is depicted in Figure 10, which shows the effect of increasing number of CNs on the signaling load.

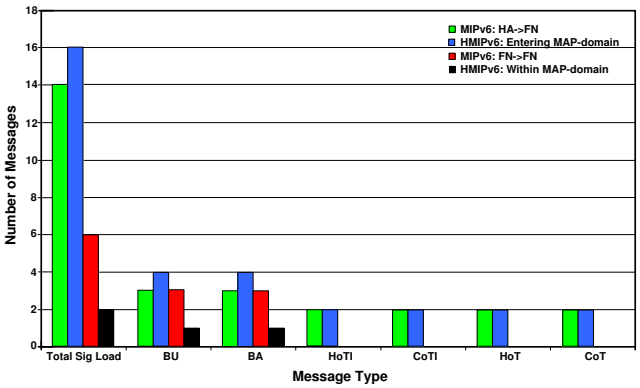



**Figure 9: Signaling Load Comparison between MIPv6 and HMIPv6**

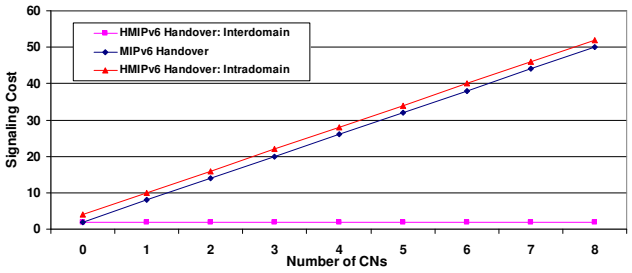



**Figure 10: Total Signaling Cost Comparison between MIPv6 and HMIPv6 as a Function of the Number of CNs**

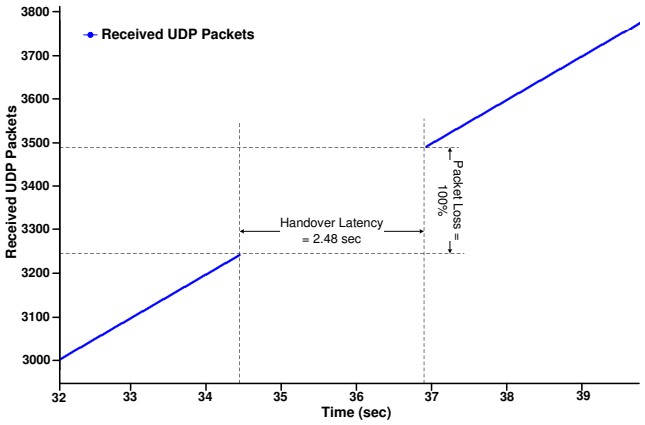
## 5.2 FMIPv6 Performance Results

As explained in section 2.1, FMIPv6 has been designed to reduce not only the handover interval but also make the handover seamless by reducing the packet loss that occurs during the normal MIPv6 handover. Figure 11(a) and 11(b) depicts the handover performance of MIPv6 and FMIPv6 respectively and thus provides a comparative view. As evident from Figure 11(a), MIPv6 not only incurs a higher handover latency but it also incurs a 100% packet loss during the handover time as the MN is practically disconnected from the network. *Note that in the MIPv6 handover scenario,the MN is moving from HA to the foreign network (FN) incurring an additional DAD delay of 1 second at the HA*. On the other hand, FMIPv6 not only incurs a smaller handover latency but there is zero packet loss due to buffering of packets at the NAR during the handover. These buffered packets eventually get forwarded to the MN after it has re-established IP connectivity with the NAR. This forwarding of buffered packets is seen as a sudden linear increase in the received UDP packets in Figure 11(b). It may be mentioned here that the seamless performance of FMIPv6 is also dependent

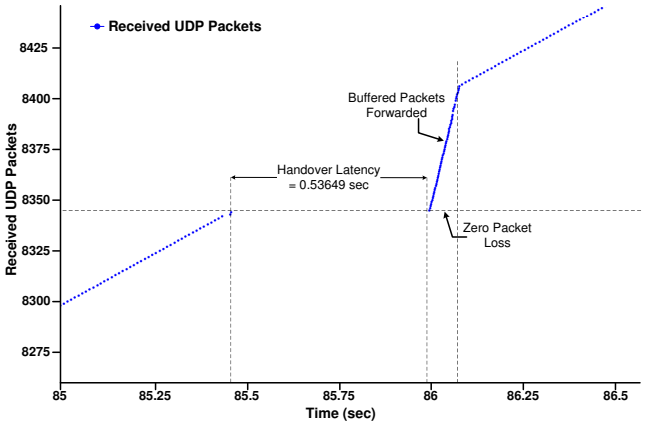
[2]The egress interface will receive and process RA messages but will not transmit any.

[3]The ingress interface will transmit RA messages but will not receive or process any.

on the size of the buffer and the data rate. For the sake of demonstration, we have kept the buffer size equal to 1000 packets and the inter-arrival time between successive packets is $10^{-2}$ seconds, hence the 0% packet loss.




(a) MIPv6




(b) FMIPv6

**Figure 11: Comparison of MIPv6 and FMIPv6 Handover Effect on a UDP Data Traffic Stream**

# 6. CONCLUSIONS

MMSEv6 is a comprehensive and sophisticated simulation engine that has been designed and developed for carrying out realistic and accurate simulation modeling of MIPv6 based protocols such as FMIPv6 and HMIPv6 on a single integrated platform. The development of MMSEv6 has been made possible because of the *extensibility* feature of the available xMIPv6 simulation model [9, 8], which accurately simulates MIPv6 protocol. In other words, MMSEv6 is an extension of the previous xMIPv6 simulation model and it has been designed in such a way so as to enable practitioners and researchers to rapidly develop and implement not only their own proposed prototype solutions and algorithms, but also enable them to test it under varied network and load conditions. As a demonstration of the rapid-prototyping properties of the MMSEv6, we have already implemented and tested, in a short amount of time, an optimization solution called *Multi-Hop Discovery of Candidate Access Routers (MHD-CAR)* protocol [10] and the performance results presented in [11].

The accuracy of the simulation framework has been ensured by modeling messages based on the actual message formats of the respective protocols. The event timers are implemented by strictly conforming to the relevant IETF standardized specifications. Since both FMIPv6 and HMIPv6 are based on MIPv6 protocol, and as the performance of MIPv6 and its various aspects has been validated against a real MIPv6 test bed, therefore this accuracy will automatically lend itself to the performance of these protocols as well.

Taking advantage of the rapid prototyping feature inherent in the MMSEv6 platform design, we are in the process of further extending the MMSEv6 engine with the Network Mobility (NEMO) Basic Support Protocol [1], to manage the mobility of Mobile Networks.

# 7. ACKNOWLEDGMENTS

The authors would like to thank Mr. Alain Tigyo for his contribution with the design and implementation of the *subnetConfigurator6* module and for his efforts for testing and validating various aspects of the MMSEv6 performance.

# 8. REFERENCES

[1] V. Devarapalli, et al, "Network Mobility (NEMO) Basic Support Protocol", Request for Comments (Draft Standard) 3963, IETF, January 2005.

[2] D. Johnson, C. E. Perkins, and J. Arkko, “Mobility Support in IPv6”, Request for Comments (Proposed Standard) 3775, IETF, June 2004.

[3] R. Koodli, "Fast Handovers for Mobile IPv6", Request for Comments (Draft Standard) 5268, IETF, June 2008.

[4] M. Liebsch, A. Singh et.al., “Candidate Access Router Discovery Protocol (CARD)”, Request for Comments (Proposed Standard) 4066, IETF, July 2005.

[5] OMNeT++ Community Site, http://www.omnetpp.org, January 2008.

[6] H. Soliman, C. Castellucia, K. El Malki, L. Bellier, " Hierarchical Mobile IPv6 (HMIPv6) Mobility Management", Request for Comments (Draft Standard) 5380, IETF, October 2008.

[7] S. Thomson, T. Norten, T. Jinmei “IPv6 Stateless Address Autoconfiguration“, Request for Comments (Proposed Standard) 4862, IETF, Sep. 2007.

[8] xMIPv6 Project Homepage: http://www.kn.e-technik.uni-dortmund.de/content/view/232/lang,de/

[9] F. Zarrar Yousaf, C. Bauer, C. Wietfeld, “An Accurate and Extensible Mobile IPv6 (xMIPv6) Simulation Model for OMNeT++”, 1st ACM/ICST International OMNeT++ Workshop on the SIMUTools Conference, Marseille, March 2008.

[10] F. Zarrar Yousaf, C. Wietfeld, “Multi-Hop Discovery of Candidate Access Routers (MHD-CAR)”, draft-yousaf-ietf-network-mhdcar-00.txt, Internet Draft, IETF, April 2008. Work in progress.

[11] F. Zarrar Yousaf, C. Wietfeld, “Multi-Hop Discovery of Candidate Access Routers (MHD-CAR) for Fast Moving Mobile Nodes”, 19th IEEE International Symposium on Personal, Indoor and Mobile Radio Communications (PIMRC), Cannes, 2008.