
# Indian Institute of Management Calcutta

**Working Paper Series**

**WPS No. 769**
**October 2015**

## An Experimentation with Simulating Mobile IPv6 (MIPv6) in NS-3 to handle User Mobility

**Manoj Kumar Rana**
School of Mobile Computing and Communication, Jadavpur University, Kolkata, India.

**Swarup Mandal**
Information Technology and Services, Wipro Limited, Kolkata, India

**Bhaskar Sardar**
Dept. of Information Technology, Jadavpur University, Kolkata, India

&

**Debashis Saha**
Professor, IIM Calcutta, Diamond Harbour Road, Joka P.O., Kolkata 700 104 India
http://facultylive.iimcal.ac.in/workingpapers

# An Experimentation with Simulating Mobile IPv6 (MIPv6) in NS-3 to Handle User Mobility

*Manoj Kumar Rana[1], Swarup Mandal[2], Bhaskar Sardar[3] and Debashis Saha[4]*

[1]School of Mobile Computing and Communication, [3]Dept. of Information Technology
Jadavpur University,Kolkata, India
[1]manoj24.rana@gmail.com, [3]bhaskargit@yahoo.co.in
[2]Information Technology and Services, Wipro Limited, Kolkata, India
swarup.mandal@wipro.com
[4]MIS Group, Indian Institute of Management (IIM)Calcutta,Kolkata, India
ds@iimcal.ac.in

## ABSTRACT

Mobility management is becoming an important issue day by day due to the immense proliferation of wireless access networks globally. The IETF has standardized Mobile IPv6 (MIPv6) to handle user mobility in IP-based mobile data networks. Worldwide the researchers are putting in a lot of effort to render the mobility management as seamless as possible in order to enhance user experience. These researchers need a simulation environment to test their novel ideas before taking it to the proof-of-concept level. Although ns-2 simulator supports MIPv6, unfortunately its successor, the ns-3 simulator, does not support MIPv6 conforming to the IETF specification. This has created a huge void in the simulation based testing of the innovative ideas put forth by network researchers. To overcome this shortfall, we present an implementation approach of MIPv6 for ns-3 and subsequently carry out its performance evaluation. We provide detailed description of the implemented MIPv6 module and also preliminary results by executing the MIPv6 module in ns-3.

## General Terms

Mobility management, IPv6, Mobile data, User experience, Performance, Design.

## Keywords

Network simulator, ns-3, Mobile IPv6, IP Mobility Management, Handoff Management, Simulation.

## 1. INTRODUCTION

Recently, we have witnessed an explosive growth in IP enabled mobile devices such as Smartphones, tablets etc. Consequently, the Internet Engineering Task Force (IETF) has standardized several IP mobility management protocols to gracefully handle the mobility of such devices in the next generation all-IP mobile networks. Starting with the basic Mobile IPv6 (MIPv6), more advanced protocols such as fast handover for MIPv6 (F-MIPv6) and hierarchical MIPv6 (HMIPv6) have also been released by the IETF MIPSHOP working group. Although these new protocols differ from MIPv6 to some extent with respect to their functionality, they maintain the same basic MIPv6 protocol stack intact. Therefore, simulation study of next generation all-IP mobile networks (e.g., 4G/5G) demands the basic MIPv6 implementation in the first place. Hence, the current network simulators must have MIPv6 module in built to draw the attention of the mobility research community.

In the last few decades, network simulator ns-2 was heavily used by the mobility research community to conduct various experiments on simulated mobile networks. The recent research works also heavily depend on the advanced version of ns-2, namely ns-3. It has better alignment with real systems and also supports advanced features. However, even today, ns-3 does not contain built-in MIPv6 package in its main tree. Although many Linux native stack based MIPv6 implementations are available, they require users to modify the Linux native stack to make changes in MIPv6 implementation. For a naïve ns-3 user, it is quite cumbersome to change Linux stack, in effect preventing the researchers to use ns-3 widely.

So, we have implemented MIPv6 using the basic ns-3 stack only. In this paper, we present the progress of our work on the implementation of MIPv6 and its validation. Our implementation would help an ordinary user of ns-3 to easily and/or modify MIPv6 code for implementing advanced versions of mobility management schemes.

Rest of the paper is organized as follows. We have briefly discussed recent research efforts on the simulation and emulation of MIPv6 in Section 2. In Section 3, we have described the structure of basic ns-3 which is required to implement new modules in ns-3. The main components of our implementation are discussed in Section 4. Preliminary simulation results are given in Section 5. Finally, Section 6 concludes this paper and sketches some potential future works.

## 2. Related Works

The available MIPv6 implementations can be broadly classified as Linux test bed implementation and simulation implementation. The Lancaster University developed the first Linux based MIPv6 implementation for kernel 2.1.90 [1]. Then, Helsinki University of Technology came up with an implementation of MIPv6 for kernel 2.4.22 as a part of MIPL project [2]. However, they followed an initial version of MIPv6 draft, which is now outdated. Then, many Linux based MIPv6 implementations have been released in the recent past such as CNI-MIPv6 [3] test bed and **UMIP** [4].

The simulation based implementations are mainly based on OMNET++, OPNET and ns-2/3. The extensible MIPv6 (xMIPv6) [5] implementation is the most eminent implementation in OMNET++. Some other OMNET++ based MIPv6 simulations can be found in [6] and [7]. On the other hand, OPNET based MIPv6 simulation model have been developed in the institute of informatics of Goettingen University by Le et. al. in 2005. Presently the most common and well known network simulator is ns-2/3. The implementation of MIPv6 in ns-2 is well known and widely used by the research community. A possible implementation of IPv6 and MIPv6 in ns-3 can be found in [8]. In [9], H. Y. Choi et. al. has described

an implementation of proxy mobile IPv6 (PMIPv6) in ns-3. To some extent, we follow the design approach presented in [9]. But MIPv6 and PMIPv6 are functionally different and so our design approach is quiet different compared to them.

## 3. Overview of ns-3

The discrete event network simulator ns-3 is composed of two basic parts: ns-3 core and the network protocol modules. These ns-3 components can be found in the source code directory ns-3-dev/src/. We have briefly discussed a few of them to enhance the understandability of our MIPv6 implementation.

The network protocols module includes implementation of IPv4, IPv6, TCP, UDP, ICMP, ARP, etc. The *InternetStackHelper* helps an user to install these protocols in a node. The most important functions of the Internet module are: the *Receive()* and *LocalDeliver()* methods of the *Ipv6L3Protocol* class, *Assign()* method of the *Ipv6AddressHelper* class, *DoDAD()* method of *Icmpv6L4Protocol* class and *Receive()* method of *Ipv6L4Protocol* class. When L2 connection is established, the *Assign()* function assigns IPv6 address to the node. The address assignment follow mandatory duplicate address detection (DAD) mechanism. The DAD process is handled by the *DoDAD()* function. In the network layer, a received packet is processed by two *Receive()* functions. The first *Receive()* function, a part of *Ipv6L3Protocol* class, checks whether it is a data packet (it can handle control or signaling packet in its own) and calls the second *Receive()* function, a part of *Ipv6L4Protocol* class, which actually checks the header and forwards it to the correct application running in the upper layer.

Although ns-3 core module contains many models, we briefly describe two models, namely, Packet and NetDevice models. A Packet is simply an instance of a buffer class. Its members are packet tag, byte tag, and metadata. The byte tags are used to tag a portion of the bytes inside the packet while the packet tags are used to tag the whole packet, and the description of the headers and trailers of the packet buffer is given by the metadata. The *Header* base class can be used to implement various types of headers which are used to implement pure virtual functions such as *GetSerializedSize(), Serialize(), DeSerialize()* and *Print()*. The NetDevice is an API used by the IP and ARP protocols to send packet through the MAC layer. One can inherit this base class to add new functionality in it. The most important function of this class is *Send()* function that sends a packet from layer 3 to a particular Network Device based on the decision of IPv6 routing protocol.

## 4. Implementation of MIPv6 in NS-3

We have used existing classes of ns-3 as well as defined new classes (non-derived) for implementing MIPv6 in ns-3. Figure 1 show these classes. The association between a group of classes (e.g., Ipv6L3Protocol class and Ipv6MobL4Protocol and Ipv6TunL4Protocol) indicates an event driven triggering mechanism. We inherit the Header base class to define MIPv6 specific classes. We extend these classes further to support BU processing and data packet processing.

The *MIPv6MN* class implements BU process by using a callback mechanism. When the newly configured IPv6 address reaches “PREFERRED” state, the BU process is triggered. The m_NewIpCallback function of *MIPv6MN* class is used to receive a trigger from *Icmpv6L4Protocol* when the DAD process completes. The BU message is forwarded to the *NetDevice* object of the HA/CN. Then the *MIPv6MobL4Protocol* of the HA/CN sets up the tunnel towards MN. Similarly *MIPv6MN*

sets up the tunnel after receiving BA. We have defined new functions to send and receive mobility related signaling messages such as *SendMessage()* and *Receive()*. The *Receive()* function uses new handler functions *BAHandle()*, *HoTHandle()* and *CoTHandle()* to process received signaling messages. The *BList* object *MIPv6MN* class is updated with the information such as home agent address, lifetime, BU state, CoA, HoA etc. In the similar way the *MIPv6CN/HA* classes handle the mobility messages and update the *BCache* object. All mobility messages are demultiplexed by the *MIPv6Demux* class and forwarded

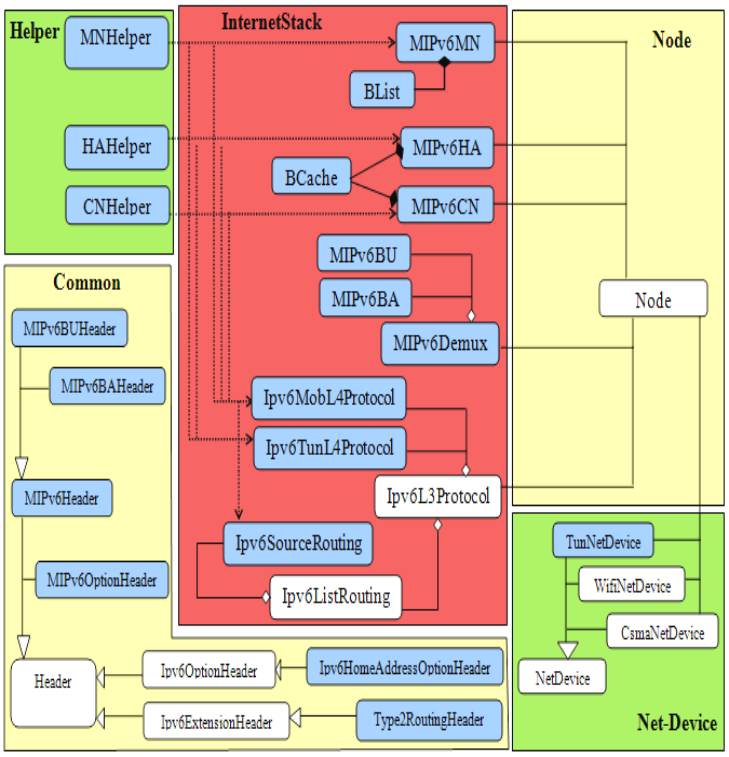


**Figure 1. MIPv6 Classes**

to the proper handler class of the *MIPv6MN/CN/HA* classes.

The *Ipv6MobL4Protocol* and *Ipv6TunL4Protocol* classes use same *Receive()* function to handle mobility messages and data packets respectively. The *Receive()* function of *Ipv6MobL4Protocol* class forwards the mobility message to the corresponding *Receive()* function of the *MIPv6MN/CN/HA* class while the *Receive()* function of *Ipv6TunL4Protocol* class receives data packets from the *Ipv6L3Protocol* class.

We have defined a new class, *TunNetDevice*, by inheriting *NetDevice* class to enable IP-in-IP encapsulation de-capsulation of packets and transmission over physical interface of a node. The helper classes *MNHelper*, *HAHelper* and *CNHelper* classes define overloaded *install()* method to allow the users to install MIPv6 in a node.

To process data packet for MN to HA direction *Ipv6SourceRouting* class is used to determine the outgoing tunnel net device and encapsulate the packet. This encapsulated packet is then passed to *Ipv6L3Protocol*. The HA uses *Ipv6TunL4Protocol* class to de-capsulate the packet and sends it to the CN. In the similar way, the packets from the HA are routed to the MN's CoA. To implement route optimization, we use m_NewRouteCallback method of Ipv6L3Protocol class which changes the destination address to MN's CoA.

## 5. Simulation Results

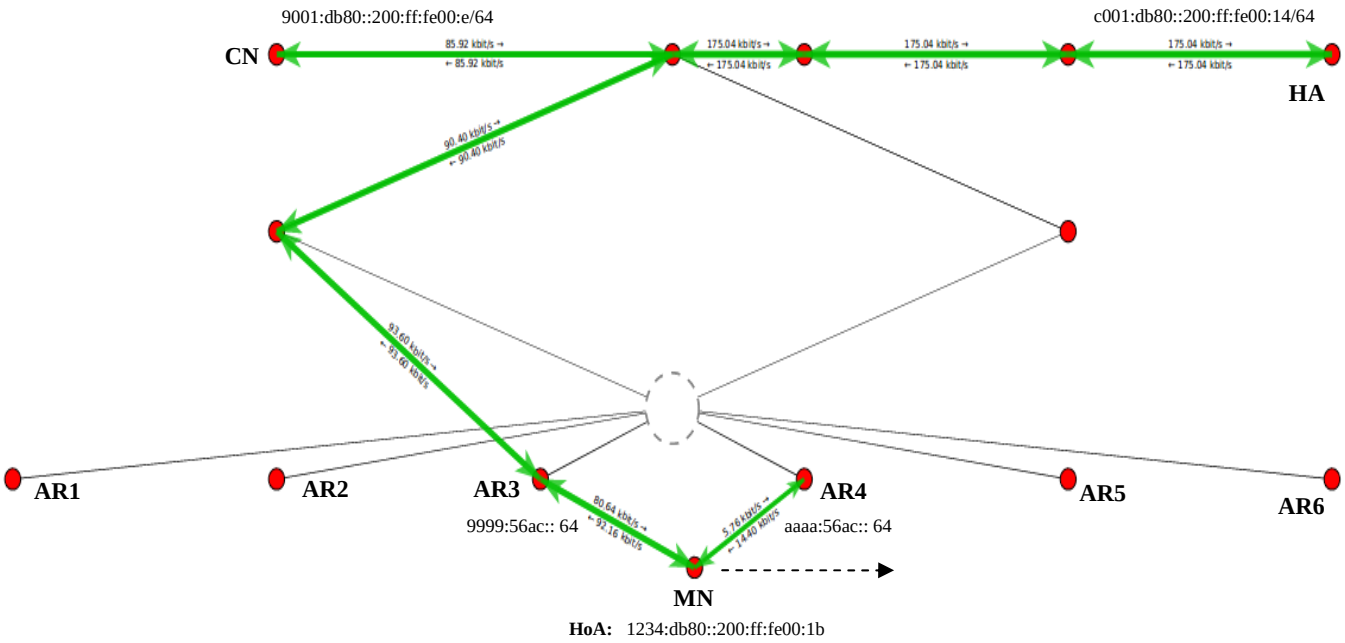


**Figure 2. Simulation Topology**

This section reports preliminary results to verify the correctness of our implementation. Figure 2 shows the simulation environment. For the wired links, we have used data rate and link delay of 100 Mbps, 20 milliseconds respectively. For the wireless links, we have used date rate and link delay of 11 Mbps, 10 milliseconds respectively. The CN and the MN runs an echo client server application based on UDP. The PCAP trace file is used to measure the handover performance of MIPv6. Figure 3 and 4, shows the handover packet trace at the MN. The MN sends the last packet to the CN through AR1 at *t*=30.181347 second. The MN receives RA from AR2 at *t*=30.221232 second and starts address configuration process. At *t*=31.350586 second, the MN sends a BU to its HA and receives corresponding BA at *t*=31.592070 second. Then the MN starts receiving packets through AR2 *t*=31.794587 second. Then the MN sends HoTI and CoTI packets at *t*=31.592070 second and *t*=31.712474 second respectively. The MN receives HoT and CoT packets at *t*=31.712474 second and *t*=31.836580 second respectively as shown in Figure 3 and Figure 4. The MN sends BU to the CN at *t*=31.836580 second, and receives corresponding BA at *t*=31.957026 second, as shown in Figure 4. We define the handover delay with the HA as the time difference between the receipt of the last packet from AR1 and receipt of first packet from AR2. Hence, the handoff delay is 1.61324 second. Also we define the handoff delay with the CN as the time difference between the last packet from AR1 and the reception of the BA from CN. From the PCAP traces, we observe that the handover delay (with HA and CN) is 1.75679 second.

```
29.927232 IP6 fe80::200:ff:fe00:17 > ff02::1: ICMP6, router advertisement, length 80
29.981347 IP6 1001:db80::200:ff:fe00:5 > 9999:56ac::200:ff:fe00:1b: IP6 c001:db80::200:ff:fe00:14 > 1001:db80::200:ff:fe00:1b: IP6 9001:db80::200:ff:fe
00:e.49153 > 1234:db80::200:ff:fe00:1b.9: UDP, length 1024
29.981347 IP6 9999:56ac::200:ff:fe00:1b > 1001:db80::200:ff:fe00:5: IP6 :: > c001:db80::200:ff:fe00:14: IP6 ::.9 > 9001:db80::200:ff:fe00:e.49153: UDP,
 length 1024
29.988232 IP6 fe80::200:ff:fe00:17 > ff02::1: ICMP6, router advertisement, length 80
30.025232 IP6 fe80::200:ff:fe00:17 > ff02::1: ICMP6, router advertisement, length 80
30.079232 IP6 fe80::200:ff:fe00:17 > ff02::1: ICMP6, router advertisement, length 80
30.130232 IP6 fe80::200:ff:fe00:17 > ff02::1: ICMP6, router advertisement, length 80
30.163232 IP6 fe80::200:ff:fe00:17 > ff02::1: ICMP6, router advertisement, length 80
30.181347 IP6 1001:db80::200:ff:fe00:5 > 9999:56ac::200:ff:fe00:1b: IP6 c001:db80::200:ff:fe00:14 > 1001:db80::200:ff:fe00:1b: IP6 9001:db80::200:ff:fe
00:e.49153 > 1234:db80::200:ff:fe00:1b.9: UDP, length 1024
30.181347 IP6 9999:56ac::200:ff:fe00:1b > 1001:db80::200:ff:fe00:5: IP6 :: > c001:db80::200:ff:fe00:14: IP6 ::.9 > 9001:db80::200:ff:fe00:e.49153: UDP,
 length 1024
30.221232 IP6 fe80::200:ff:fe00:18 > ff02::1: ICMP6, router advertisement, length 80
30.287232 IP6 fe80::200:ff:fe00:18 > ff02::1: ICMP6, router advertisement, length 80
30.350232 IP6 fe80::200:ff:fe00:18 > ff02::1: ICMP6, router advertisement, length 80
30.408232 IP6 fe80::200:ff:fe00:18 > ff02::1: ICMP6, router advertisement, length 80
30.452232 IP6 fe80::200:ff:fe00:18 > ff02::1: ICMP6, router advertisement, length 80
```

**Figure 3. IPv6 Packet Trace at MN from the Reception of the Last UDP Packet to Sending of the CoTI Message**

```
31.350586 IP6 aaaa:56ac::200:ff:fe00:1b > c001:db80::200:ff:fe00:14: mobility: BU seq#=6 AHL lifetime=262128
31.371232 IP6 fe80::200:ff:fe00:18 > ff02::1: ICMP6, router advertisement, length 80
31.422232 IP6 fe80::200:ff:fe00:18 > ff02::1: ICMP6, router advertisement, length 80
31.468232 IP6 fe80::200:ff:fe00:18 > ff02::1: ICMP6, router advertisement, length 80
31.522232 IP6 fe80::200:ff:fe00:18 > ff02::1: ICMP6, router advertisement, length 80
31.583232 IP6 fe80::200:ff:fe00:18 > ff02::1: ICMP6, router advertisement, length 80
31.592070 IP6 c001:db80::200:ff:fe00:14 > aaaa:56ac::200:ff:fe00:1b: mobility: BA status=0 seq#=6 lifetime=262128
31.592070 IP6 aaaa:56ac::200:ff:fe00:1b > 9001:db80::200:ff:fe00:e: mobility: HoTI
31.618232 IP6 fe80::200:ff:fe00:18 > ff02::1: ICMP6, router advertisement, length 80
31.650232 IP6 fe80::200:ff:fe00:18 > ff02::1: ICMP6, router advertisement, length 80
31.698232 IP6 fe80::200:ff:fe00:18 > ff02::1: ICMP6, router advertisement, length 80
31.712474 IP6 9001:db80::200:ff:fe00:e > aaaa:56ac::200:ff:fe00:1b: mobility: HoT nonce id=0x0
31.712474 IP6 aaaa:56ac::200:ff:fe00:1b > 9001:db80::200:ff:fe00:e: mobility: CoTI
31.731232 IP6 fe80::200:ff:fe00:18 > ff02::1: ICMP6, router advertisement, length 80
31.777232 IP6 fe80::200:ff:fe00:18 > ff02::1: ICMP6, router advertisement, length 80
31.794587 IP6 2001:db80::200:ff:fe00:9 > aaaa:56ac::200:ff:fe00:1b: IP6 c001:db80::200:ff:fe00:14 > 2001:db80::200:ff:fe00:1b: IP6 9001:db80::200:ff:fe
00:e.49153 > 1234:db80::200:ff:fe00:1b.9: UDP, length 1024
31.794587 IP6 aaaa:56ac::200:ff:fe00:1b > 2001:db80::200:ff:fe00:9: IP6 :: > c001:db80::200:ff:fe00:14: IP6 ::.9 > 9001:db80::200:ff:fe00:e.49153: UDP,
 length 1024
31.824232 IP6 fe80::200:ff:fe00:18 > ff02::1: ICMP6, router advertisement, length 80
31.836580 IP6 9001:db80::200:ff:fe00:e > aaaa:56ac::200:ff:fe00:1b: mobility: CoT nonce id=0x0
31.836580 IP6 aaaa:56ac::200:ff:fe00:1b > 9001:db80::200:ff:fe00:e: mobility: BU seq#=6 AHL lifetime=262128
31.879232 IP6 fe80::200:ff:fe00:18 > ff02::1: ICMP6, router advertisement, length 80
31.933232 IP6 fe80::200:ff:fe00:18 > ff02::1: ICMP6, router advertisement, length 80
31.957026 IP6 9001:db80::200:ff:fe00:e > aaaa:56ac::200:ff:fe00:1b: mobility: BA status=0 seq#=6 lifetime=262128
31.982403 IP6 2001:db80::200:ff:fe00:9 > aaaa:56ac::200:ff:fe00:1b: IP6 c001:db80::200:ff:fe00:14 > 2001:db80::200:ff:fe00:1b: IP6 9001:db80::200:ff:fe
00:e.49153 > 1234:db80::200:ff:fe00:1b.9: UDP, length 1024
31.982403 IP6 aaaa:56ac::200:ff:fe00:1b > 2001:db80::200:ff:fe00:9: IP6 :: > c001:db80::200:ff:fe00:14: IP6 ::.9 > 9001:db80::200:ff:fe00:e.49153: UDP,
 length 1024
```

**Figure 4. IPv6 Packet Trace at MN from the Reception of the CoT Message to Sending of the BU Message to CN**

## 6. Conclusion and Future Work

In this paper, we have provided an implementation of MIPv6 in ns-3 network simulator. We have implemented MIPv6 with utmost care and close conformance and strict adherence to IETF standards. Our preliminary result proves the correctness of our implementation in terms of handover delay.

Our implementation of MIPv6 module may be extended easily for implementing new protocols like FMIPv6, HMIPv6 in ns-3 in near future. However, implementation of dynamic HA address discovery, dual stack MIPv6 support, and HA multicasting membership control are planned for future.

## REFERENCES


[1] http://www.cs-ipv6.lancs.ac.uk/MobileIP/

[2] Tuominen, A., J., and Petander, H. 2001. MIPL Mobile IPv6 for Linux in HUT Campus Network MediaPoli. *Proceedings of Ottawa Linux Symposium* 01 (July, 2001). DOI= https://www.kernel.org/doc/ols/2001/mipl.pdf

[3] Yousaf, F., Z., Bauer, C., and Wietfeld, C. 2008. An accurate and extensible mobile IPv6 (xMIPV6) simulation model for OMNeT++. http://www.kn.e-technik.tu-dortmund.de/en/forschung/ausstattung/xmipv6.html

[4] Aramoto, M., Nakamura, M., Sugimoto, S. and Takamiya, N. 2008. UMIP: USAGI-patched Mobile IPv6 for Linux. http://umip.linux-ipv6.org/

[5] Yousaf, F., Z., Bauer, C. and Wietfeld, C. 2008. An accurate and extensible mobile IPv6 (xMIPV6) simulation model for OMNeT++, *Proc.Simutools '08*, Article No. 88.

[6] Carmona-Murillo, J., and Gonzalez-Sanchez, J.-L. 2008. Handover Performance Analysis in Mobile IPv6: A Contribution to Fast Detection Movement, *Proc. Int'l Conf Wireless Information and Systems*.

[7] Zhang, Y., and Bi, H. 2012. The Simulation of Hierarchical Mobile IPv6 with Fast Handover using NS2. *Procedia Engineering, 2nd SREE Conference on Engineering Modelling and Simulation* (CEMS 2012). 37, 214–217.

[8] Mauchle F., Frei S. and Rinkel A. 2010. Simulating Mobile IPv6 with ns-3, *SIMUTools 2010,* March 15–19, Torremolinos, Malaga, Spain.

[9] Choi, H., -Y., Min, S., -G,. Y.-H. Han, Park, J. and Kim, H. 2010. Implementation and Evaluation of Proxy Mobile IPv6 in NS-3 Network Simulator. *Proc. Ubiquitous Information Technologies and Applications* (CUTE), 2010.