IPv6 macro mobility simulation using OMNeT++ environment

Sándor Imre, Róbert Schulcz, Csaba Csegedi, Szabolcs Vajda
Budapest University of Technology and Economics,
Dept. of Telecommunication, Mobile Communications Laboratory
H-1117 Pázmány P. S. 1/D, Budapest, Hungary
Tel: +36 1 463 32 27, email: {imre, schulcz}@hit.bme.hu

ABSTRACT

In this paper we will present a new simulation environment for examining IPv6 macro mobility. The subject of our investigation now is only macro mobility, because nowdays so called "micro mobility" belongs to lower layers. The simulator is written in OMNeT++ Discrete Event Simulation System.

1. Introduction

Nowadays there are two keywords in telecommunications: mobility and Internet. As the capacity and speed of small handheld devices and laptop-sized computers has increased dramatically in the past few years, the demand for mobile Internet access, telephony, videoconference, messaging, etc. while being away from home or moving also became significant. The current technology trends focus on implementing all these applications based on IP (All-IP technology) [1].

The current version of IP – IPv4 – was created for wired networks and the mobility support was added only later. For this reason it can not provide efficient support for mobile devices. The next generation of IP – IPv6 – has built-in mobility support from the beginning with important new features like bigger address space, reduced administrative overhead, support for address renumbering, improved header processing and reasonable security.

This paper is organized as follows: in Section 2 we present mobility aspect of IP networks, IP micro- and macro mobility. The subject of our investigation is only macro mobility, because in nowadays solutions micro mobility belongs to lower layers. In Section 3 we present the OMNeT++ simulation tool compared to other systems available at the market today. We introduce our IP macro mobility simulation environment, the modules and messages. We have completed the simulation, tested in different scenarios the results of binding lifetime management simulation and analysis will be presented in Section 4. The results from the simulator will be verified by theoretical results. Finally we conclude our work and present our future plans.

2. IP mobility

Mobile hosts connected to the Internet via a wireless interface are likely to change their point of access frequently. A mechanism is required that ensures that packets addressed to moving hosts are successfully delivered. During handover, packet loss may occur due to delayed propagation of new location information up to the home agent. These losses should be minimized in order to avoid he degradation of service quality as handover become more frequent. Mobility management can be divided into two parts: micro- and macro mobility. Macro mobility handles interdomain handovers, while micro mobility responsible for intradomain handovers (see Figure 1.) [2].

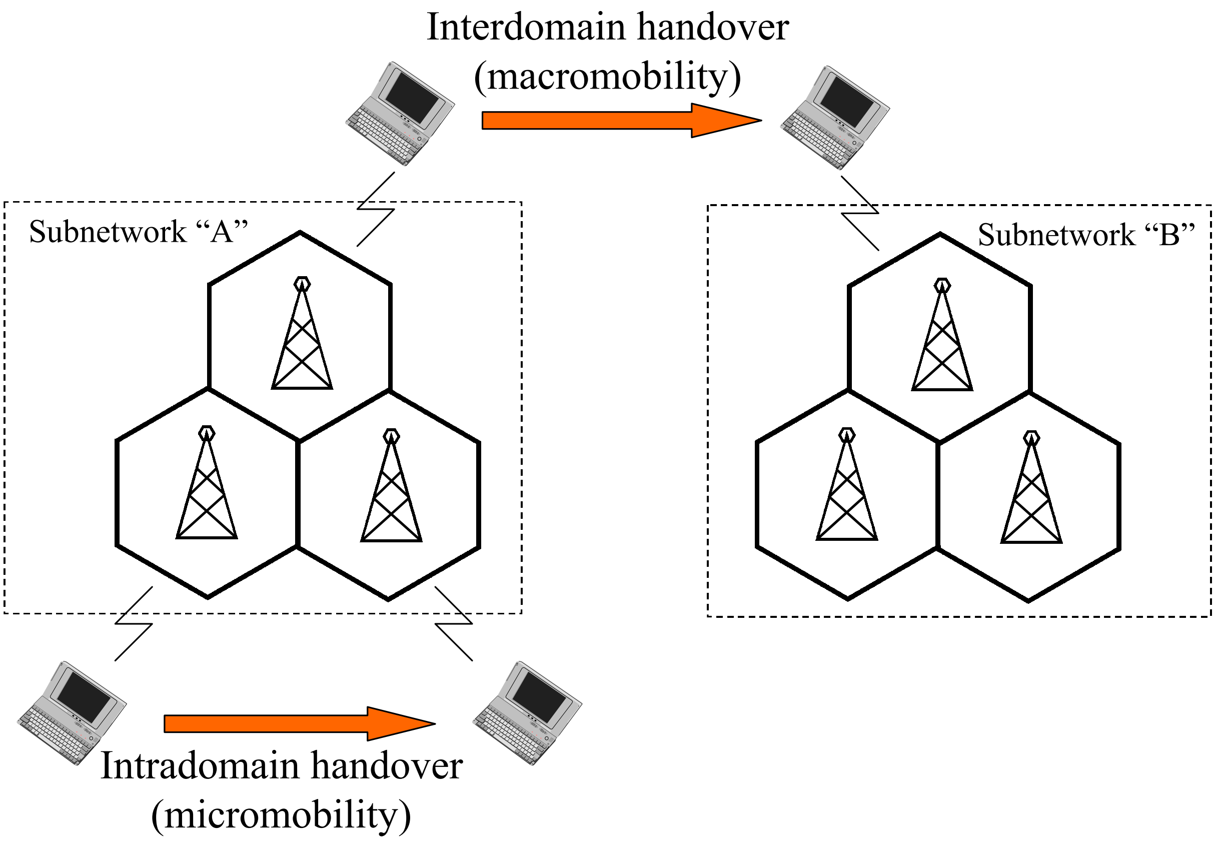

Figure 1. Micro- and macro mobility

To improve performance, the frequent handoffs (due to small radio cells) inside a given domain – also so called intra domain handovers – are handled locally by the micro mobility protocols. The role of micro mobility protocols is to hide user movement from the mobile IPv4 or IPv6 protocol, by handling user mobility locally, fast, and simple inside the micro mobility domain. IPv6 or Mobile IPv4 are responsible for wide area mobility support, called macro mobility [4].

We have investigated both mobility levels, and created a simulator framework to study the performance and interoperability of the micro- and macro mobility protocols.

2.1. IPv6 macro mobility

Every mobile device in IPv6 can always be addressed with its home address. When the mobile device is not attached to its home network, it obtains a temporary IP address – a care-of address – from the foreign network it is currently attached to. In order to be able to receive packages in this case the mobile always informs its home agent – a router in its home subnetwork – about its current care-of address. Correspondent nodes can send packages directly to the care-of address if they know it, otherwise they send them to the home address and the home agent forwards them to the mobile. The association between the home address and the care-of address is called binding. In IPv6 networks, every node contains a so-called Binding Cache to store binding information about mobile devices.

With the limited capability of mobiles and network overhead caused by triangle routing the optimization of the binding cache's size and the binding entries' lifetimes is very important. Our simulation demonstrates this issue in different network scenarios. We investigate different statistics like end-to-end delay time, rate of packets sent via triangle routing, rate of packet loss, handover frequency, etc.

2.2. IP micro mobility

The Mobile IP protocol is considered to have limitations in its capability to handle large numbers of mobile stations moving fast between different radio cells. The handover frequency should typically not exceed once a second. However, Mobile IP is well suited for interconnecting disparate cellular networks effectively providing global mobility. Resulting from this fact, several micro mobility approaches have been proposed within the IETF, which are supporting mobility in a well-defined area. Such as the two most discussed micro mobility protocols: HAWAII and CIP.

3. The simulation environment

We have developed a simulation environment to prove our concepts of Mobile IPv6 under OMNeT++. OMNeT++ (Objective Modular Network Testbed in C++) is a free, open-source discrete event simulation tool, similar to other tools like PARSEC, NS, or commercial products like OPNET [8]. Our Mobile IPv6 model can be freely downloaded along with many other models.

3.1. Modules of the macro mobility environment

Our simulation environment deals with the IPv6 Mobility Extension, especially with the binding management methods. With the simulator, we can easily build different network scenarios by providing a few simple parameters from which the simulator constructs the network automatically.

According to OMNeT++, the structure of our simulator is modular. We defined the modules and their connections in the NED language, and implemented their functions in C++. The modules are the following:

Mobile: This component represents a mobile device which changes its location and speed periodically, and sends data requests to servers and other mobiles, as well as receives data from these. The properties of this module are summarized in Table 1.

home_IP	The home IP address of the mobile
careof_IP	The actual IP address of the mobile
myHomeAgent_IP	The IP address of the home agent
BC	Binding cache for storing IPv6 binding information
IsAtHome	True if the mobile is attached to its home network
Connected	True if the mobile is attached to an access point
x, y	Coordinates of the mobile
vx, vy	Movement speeds
exp_par_MOVE	Random exponential parameter for determining the time of the first movement

exp_par_TRA	Random exponential parameter for determining the rate of the data requests
BC_REFRESH_TIME	Determines the time interval between binding cache updates
DEL_ME_DELAY	The delay between the handover and the notification of the old access point
MAX_SPEED	The maximal initial speed of the mobile
MAX_DELTA_SPEED	The maximal acceleration of the mobile

Table 1. Properties of the mobile module

Air: It represents the radio interface, but now it simply connects Mobiles to the wireline network. There is only one Air module, because OMNeT++ can not handle dynamic connections properly. The properties of this module are summarized in Table 2.

APGateIDs	Array to store the IDs of the access points
toMobiles	Array to store the IP addresses of the mobiles

Table 2. Properties of the Air module

Access Point: These elements represent all physical radio access points belonging to the same subnet. Macro mobility handovers happen between these Access Points, micro mobility handovers happen inside an Access Point. Table 3. summarizes the properties of this module.

myIP	The IP address of the access point
visitors	Array indicating the attachment points of visitor mobile nodes

Table 3. Properties of the access point module

Router: This component stands for the whole wired network between Access Points, Servers and Home Agents. It is responsible for routing packets and simulates network delays as well. Table 4. contains the properties of this module.

SwitchTable	Array describing the connections of the incoming and outgoing queues of the router.
InQueue	Queue for incoming packets
OutQueue	Array for queues storing outgoing packets
Addressbook	Addressbook containing all Server and Mobile (home) IP addresses
ROUTER_SENDING_ PERIOD_TIME	Determines the frequency with which the router checks its outgoing queues
ROUTER_DELAY_TIME	The delay representing the wired network

Table 4. Properties of the router module

Server: Common Servers generate data packets as a reply to Mobile data requests. The properties of this module are summarized in Table 5.

myIP	The IP address of the Server
BC	Binding cache for storing IPv6 binding information
BC_REFRESH_TIME	Determines the time interval between binding cache updates

Table 5. Properties of the server module

Home Agent: Home Agents implement the mobile extension management by maintaining the binding between a Mobile's home and foreign address. Table 6. summarizes the properties of this module.

myIP	The IP address of the Server
BC	Binding cache for storing IPv6 binding information
BC_REFRESH_TIME	Determines the time interval between binding cache updates

Table 6. Properties of the home agent module

The modules are connected to each other according to the following figure (Figure 2.):

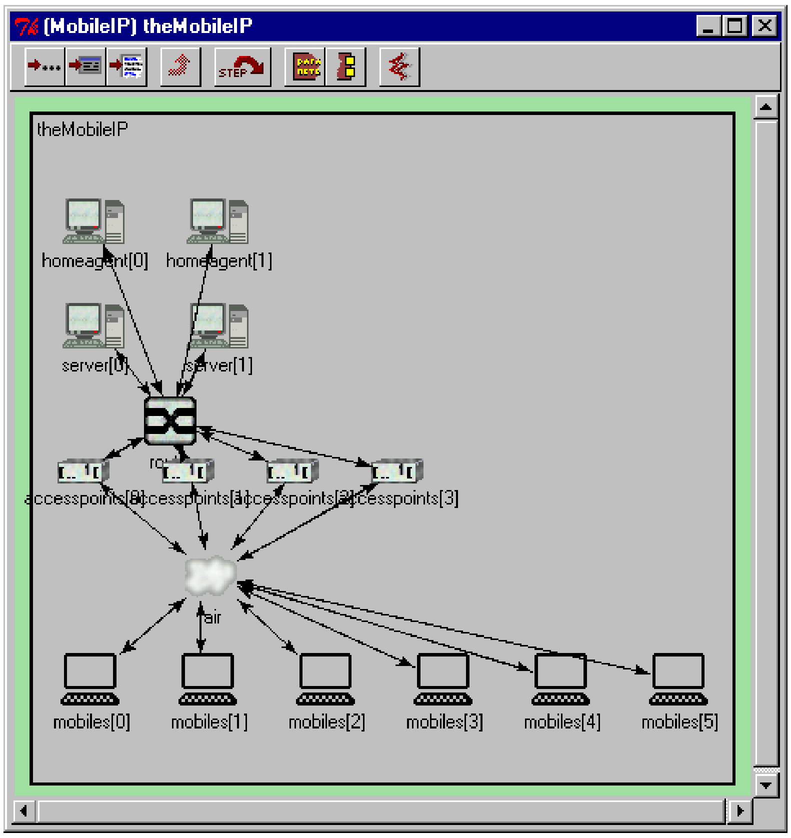

Figure 2. Sample screenshot of the simulation environment

3.2. Messages

The simulator is basically message-oriented. Modules can send messages to themselves (timer messages) or other modules (packets). These messages travel along links, either wired or wireless.

3.2.1. Mobile module's messages

Upon receipt of a packet, the mobile first checks whether it has a BU_EXTENSION (binding update extension), which indicates that the corresponding party has changed its care-of address since the last conversation. If so, the corresponding entry in the Binding Cache is updated. The handled messages are the following:

- **BC_TIMER (self message):** Timer message for refreshing the mobile's binding cache. This is a self message, which is generated with a periodicity determined by the parameter BC_REFRESH_TIME. It decrements the lifetimes of the binding entries with BC_REFRESH_TIME. If the remaining lifetime is less than TTSBR, a BINDING_REQUEST message is sent to the corresponding node.
- **MOBILE_BEGIN_TRANS (self message):** If the mobile is connected to an access point, it sends a DATA_REQ_AND_BINDING_UPDATE message to a server or an other mobile. If not, it sends an ADD_REQ message directly to the access point to obtain an IP address (this happens on lower layers, so it's not part of our macro mobility simulation). Finally it sends a MOBILE_BEGIN_TRANS self message with a delay determined randomly by the exp_par_TRA exponential parameter.
- **DATA (Server / Mobile → Mobile):** After collecting statistical information, the data packet is simply thrown away.
- **MOVE (self message):** This timer message makes the mobile node recalculate its actual speed and location coordinates randomly using the MAX_SPEED and MAX_DELTA_SPEED parameters. Knowing its new coordinates, the mobile node checks whether it has entered a new subnet (this is a lower layer functionality). If so, it sends an ADD_REQ message directly to the access point of the new subnet, to obtain a new care-of address. (This also happens on layers below IP.) Furthermore, after a delay determined by the DEL_ME_DELAY parameter, it notifies its last access point with a DEL_ME message that it has left its range. Finally, after updating the handover statistics, a new MOVE message is triggered after (about one second) delay.
- **ADD_RESP (Access Point → Mobile):** The mobile node receives its new care-of address for a freshly entered subnet in this message. Then it notifies its home agent with a BINDING_UPDATE message (an empty message with a BU_EXTENSION) of its new care-of address.
- **BINDING_ACK (Home Agent → Mobile):** In this message the Home Agent acknowledges the receipt of the mobile's BINDING_UPDATE message.
- **DATA_REQ and DATA_REQ_AND_BINDING_UPDATE (Mobile → Mobile / Server):** A corresponding mobile node asks for a data packet in these messages. The DATA_REQ_AND_BINDING_UPDATE version includes a BU_EXTENSION about the new care-of address of the corresponding node, if it

has changed since the last conversation, so the binding entries can be updated correctly. Both versions contain the req_length parameter, determining the length of the requested data packet, which is sent in a DATA message. Finally, the statistic variables describing the mobile–mobile communication are updated.

- **BINDING_REQUEST (Server / Mobile / Home Agent → Mobile):** This message informs the mobile that its binding entry is about to expire in the corresponding party's binding cache. The Mobile sends a BINDING_UPDATE message (an empty message with a BU_EXTENSION) in response.
- **BINDING_UPDATE (Mobile → Mobile / Server / Home Agent):** The binding information contained by the BU_EXTENSION of this message is already handled, so the empty BINDING_UPDATE message body is simply thrown away.

3.2.2. Messages over the Air

The functionality of the Air module is to simply relay messages between the mobiles and access points, irrespective of the message kind or contents. A packet loss or time delay can be set here. Packets received from a mobile node are transmitted through the corresponding gate to the destination access point and vice versa. The Air module maintains the mapping between its gates and the mobiles by "backwards learning": whenever a mobile sends a message via the Air module, its current care-of address is mapped to a gate. Messages coming from the access points are relayed to the corresponding gate towards the mobile if a mapping entry is found; if not, the package will be lost, and package loss statistics will be updated. This may happen if the sender is not yet aware of the fact that the mobile has left its former subnet. This module has no messages, jut relay all of them.

3.2.3. Access Point messages

The access point is the so called base station, the messages related to this unit are the following:

- **ADD_REQ (Mobile → Access Point):** A mobile node entering the range of the access point in question, asks for a care-of address. The access point chooses an unused address and sends it in an ADD_RESP message. Since the process of obtaining a care-of address in IPv6 takes place in layers below IP, these messages are sent directly between the Mobile and the Access Point in our simulation, instead of the ordinary packet relay through the Air interface.
- **DEL_ME (Mobile → Access Point):** A mobile informs the access point that it has left its range. The care-of address used by that mobile will be marked as unused.
- Every other package is simply forwarded towards its destination: messages from the Router towards the Air interface; messages from the Air towards the router (or back towards the Air, if the destination mobile is in the same subnet).

3.2.4. Self messages at the Router object

The router module consists of two parts: a router part simulating simple router functions, and a network part, simulating the network transfer on the wired medium. The router part consists of in- and outgoing queues connected according a switching table, while the network part is simply simulated by a delivery delay. The router also includes an Addressbook object, which contains the IP address of every server, home agent and the home IP address of every Mobile in the network. (These modules register their addresses upon starting their activity.) Thus, the global Addressbook object allows a Mobile to select a partner for communication easily. Messages:

- **ROUTER_INOUT_MSG (self message):** This message makes the router pick the first packet from its incoming queue, and put it in the corresponding outgoing queue according to the switchtable.
- **ROUTER_SEND_MSG (self message):** The router picks the first packet (if there is any) of every outgoing queue, and sends it to the corresponding destination (a home agent, access point or server). Finally it initialises a new ROUTER_SEND_MSG with a delay of ROUTER_SENDING_PERIOD_TIME seconds.
- Every other message is regarded as an incoming data packet, and will be put in the incoming queue. Then a ROUTER_INOUT_MSG is triggered with a delay of ROUTER_DELA Y_TIME seconds.

3.2.5. Servers' messages

Just like the mobile, upon receipt of a packet, the server first checks whether it has a BU_EXTENSION (binding update extension), which indicates that the corresponding party has changed its care-of address since the last conversation. If so, the corresponding entry in the binding cache is updated. The handled messages are the following:

- **DATA_REQ and DATA_REQ_AND_BINDING_UPDATE (Mobile → Mobile / Server):** A corresponding mobile node asks for a data packet in these messages. The DATA_REQ_AND_BINDING_UPDATE version includes a BU_EXTENSION about the new care-of address of the corresponding node, if it has changed since the last conversation, so the binding entries can be updated correctly. Both versions contain the req_length parameter, determining the length of the requested data packet, which is sent in a DATA message. Finally, the statistic variables describing the mobile–server communication are updated.
- **BC_TIMER (self message):** Timer message for refreshing the Server's binding cache. This is a self message, which is generated with a periodicity determined by the parameter BC_REFRESH_TIME. It decrements the lifetimes of the binding entries with BC_REFRESH_TIME. If the remaining lifetime is less than TTSBR, a BINDING_REQUEST message is sent to the corresponding node.
- **BINDING_UPDATE (Mobile → Mobile / Server / Home Agent):** The binding information contained by the BU_EXTENSION of this message is already handled, so the empty BINDING_UPDATE message body is simply thrown away.

3.2.6. Home Agent

Since the Home Agent intercepts packages on behalf of the remote mobiles, its first task is to check whether the packet is addressed to itself or a mobile belonging to it. If it's for a mobile, which is currently not attached to its home network, the Home Agent looks for a corresponding entry in its binding cache. If an entry is found, the package will be forwarded to the care-of address, otherwise it will be lost. On the other hand, if a package is explicitly addressed to the home agent, the message handling is the following.

First, the Home Agent checks whether the packet has a BU_EXTENSION (binding update extension), which indicates that its mobile has changed its care-of address since the last conversation. If so, the corresponding entry in the binding cache is updated. If the "Ack"

(Acknowledge) flag is set in the extension, the home agent confirms the receipt of the new care-of address by sending a BINDING_ACK (Binding acknowledge) message to the mobile. Messages:

- **BC_TIMER (self message):** Timer message for refreshing the server's binding cache. This is a self message, which is generated with a periodicity determined by the parameter BC_REFRESH_TIME. It decrements the lifetimes of the binding entries with BC_REFRESH_TIME. If the remaining lifetime is less than TTSBR, a BINDING_REQUEST message is sent to the corresponding node.
- **BINDING_UPDATE (Mobile → Mobile / Server / Home Agent):** The binding information contained by the BU_EXTENSION of this message is already handled, so the empty BINDING_UPDATE message body is simply thrown away.

3.3. Simulation scenarios

The graphical representation does not show the actual physical location and movements of the mobile nodes. Our network is square shaped, and contains subnetworks that are also squares. The size of the subnets is 10x10m, and there are *n*x*n* subnetworks in our network, where "*n*" is a global parameter. Therefore there are *n*x*n* access points. The global parameter "server_number" determines the number of servers, "homeAgent_number" determines the number of home agents, and "mobile_number" determines the number of mobile nodes. At the beginning of the simulation, the mobile nodes are placed on a random location, with a random speed (*x*, *y*) of movement. This speed changes periodically and randomly during the simulation. If a mobile leaves the simulation area, it enters on the "other side" again. This scenario can be seen on Figure 3. With the help of separators between cells any obstacle (like walls etc.) can be simulated.

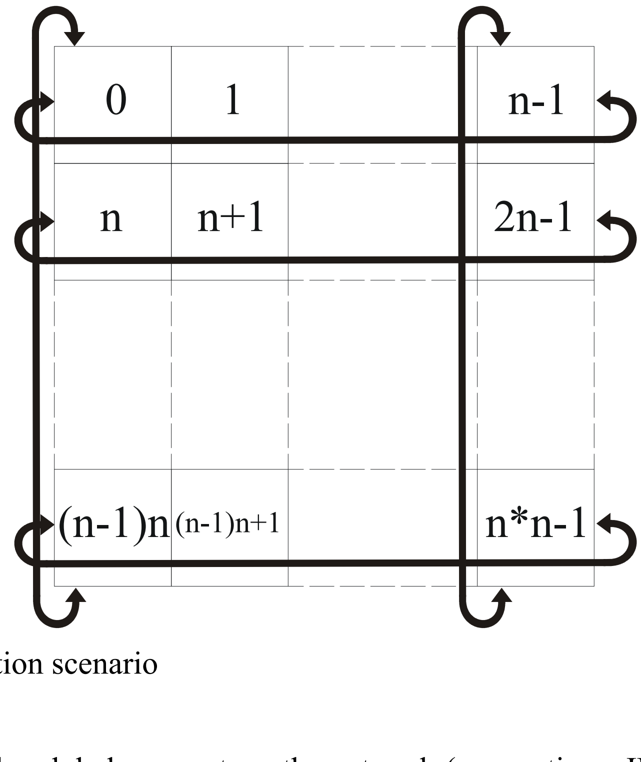

Figure 3. Simulation scenario

After providing the global parameters, the network (connections, IP addresses, etc.) is constructed automatically by the simulation environment.

Mobile nodes begin their activity with registering themselves at their current access point. If this point of attachment is not in their home subnet, they get a care-of address for using the foreign subnet temporarily. The process of obtaining this address takes place on layers below IP, therefore we simply simulate it by exchanging direct messages between the mobile and the access point. Having received a care-of address, mobiles randomly choose a partner for communication by selecting a destination IP address from the global Addressbook. Then they start communicating by sending a data request to the destination party, who will reply with a packet of the length given in the data request. mobiles also maintain a binding cache, which stores the care-of address – home address entries for each mobile they are communicating with, together with a lifetime value for each entry.

Packets sent by mobiles first arrive to the Air module, which simply transfers them to the access point belonging to the mobile in question. Similarly, in the other direction, access points forward the packets addressed to their current mobiles to the Air interface, which transmits them to the mobiles – unless they've left the subnet without informing its access point yet; in this case, the packets will be lost.

Access points serve as an attachment point to the wired IPv6 network. They forward packages between the wired network (the router module) and the mobiles currently attached to them, via the Air module. Also, they provide their mobiles with care-of addresses, so that they can be addressed at their current location.

The router in our simulation represents not only a router on the border of the wired network, but also the whole wired network beyond. It stores incoming packets in its in-queue, and relays them with a delay towards their destination.

Servers have a simple task: they send data packets of the desired length in response to data request packages of mobiles. They also maintain a binding cache to keep track of the care-of addresses of the mobiles they are communicating with.

Home agents also maintain a binding cache for their mobiles. Their main activity is to listen for packages sent to mobiles, which are not currently present in their home network. If such packages arrive, home agents intercept and forward them to the destination mobile, using the corresponding binding entry. This is only possible if home agents always have a valid binding entry for their absent mobiles. Therefore, Home Agents reply with a binding acknowledge messages to a binding update. A mobile has to repeat unacknowledged binding updates.

4. Results

To validate the reliability of our simulation environment, we performed some measurements. The ability of OMNeT++ ensured that the data collection and evaluation was much more easier than in case of a program written in plain C/C++, with not more than about 30-40% of extra overhead. The following statistical parameters can be obtained from our simulator:

- Packets lost in the Air module (packets/sec)
- Total packets transmitted across the Air module (packets/sec)
- Handover point of times for the mobile stations
- Average number of handovers of the mobile stations
- Total packets received by the mobile module (packets)
- Average packets received by mobile modules (packets/sec)
- Average delay of the data packets at the mobile module

- Average delay of the data request packets at the mobile station
- Proportion of the triangle routed packets received by the mobiles
- Ratio of triangle routed and directly sent data packets received by mobile modules
- Ratio of the triangle routed spontaneous data packets received by the mobile stations
- Number of data request packets received by servers (packets/sec)
- Average delay of data request packets at the server modules

4.1. Delay of the data packets

Figure 4. shows the end to end delay of the data packets received by two different mobile station modules, in a length of time.

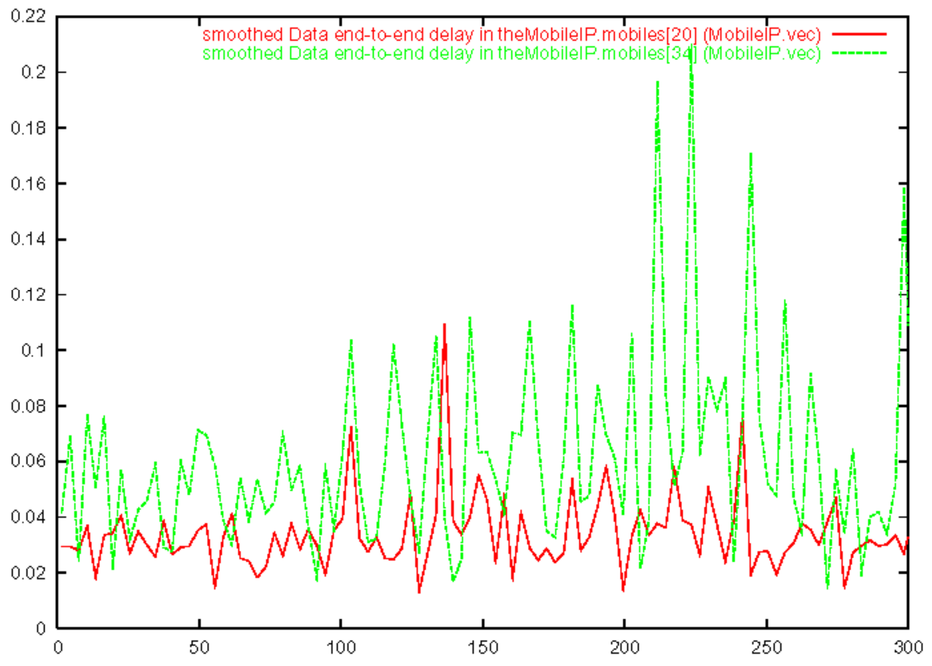

Figure 4. Data end to end delay

4.2. Triangle routing

With the limited capability of mobiles and network overhead caused by triangle routing the optimization of the binding cache's size and the binding entries' lifetimes is very important. We demonstrate this issue in different network scenarios. We investigate different statistics like rate of packets sent via triangle routing, rate of packet loss, handover frequency, etc.

We examined the Binding management methods of IPv6, which aim at optimizing the network for less triangle routing. The problem is illustrated by the on Figure 5.

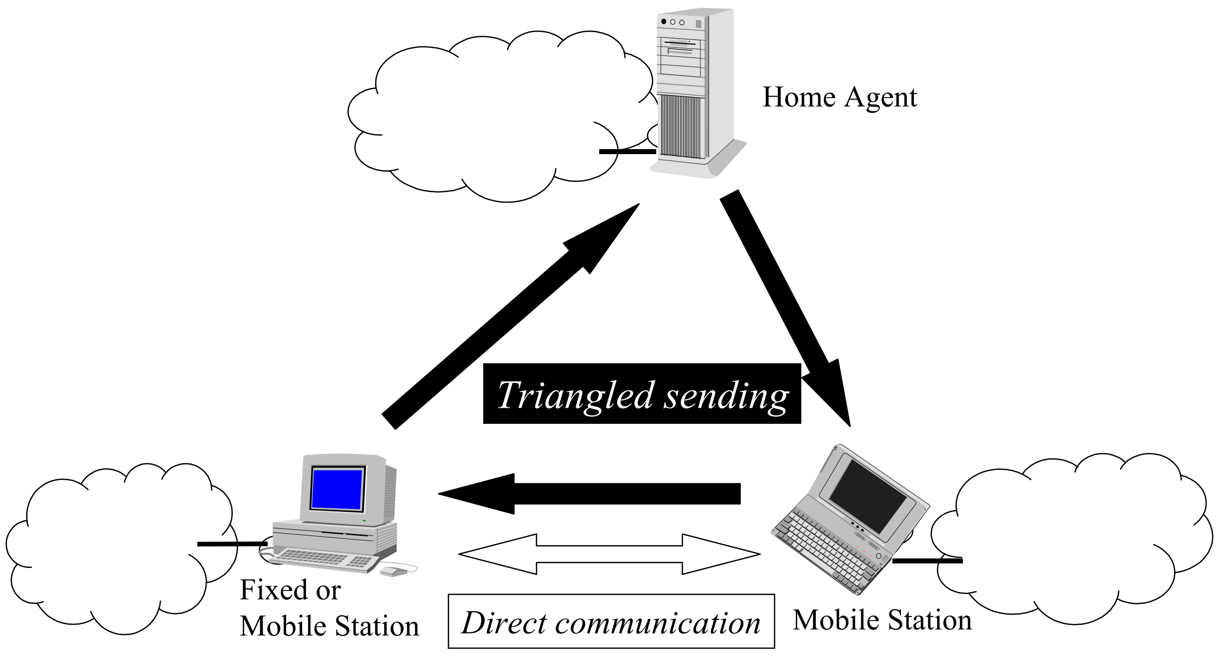

Figure 5. Triangled and direct sending

We investigated a network consisting of 50 mobile nodes, 9 subnets, 5 servers and 7 home agents. (In case of a smaller network, we could not draw general conclusions, while a bigger one would not provide us with more information.) First we examined the effect of the Binding Cache's size on the rate of triangled packets, then the effect of the binding entries' lifetimes on the rate of triangled packets. We traced only the data request and spontaneously sent packets, because these are the only packets in our network to cause triangling.

4.3. Examining the Binding Cache's size

By running our simulator with different Binding Cache sizes between 1 and 55, we came to the following results seen on Figure 6.

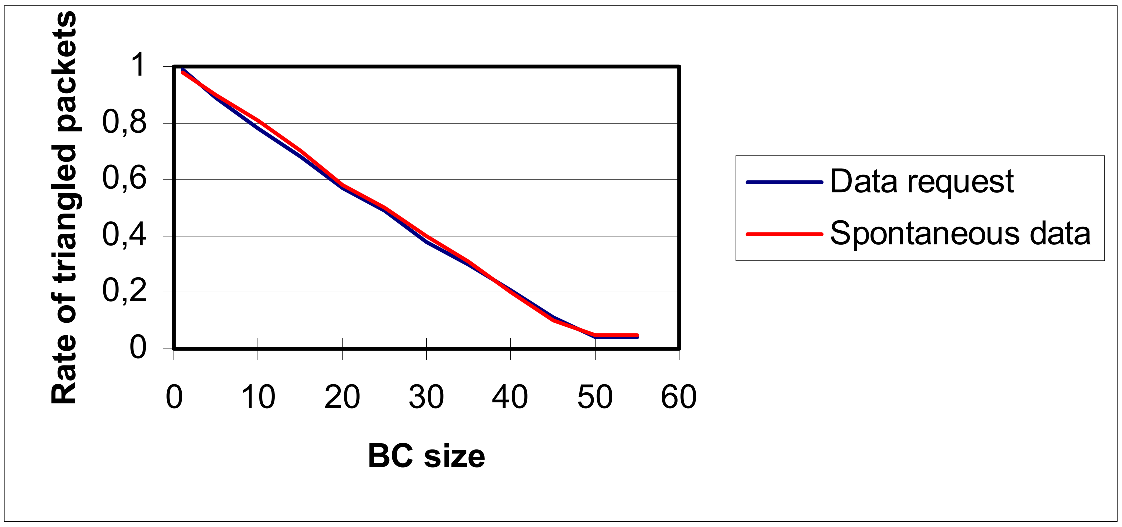

Figure 6. The effect of BC size

It can be seen that increasing the cache's size linearly decreases the rate of the triangled packets.

4.4. Examining the binding entries' lifetimes

By running our simulator with different entry lifetimes between 0 and 100 seconds, we came to the following results, as seen on Figure 7.

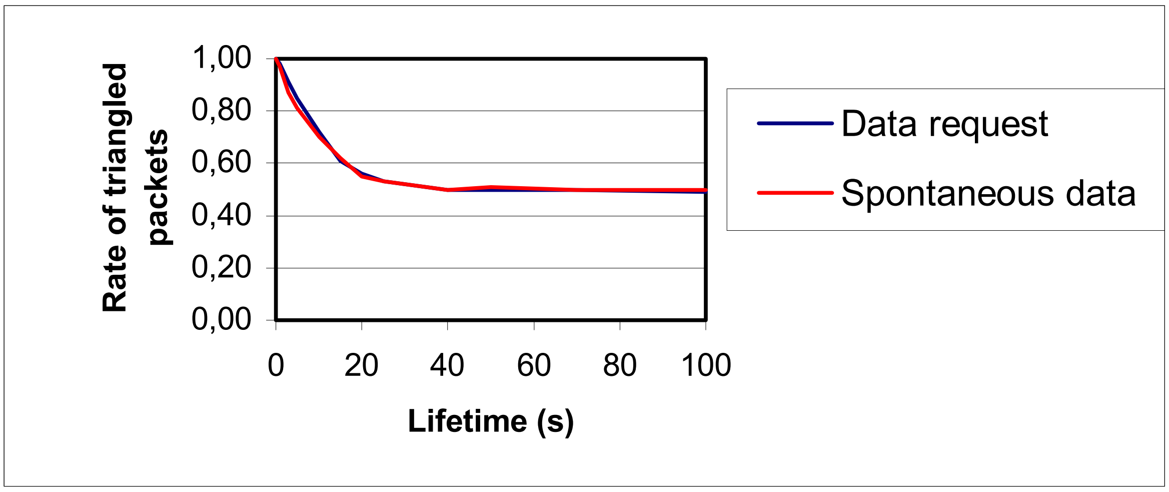

Figure 7. The effect of BC lifetime

It can be seen that by increasing the lifetimes first quickly decreases the rate of the triangled packets, until it reaches the fifty percent level. (The Binding Cache's size in this scenario was 25, that is, the half of the mobile's number.)

5. Conclusions and future plans

In this paper we have introduced the results of our work with the mobile IP macro mobility simulation environment developed in OMNeT++. We have completed the simulation environment, tested in different scenarios the results of binding lifetime management simulation and analysis. The simulator was verified by theoretical results. The current version of the simulator is capable of simulation of mobile terminals operation at willing IP environments.

In the future we would like to improve it so that it can be used to design real IPv6 based mobile networks. Some goals:

- simulating packet loss on the Air interface
- sophisticated statistic tool
- dynamic graphical representation

Since our simulator investigates the IP mobility extensions in general, it can also be used for researching various fields of IPv6 mobility.

Acknowledgement

This project is supported by OTKA F032268 (Hungarian Scientific Research Found)

References:

[1] Thomas Eklund: IP version 6 – The next Generation Internet Protocol, 1996

[2] David B. Johnson: Mobility Support in IPv6, Intenet Draft, http://www.ietf.org/internet-drafts/draft-ietf-mobileip-ipv6-14.txt, 2000

[3] Preetha P. Kannadath and Hesham El-Rewino: Simulating Mobile IP Based Network Environments, University of Nebraska at Omaha, 2000.

[4] Charles E. Perkins: Mobile IP – Design Principles and Practices, Addison-Wesley, 1998

[5] Thomas Narten, et. al.: Neighbor Discovery for IPv6, RFC 2461, http://www.ietf.org/rfc/rfc2461.txt, 1998

[6] Susan Thomson, Thomas Narten: IPv6 Stateless Address Autoconfiguration, RFC 2462, http://www.ietf.org/rfc/rfc2462.txt, 1998

[7] Derek Lam, Donald C. Cox, Jennifer Widom: Teletraffic Modeling for Personal Communications Services, IEEE Communications Magazine, 1997. February

[8] Varga András: OMNet++ version 2.0 User Manual, BUTE, http://www.hit.bme.hu/phd/vargaa/opp-docs/usman.htm, 2001