Universität Karlsruhe (TH)

Institut für Telematik

Discrete Event Simulation And Evaluation Of A Firewall Aware Architecture For Mobile IP

Diploma thesis

at

Institute for Telematics, Department of Informatics
Universität Karlsruhe (TH),
Karlsruhe, Germany

in cooperation with

Department of Informatics and Networks,
Ecole nationale supérieure des télécommunications (ENST),
Paris, France

by

cand. inform.
Artur Hecker

Date of registration: 01.10.2000
Date of return: 31.03.2001

Supervisors:

Prof. Dr. Dr. h.c. mult. Gerhard Krüger, University of Karlsruhe
Prof. Dr. Samir Tohmé, ENST Paris
Dr. Ing. Günter Schäfer, ENST Paris
Dipl. inform. Frank Pählke, University of Karlsruhe

Ich erkläre hiermit, dass ich die vorliegende Arbeit selbständig verfasst und keine anderen als die angegebenen Quellen und Hilfsmittel verwendet habe.

Artur Hecker

Acknowledgements

I'd like to thank Günter Schäfer and Frank Pählke for having given so generously of their time during my research for this paper, and for providing an insight into scientific work. Special thanks to Andras Varga for his outstanding, fast and precise mail-support with OMNeT++. Thanks also to Erik Blaß and Robert Rodewald, not only for their proofreading but also for encouraging me during this work. Thanks to Konstantinos Dimou for patiently hearing my explanations and to Basheer Dargham for his qualified help with the GNU software. Many thanks to all the people at the ENST, Paris who have made this work possible and to all my friends in Paris and Karlsruhe who motivated me all over the time.

Thanks to my whole family, in particular to my mother and to my father, for being there for me during my studies.

Abstract

Since Mobile IP in its current available version still has a lot of inconvenient disadvantages, several new proposals and architectures have been researched in the last years trying primarily to improve the security, to facilitate the system configuration and administration and to provide better support for fast handoffs. One of these projects called FATIMA has been developed at the University of Karlsruhe for one and a half years. To prove the feasibility, to see the possibilities of the main used ideas and to test the tangibility of the new concept, this work was issued.

This work consisted of four main steps. As the first step the present FATIMA draft had to be re-read and verified for concept errors. As the second step, a concept for a verification and performance comparison of FATIMA was developed. The decision has been taken to build a discrete event simulation of a FATIMA network based on OMNeT++ simulator. In the third step, the developed concept has been realized. During the implementation work, several draft changes have been proposed and some concept errors could be found. The implementation additionally included the simulation of standard Mobile IP agents in order to be able to obtain comparable results. In the fourth and final step, different scenarios with different numbers of mobile nodes and in different modes (FATIMA-only, mixed, Mobile IP only) have been tested providing first comparative performance results for the planned features. Eventually, these results have been evaluated, illustrated and interpreted.

Table of Contents

Table of Illustrations

1 Introduction

1.1 Motivation

Since 1993 the Internet experiences a rapid, boom-like growth. The number of attached computers, so-called hosts, grows exponentially. With this spreading, the Internet finally says good-bye to its original reputation of being a military or scientific network. Its techniques become conventional. The rising commercial meaning in the following years and the associated demand on computer and network markets force the industry to accelerate the search for new ideas. Fortunately, the miniaturization and the availability of common computer technology show new opportunities. From the gray inconspicuous box under the table the computer evolves to something really personal. Laptops, notebooks, personal digital assistants and even mobile telephones are to understand the same protocols, to support the same standards and last but not least for marketing reasons to be connected to the „information highway“, the Internet. The mobile computing experiences the same boom as already the Internet itself.

However, the original Internet protocols, above all the IPv4 protocol →[RFC0791], were developed regardless of any mobility concerns. 1996, after successful research for the next generation protocols like IPng, [IETF] (*Internet Engineering Task Force*) defines a new extension for IPv4, so-called *Mobile IP* →[RFC2002]. Compatibility and relative implementation simplicity of this solution are remarkable, but unfortunately some weaknesses prevent it from being widely used. Additionally, the security considerations in the modern Internet sometimes completely foil even the basic functionality.

In the past years a lot of engineering work has been invested to extend the released standard, which shows how important a mobility solution for the Internet is considered by the industry. The consequence is the opulence of new propositions and proprietary solutions, which are rarely usable one with each other. Nevertheless, this active work provides a real source of ideas, which hopefully will be finally combined resulting in an Internet standard fulfilling the most significant current wishes.

Among other propositions there is a system draft called [FATIMA], which is currently being developed at the Institute for Telematics, University of Karlsruhe, Germany. Basically, this proposition integrates a lot of known ideas and methods into a complete IP-mobility solution, which is claimed to be downwards compatible to the Mobile IP standard. However, the relative complexity of the system is much higher than in the existing standards. So, a lot of questions have been raised forcing to doubt whether the presented solution is usable at all under real circumstances.

In order to be able to give answers to some of these questions the need for an applied research with practical results has been recognized. Thus, the formulation of this work has been issued with the main goal of achieving a qualitative and quantitative evaluation of a Mobile IP network according to the new [FATIMA] draft. Thereupon this task has been carried out at the Department of Informatics and Networks, ENST Paris, France.

1.2 Outline of this work

After some motivation for the support of mobility in the Internet has been given, we want to point out which real possibilities exist on the market and in theory. Trying to specify what is and what is not meant by mobility in this context, we will first of all present the only current Internet standard which has been defined by [RFC2002] (*Request For Comments*). With the help of this example it will be probably possible to understand the fundamental mechanisms and to show the advantages and also the weaknesses of some of the used solutions. Where improvements and reasonable workarounds for the problems have already been provided they will be shortly presented. In particular we will talk about the problems with the firewall-support and the solution proposed by [RFC2344].

After having presented these known issues and therefore having built a common basis, we will introduce a new concept to build a firewall-aware IP network supporting mobility, called [FATIMA]. Since this new proposal is still work in progress, it is not sure whether all of the mentioned ideas will work fine / work at all in reality. This way we will approach the main objective of this work: the evaluation of the network environment built according to the FATIMA-concept. Taking a look at the basic possibilities to evaluate such a complex system with a lot of mobile participators, we will pick out the simulation as the best method and show its advantages. After having shortly presented the chosen free simulation engine, OMNeT++ →[OMNeT Web], the essential conceptual decisions and the associated simulative limitations will be discussed. The completions and even changes of the FATIMA-concept, which have become necessary during the implementation of the simulation, will be explained separately.

Since the implementation itself was the main part of this work, it will be discussed in detail. The interesting classes and methods and their purposes will be introduced in order to explain the functionality of the produced software and to make it reusable. Finally, we will come to the achieved quantitative results and draw some comparisons between the [RFC2002] compliant environment and the FATIMA network, measured with our simulation software. Interpreting the results where it is possible we will try to give a small outlook on what we can await in the next time.

The instructions on how to run the simulation and to build your own network and more details will be given in the appendixes.

2 Mobile IP: Mobility extension for the classical Internet

2.1 Definition of mobility in the classical Internet

In classical telecommunication we often distinguish between certain so-called layers. Theoretically, a mobility mechanism could be set up on many of these. The well-known ISO/OSI communication model consists of seven different layers. Most of them could provide some mobility support. That is why first of all it should be stated more precisely what we want to understand under mobility in this context and why any other solution would not be satisfying.

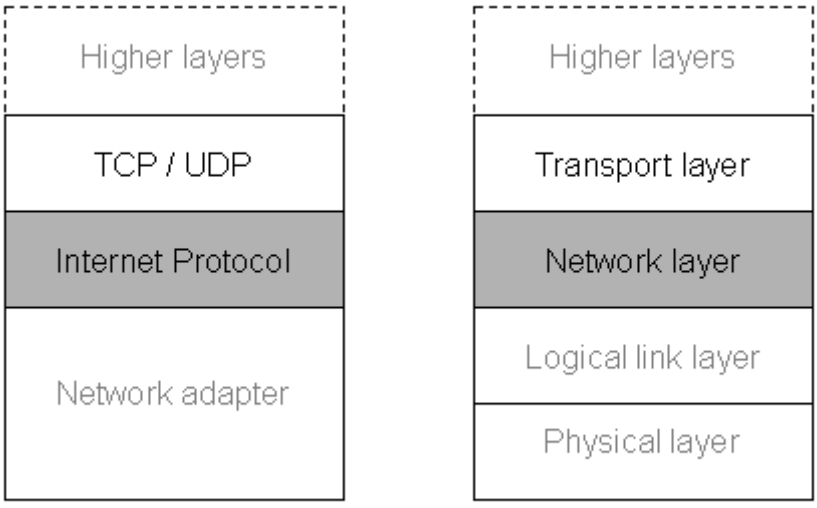

Figure 2.1-1 Mobility definition in the network layer (TCP/IP - ISO/OSI)

The classic Internet is based on a protocol simply called IP (*Internet Protocol,* →[RFC0791]). This protocol makes direct use of some physical network adapter and more or less represents the abstraction of the particular physical layer. Reclassified into the ISO/OSI layers (see Figure 2.1-1) it would occupy the network layer since it enables the inter-network communication by giving some unique addresses to all the participants and by introducing hierarchical *routing* between two arbitrary participants, i.e. direction-finding mechanism. In the classic Internet this unique address (*IP-address*) is the only well-defined identifier by which mean it is possible to tell apart all the computers connected to the Internet (so-called *hosts*). If we want to talk about host-mobility in the Internet, we therefore have to presume that this address remains unchanged. All the protocols above use the basic IP functionality to establish a connection between two hosts and add their own functionality. E.g. the transport protocols primarily add the differentiation on the application-level. The transport protocols used in the Internet are TCP →[RFC0793] and UDP →[RFC0768], both of which define so-called *ports*. In this way it is possible to address some specific application running there and not just the whole host, like by the means of IP. All the higher layers are mainly designated to some extended application support. For that reason they are not part of the Internet protocol suite.

Thus, where could we implement the necessary mobility support? Apparently, the higher levels are not a good place since they are not standardized in the classic Internet definition and, moreover, each solution implemented higher in the layer hierarchy is not available for the levels beneath. Since the physical and the logical link layers are implemented in hardware on the appropriate network adapter, we can't add any generally available mobility there. So, it seems that we have two possibilities: the network and the transport layers. The difficulty with the transport layer as the mobility supporting instance does not only arise the problem of having several transport protocols and in this manner more programming work. As explained above, we also have to leave the IP-address unchanged. The IP-address is a part of the definition of IP itself and its use is restricted by the rules defined there. Naturally, the transport protocols couldn't guarantee the use of whichever IP address, since they just use the network layer, which is responsible for the addressing. Hence, using transport layer for mobility reasons ends in searching for some new methods for host identification, which do not exist in their standard definition.

Obviously, it is inevitable to insert mobility concepts into the network layer, i.e. to extend the standard IP definition to support host movements within the whole Internet without changing the given address or the remaining IP-configuration. That is what we will call *mobility* in this context. This mobility definition provides us a great advantage of having defined the mobility on the lowest possible level. With this definition we will not have to change our transport layer protocols in order to support host movements. However, it bears some problems, which will be explained in the next chapter.

2.2 Routing in IPv4 vs. mobility

The Internet is a network interconnecting different local networks. These connected networks could be some scientific, private, governmental or enterprise-networks with whatever physical layer. The owners just have to ensure that his network supports TCP/IP protocol suite, that the network is physically connected to some Internet access point and to order some official IP-address segment from a responsible organization.

The obtained IP-address segment will be contiguous and there will always be a possibility to name a network base address and some mask describing if some IP-address belongs to this IP-segment or not. Furthermore, in the original configuration style all the addresses in this network would be distributed statically, i.e. the same physical computer would always come online with the same address. In the meantime some protocols exist for dynamic configuration →[RFC2131] of a booting computer but, as explained above, this solution brings up the question of how to distinguish between two different hardware devices on this level if they use the same IP-address. Besides, some computers have to be configured with the same address, like e.g. the access points to other networks (*IP-router,* i.e. a host connected to more than one physical network and knowing where to send the packets to) and typically some additional hosts providing network services. So, in general we should speak about a static IP-configuration. Given an IP-address we can therefore always determine where this address comes from, i.e. the addresses are even bound to some specific location.

Like the configuration, the IP-routing is in fact static, too. Based on some tables held by the routers all over the world, every change or update takes a while until it is accepted and every concerned packet takes the new path. The greater the load on some router, the

shorter its routing table should be. For that reason the routing tables of the real backbone-routers hardly ever contain host entries for single hosts; all the routing is done by comparing some IP-address with some network mask.

Provided with these facts, we are now able to understand, that, connected to some desired network with its old address after having moved, a host would never get packets addressed to it. Since the routing is based on the network masks and because the IP-addresses are locally bound, all the sent packets would still arrive in the original network of this host and not at his new location. On the other side, the propagation time for a so-called *binding update* (i.e. routing entry update for this specific host) within the whole Internet would be terribly long. The complexity of such a solution would be enormous: assuming that we have several thousands mobile hosts, we would have to manage routing tables with more than thousand entries in *every* router.

As we can see, the mobility support seems to conflict with the basic routing rules of IPv4 →[RFC0791], the current version of the Internet Protocol.

2.3 System overview (RFC2002)

Since we will be talking about mobility here, we will have to deal with (at least two) different networks, different hosts and probably some systems, providing support for mobility. That's why it is very important to make terms about the nomenclature first and to work out the necessary and minimal scenarios, which have to be managed by a system claiming mobility support (see Figure 2.3-1).

We will begin with the subject of this work, some host which is potentially mobile and which thus could be moved away from its momentary location i.e. network. Such a host will be called *Mobile Node* (MN) in this document. The original location of such an MN, its pre-configured network, will be called *Home Network*. In this network the MN has been given an address, which is the *Home Address* of this MN. Some network visited by our MN at some point of time will be thus called Visited Network or *Foreign Network*. Normally we will need some temporary address for each Foreign Network describing the actual location of our MN. This address is the *Care-of-Address* (CoA) of our MN. A host communicating with our MN in some possible scenario will always be called *Correspondent Node* (CN). Please note that the CN doesn't have to be a member of one of the two mentioned networks. Luckily, in this case we don't need to deal with its origin.

In most scenarios we will not need to deal with more than two networks at the same time. The only involved networks will remain our Home Network and the Foreign Network being visited. The change from one network to the other will be called handover. The interesting case of a handover between two Foreign Networks wasn't explicitly defined by [RFC2002], another reason why we will hardly ever need more than two networks in our scenarios. Nevertheless, the [RFC2002] "IP Mobility support" issued in October 1996 defines some system components that have to be explained more precisely.

The part of the system installed in our Home Network maintaining all the registration and data traffic support is called *Home Agent* (HA). The presence of at least one HA is obligatory. Its counterpart in the Foreign Network is called *Foreign Agent* (FA). In

order to support the spreading of Mobile IP in the beginning the usage of an FA has been defined optional. However, it is explicitly recommended and the standard deals a lot with the definitions around the FA, which is one of the most important parts of a Mobile IP system. If the FA is not used its functionality has to be integrated into the MN. Since we cannot guarantee that every potential destination network is equipped with a RFC2002-compliant Foreign Agent, every MN has to be capable of dealing with the control traffic and establishing the connections without the assistance of a FA. Transferring the functionality into the mobile nodes enables every administrator to use mobile computers and Mobile IP in his own network, because he does not have to rely on any support in the visited networks. Nevertheless, it doesn't change the main protocol ideas much, since each necessary function has to be provided in both cases. In this particular work, we will concentrate on the cases with an available FA since these require more control traffic.

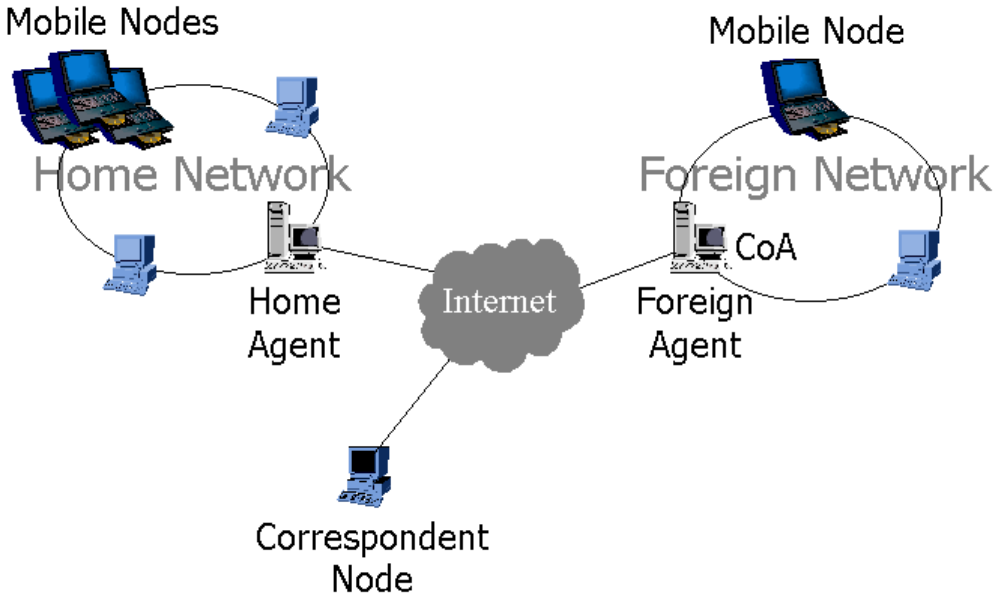

Figure 2.3-1 An overview of a Mobile IP system with the main participants

The basic idea of the Mobile IP concept solving the main difficulties presented in Chapter 2.2 is the following illustrated in Figure 2.3-2. Every MN visiting some foreign network obtains a valid address in that network by some mean. Then it registers with its Home Agent by sending it an IP datagram containing its original Home Address and the obtained CoA. From now on the HA installed in the Home Network intercepts all the packets destined to an absent MN and forwards them to the actual location of the MN which is represented by the received CoA. The forwarding is done by establishing a tunnel to the CoA i.e. by sending the original packet as payload of the new packet destined for the new location. The answers of the MN are then sent directly to its CN. Having returned home an MN informs the HA which then stops intercepting the packets.

One can easily see that this simple explanation makes use of plenty of different mechanisms like e.g. dynamic IP configuration in the foreign network, intercepting the packages destined for other hosts or IP-tunneling. Apart from that, Mobile IP provides some identification mechanisms to ensure that the HA is really communicating with one of its MNs and not with some attacker. And as usual, in reality we need some more work to be done. Fortunately, Mobile IP is based on well-known protocols defined by other RFCs. In the following we will therefore discuss which mechanisms are used in which manner, which answers are required for which requests and take a look at some typical scenarios and the exact packet formats. However, we do not pretend to describe the whole Mobile IP standard since we would have to copy the complete [RFC2002] otherwise. We will try to describe the parts, which are considered the most important in this context.

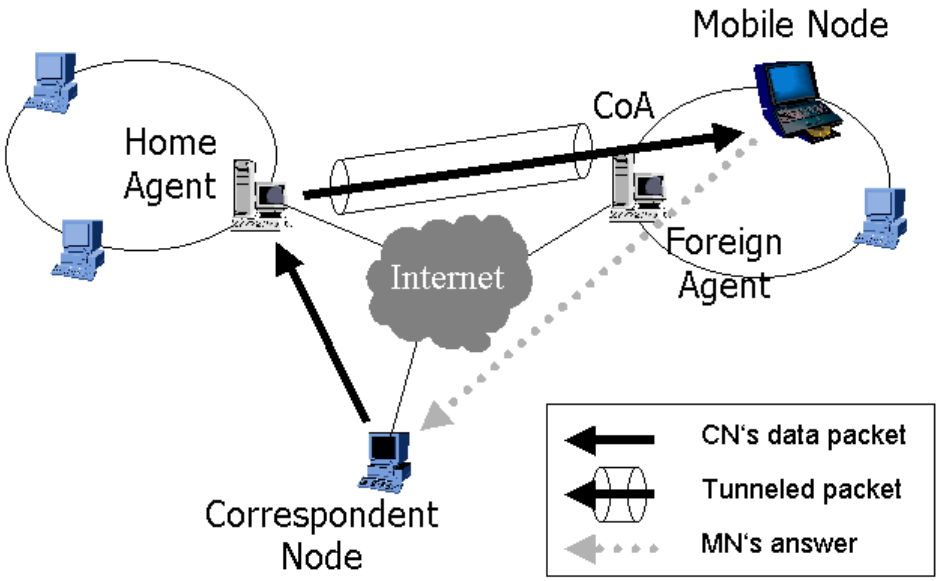

Figure 2.3-2 Basic idea of the Mobile IP concept

As with every mobility supporting system, Mobile IP distinguishes between control and data traffic. Control traffic doesn't transport any data. This traffic is necessary to track the momentary locations and situations of the mobile participants in order to be able to establish data connections with these at any point of time. Generally, data traffic is responsible for data transport with respective solutions for any imaginable difficulties in the mobile area, like complex routing, unstable links, limited bandwidth, etc. In this manner every used mechanism can be assigned to one of these two groups.

The mobility support of a Mobile IP system begins when the MN receives a so-called advertisement message, which was not sent by his Home Agent. Advertisement messages are broadcast messages sent periodically by every Mobile IP agent. The only exception is the case of a network supporting Agent Recognition by Link Layer means, which is rare and will not be discussed here. Having received an advertisement sent by

some unknown agent, a MN waits for the adequate message of his responsible Home Agent. If the message does not arrive within the time out time, the MN assumes that it has left its home network and sends the first registration request message. As long as a MN receives the periodic advertisements of his Home Agent, it acts almost like every other "normal" Internet host with the only exception that it has to process the advertisement messages.

For its advertisements, Mobile IP extends standard methods defined in [RFC1256] *ICMP Router discovery messages* and implements a mechanism called *Agent Discovery*. The extension added by Mobile IP carries some specific data, which is essential for a Mobile IP system, because the [RFC1256] was originally designed to configure a host connected to a network with the responsible router or even to enable some load balancing in a case where several different routers are available. Mobile IP needs to be capable of differentiating between the two agent types and the offered services since a network administrator still could want to prohibit some functionality defined optional by the Mobile IP standard.

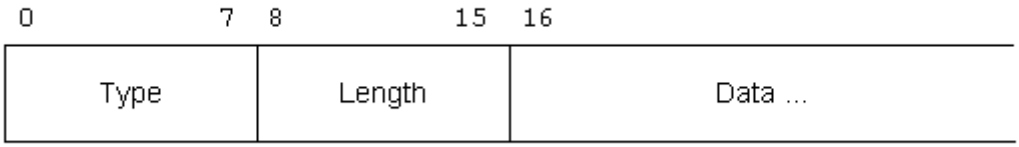

Figure 2.3-3 Generic extension format used in Mobile IP

The extension, herein before mentioned, is a general mechanism separately defined in the Mobile IP standard. It is used with the advertisement messages but also with the registration messages, because it allows to easily extend almost every packet format. In particular, it's possible to append one extension to the other without having to change anything in its predecessor. As we can see in the Figure 2.3-3, there are just two bytes indicating the type of this extension and the length of the following extension data. The type of the extension determines the format of the Data field. Mobile IP inserts such a frame in the data field of the deeper protocol and corrects its overall length field. If needed, one extension can be appended to the other; the overall length field has to be corrected then. The range of the Type field is divided into two groups. When an extension with type 0...127 is received but not recognized, the whole message has to be silently discarded. When an extension with type 128...255 is received but not recognized, this particular extension is ignored but the rest of the message has to be processed.

There are two independent sets of extension types. The first numbering space is defined for the Mobile IP control messages. It is carried by the transport protocol used for these purposes in the registration messages (see later). The second numbering space is carried by the ICMP control messages →[RFC0792] like in our case by the *Mobile Agent Advertisement* message.

The Mobile Agent Advertisement message is a specific extension inserted directly into the ICMP Router Advertisement message as it is defined in [RFC1256]. The Type field of the ICMP-advertisement is set to a predefined value (16) in order to indicate that this

ICMP message is a Mobile Agent Advertisement message. The other fields of the ICMP message will not be discussed here →[RFC0792].

The information important in this scope is carried within the extension itself as shown in Figure 2.3-4. The Type and Length fields are parts of the extension itself and not of interest here. First of all, this extension determines by several flags if the submitting agent is a Home Agent (Flag H) or a Foreign Agent (Flag F). With the other flags it informs the MNs about both the offered optional features and the state of the respective agent (Flag B – "Busy"). The Sequence Number field is used for crash recognition. In the Registration Lifetime field the longest possible registration lifetime in seconds (0xFF – infinity) is published. And finally, the available Care-of-Addresses of the advertised Foreign Agent are supplied here which is probably the central informational part of such an advertisement.

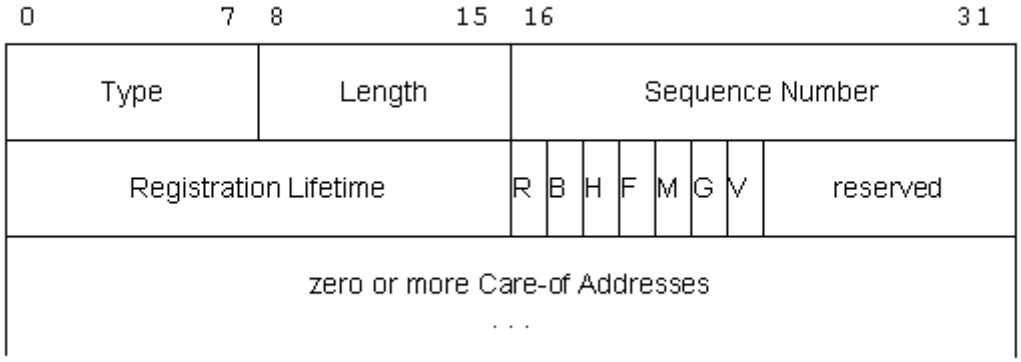

Figure 2.3-4 Mobile Agent Advertisement message as a specific extension

Having detected a handover by hearing and processing the periodic advertisements, a MN sends a registration request message to the last heard agent. This is the same procedure for the handovers between two foreign networks and even if the MN has moved back to its home network as you will see below. In the cases where no Foreign Agent is available, a MN waits three timeout times in sequence. Then it assumes that it has lost contact to the last network. Thereafter it obtains some valid IP-address in the new network (e.g. through DHCP, [RFC2131]) and registers directly with its Home Agent with this new obtained address, called *co-located CoA* in this case. In the case where a Foreign Agent is present, an address of this FA itself becomes the CoA, which is then consequently called *foreign agent CoA*. In the last case a MN can keep on using its home network address since it is always communicating through the FA.

Mobile IP uses UDP (*User Datagram Protocol,* [RFC0768]) as a transport protocol for its registration messages. A host serving as an agent has therefore to hear for the incoming (de-)registration requests on the well-known port assigned to Mobile IP which is the UDP port number 434.

The registration messages are transported as the payload of the UDP messages. All the registration messages have to begin with a Type field, which determines the kind of the following registration message and therefore the format of the packet. Two types have been defined till now, a *Registration Request* message (Type 1) and a *Registration Reply* message (Type 3).

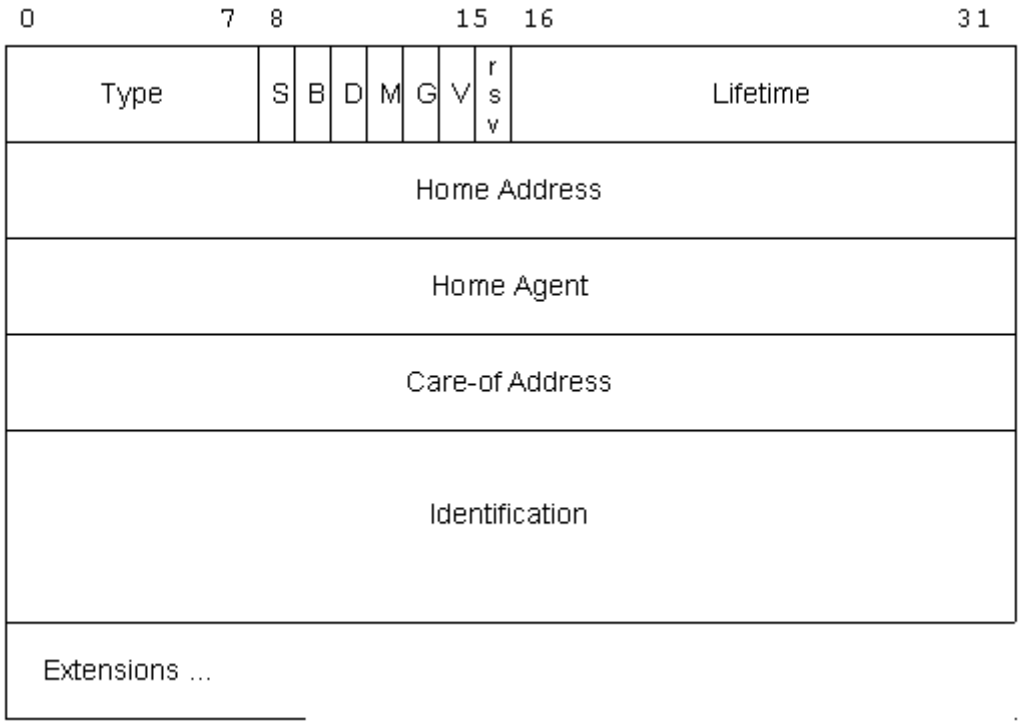

Figure 2.3-5 Mobile IP Registration Request message

As illustrated in Figure 2.3-5 the Registration Request consists of six basic fields. The first field divided into seven sections carries the flags by which an MN can request some services from his Home Agent. For example, Flag B requests broadcast message forwarding and Flag D describes the mode in which the MN is currently working indicating that the decapsulation is done by the MN itself in the case in which the MN is using a co-located CoA. The next flags determine the used encapsulation type. If none of them is set standard IPIP encapsulation ([RFC2003]) is used. The Lifetime field is used by the MN to request for which period of time it wants to register. A value of zero indicates the request for *deregistration*. If all the bits are sets, infinity is meant. The next field, Home Address, contains the home IP-address of the registering MN. Home Agent is the IP-address of the responsible HA to which this request will be sent. This field is needed since the IP-header of the packet carrying this message doesn't necessarily contain this address – if an FA is present the message is addressed to the FA, which then needs a possibility to forward the request to the responsible agent. In the next field an MN publishes the obtained Care-of-IP-address, either the one obtained from the advertisement or by other means like DHCP. The Identification field carries a 64-bit number, randomly constructed by the mobile node, used for matching Registration Requests with Registration Replies and for protecting against replay attacks of registration messages.

If a Foreign Agent is present it receives the Registration Request from the MN. It analyses the message and if there is no problem with it, it forwards the message to the included HA. If a FA is not currently capable of taking care of this MN it has the possibility to response with a Registration Response message carrying one of the correspondent failure codes, which are defined in the Mobile IP standard. For example, the FA could already be too busy dealing with a too great amount of MNs. If everything

is all right so far, the responsible HA will finally receive the Registration Request message either forwarded by the Foreign Agent or sent directly by the MN. The Home Agent answers with the Registration Response message to the respective sender either including the confirmation of a successful registration or the respective error code in the Code field as illustrated in Figure 2.3-6.

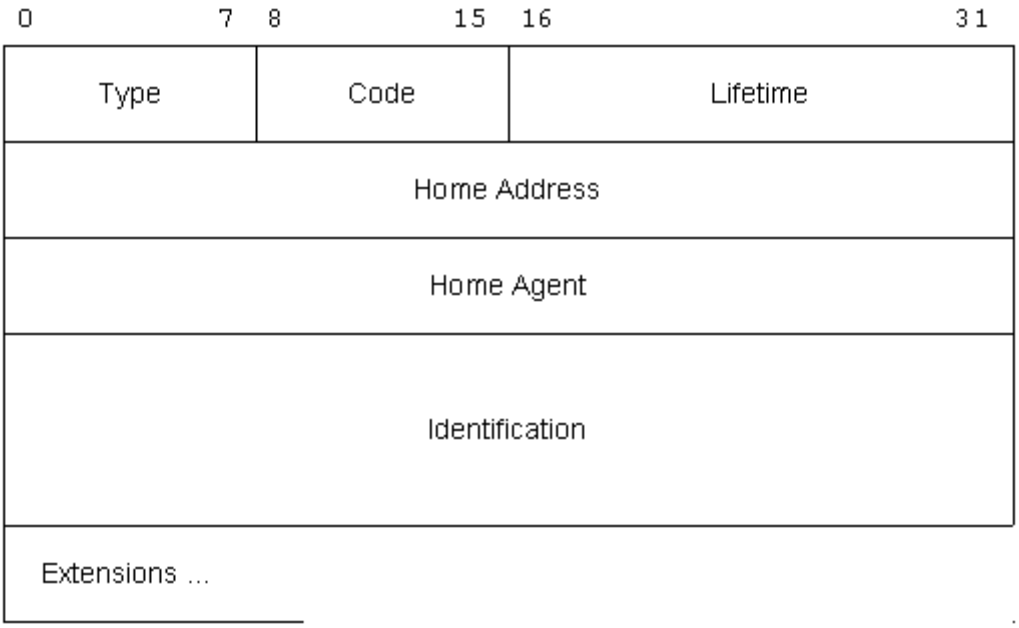

Figure 2.3-6 Mobile IP Registration Reply message

The lifetime field has to be ignored if the Code field indicates an error by being not zero. Otherwise it contains the time period till the confirmed registration expires. A value of zero confirms deregistration. The Home Address field carries the original IP of the MN and enables the FA to forward this reply to the right mobile node. The Home Agent field carries the address of the sending HA. The Identification serves the same purpose as described in Figure 2.3-5.

We should pay more attentions to one point in the registration control traffic defined by Mobile IP standard. In both registration messages (see Figure 2.3-5 and Figure 2.3-6) there is a variable length field called Extension. Every extension according to the format defined by the Mobile IP standard can be used here. For some cases however, Mobile IP prescribes the usage of specific extensions, e.g. every Registration Request and Reply has to carry at least one extension, which is mandatory. This type of extensions deals with the security or to be more precise with the authentication of the mobile participants. Obviously, every agent has to provide some (incomplete) services for unknown hosts acting like mobile nodes. Their identities have to be proven to avoid several types of security problems, which could occur otherwise. The IP-address as the only identification criterion is not enough since it does not provide any protection against selective manipulation (*IP-spoofing*), which makes an intrusion of unauthorized hosts in home and also in visited networks possible. Because of that, Mobile IP defines the *Authentication Extensions* for every reasonable pair of agent and node: *Mobile-Home Authentication Extension* to provide authentication of an MN to the responsible HA, *Mobile-Foreign Authentication Extension* to provide authentication of an MN to the available FA if possible and *Foreign-Home Authentication Extension* to provide

authentication of one agent to the other. The security association between two collaborating mobility-supporting networks needed for verification reasons has to be installed before. This association is called *Mobile Security Association (MSA)*. The first type of extension, the Mobile Home Authentication, must be included in *every* registration request/reply since every Home Agent has to share security associations with all of its mobile nodes. The other extensions may be included if a correspondent MSA exists between the visited and the home network.

All the security extensions have the same format and transport a security parameter index value (*SPI*) pointing into the host's own database and the authenticator value computed by some algorithm parameterized by this SPI (→[RFC2401]). The authenticator is a cryptographic hash-value built as a checksum over the UDP-payload and all prior extensions and therefore securing the data in the registration messages from being manipulated. The default used algorithm is the *keyed MD5* (→[RFC1828]) in "prefix+suffix" mode (see [Beth et al. 98], [Geiselmann 97]).

The request/reply procedure is repeated for every handover i.e. for every case where an MN relying on its move detection mechanisms assumes that it has been moved. In this way it is guaranteed that a responsible HA always knows the exact location of all of his MNs. The HA maintains a list of mappings of the absent MNs' Home Addresses to their actual locations, i.e. CoAs. According to the requirements of a MN claimed in its last Registration Request, the HA can now participate in the data traffic. It intercepts every packet destined explicitly to this MN; in addition the local broadcast packets if required and re-sends them to the listed corresponding CoA using the *IPIP-Encapsulation* (→[RFC2003]) shown in Figure 2.3-7.

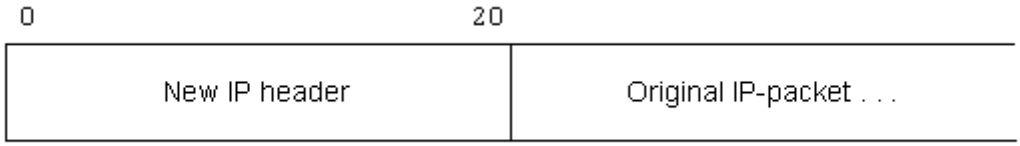

Figure 2.3-7 The format of IPIP encapsulation message

The new IP-header describes the tunnel and is defined to be "From: MN, To: CoA". The encapsulated packet is the original packet with the IP-header having the form "From: some sender, To: MN or broadcast address".

The interception of the packets is based on a mechanism called *ProxyARP*. ARP, *Address Resolution Protocol*, is the interface of every IP layer to the respective Link Layer. Defined within the IP definition, it is responsible for the mapping of an appropriate local IP address to a physical address of the local network adapter, typically called *MAC-Address* (after *Medium Access Control* layer) in order to enable the IP-to-MAC-address resolution and local packet delivery by Link Layer means. On broadcast networks, the ARP is implemented as a question/answer system: the ARP sub-layer being ordered to deliver packets, submits a broadcast packet asking to which network adapter (i.e. MAC address) this IP address is currently assigned. The responsible host receives the broadcast message and, having compared the sent IP-address with the own one, it answers with a confirmation packet to the sender of the broadcast message saying that this IP-address is assigned to its MAC-address. The answer is cached in the local *ARP table* of the asking host for some defined period of time. Thus, the IP layer can now instruct the Link Layer to deliver the packet to the obtained MAC address.

ProxyARP is a feature using the same mechanism for some more unusual and particular purposes. It inserts a so-called *public permanent* entry in the ARP-table of some host. Public means that the host publishes this entry on the local link on requests, i.e. the host doesn't use this information just for itself. Permanent means that the entry does not expire and has to be added and deleted "manually", i.e. the IP Layer itself never deletes (nor creates) such entries. Once being supplied with a new ProxyARP-entry by some authority, as e.g. the Home Agent software running on the host itself in our case, the host begins to answer the ARP-requests for some listed IP with its own MAC address. Therefore, this host receives the packets destined to this IP from now on. Normally, such a received packet would be dropped immediately by the host's IP-layer since it is not really destined for the host's pre-configured IP. However, having activated the *forwarding*-feature of the IP-layer, i.e. the ability to re-send the foreign packets, such a packet is routed according to its receiver entry. So, in our case, we have to provide a routing table entry for the absent MN, which leads through the logical IPIP device. Being encapsulated, the packet is sent as a packet from us to some not-local location, i.e. through the local gateway and further, by common IP-routing means.

Summarizing we see that each packet coming from an exterior CN and being intercepted by the HA is tunneled to the CoA. Being decapsulated there, it finally arrives at the MN's new location. Conversely, the MN answers directly to the CN by the standard IP means, as shown in Figure 2.3-2. That is called *triangular routing.*

Finally, we see how the basic problems we were talking about in Chapter 2.2 have been solved. In particular, we've got to know the mechanisms, which control and adjust the operational modes and enable the data traffic. We've also seen how the traffic flows afterwards. Evidently, Mobile IP is a well-defined standard, which was proven to be working in practice. But is Mobile IP the perfect solution?

2.4 Weaknesses of standard Mobile IP

In this chapter we want to talk about points in the Mobile IP definition where improvements seem possible. We do not want to try to propose solutions immediately but to look for ineffective and error-prone parts of the definition.

Analyzing the data traffic, one can easily see that the introduced triangular routing is pretty expensive. Obviously, the way from the CN to the MN is significantly more complex than the opposite direction. The problem is however, that this way has to be taken for every packet since we can't assume that an appropriate CN would learn the new location by some means. Intercepting and tunneling each packet increases the load at the Home Agent, in particular if we suppose that we have to deal with several hundred mobile participants. Apart from that, we should not forget that in addition each HA has to manage all the control traffic in order to avoid losing contact with one of its MNs. Furthermore, the encapsulation increases the overall amount of traffic by adding an additional IP-header to each sent packet, which are at least 20 bytes per packet for the common IPIP-encapsulation (see Figure 2.3-7). We should think about it since we talk about mobile links with narrower bandwidth here. Even optional possibilities presented in the Mobile IP standard like Van Jacobson header compression or Minimal Encapsulation cannot avoid increasing the load while encapsulating packets. Overall, we could say that this triangular routing is a kind of *sub-optimal routing*.

Another weak point of every bigger Mobile IP system is the *system configuration* and the *accounting*. Any bigger Mobile IP system normally consists of several FAs on one side, trying to distribute the traffic caused by the registering/registered foreign mobile nodes. On the other side, there could be several HAs each responsible for its respective subnet. First of all, each administrator of a bigger enterprise network will want to have a possibility to control the host access of foreign mobiles. That can have a lot of reasons. E.g. it could be necessary to deny access of some MNs. Another likely example could be the wish to log the amount of produced traffic per registered MN in order to be able to gather billing information or just to have an overview about the network. Even if such accounting mechanisms could be somehow offered by some specific implementation of Mobile IP there is no defined structure among the Mobile IP components. So, the network administrator will have to separately provide e.g. the access lists to each FA in his network. The same is true for the HAs. More generally spoken, there is no possibility to configure a Mobile IP network as a consistent system. A detailed administrational work is needed to configure each agent as a separate host and to achieve the wanted overall result at the end.

A problem rather concerning the control traffic is the way in which a handover has to be processed due to the Mobile IP rules. Let us remain at the same example scenario, which involves a bigger enterprise network with several Foreign Agents. We can easily imagine that a visiting Mobile Node frequently changes the FAs within this visited network. On the contrary, an inter-network handover, i.e. a handover to an exterior FA, should be less likely since it would mean that the participant really moves away from his actual geographical location. Such an inner-network FA-change is what we want to call a *local handover*. What will happen in this situation? Mobile IP doesn't distinguish between the two handover cases, so the same traffic occurs in both situations always incorporating the responsible Home Agent. Each time when an MN changes its responsible agent within an area of perhaps just hundred meters a registration request is sent to the MN's home network. Evidently, this solution is not optimal since the MN has already received a confirmation of his HA for some fixed period of time (see Chapter 2.3, Figure 2.3-5 and Figure 2.3-6). As long as this confirmation is still valid, it would be theoretically possible to just rearrange the responsibilities within the local network what would reduce the control traffic and the load at the Home Agent.

A more factual example, which confuses some existing applications, is the acquired *asymmetry of the TTL-values* in the sent IP packets. TTL-value (*time-to-live*) is a field in the IP header of each packet indicating the number of routers, which will forward that packet before it will be dropped. This way the packets never travel around infinitely. Each router decreases the TTL-value by one and drops the packet, if the value becomes zero. With this simple method we could e.g. mark packets as "local-only" by setting the TTL value to "1" since our own router wouldn't forward any of them to its exterior interface. Since the MN responds directly to the CN without the detour over the home network, the values of the TTL can differ considerably. This is especially true, if we notice that the tunnel between the HA and the CoA never costs more than one unit since the encapsulated packet is not modified by the routers and the outer packet is of no importance for the destination host, the MN. In the worst case, it is possible that CN's packets reach the MN without any problems while MN's packets never ever reach the CN. E.g. when MN is sending packets to its home network, the TTL-value may be so low that the packets expire before reaching their destination.

Finally, we should discuss another highly important topic in this scope: *security*. Transferring data to and from a great number of mobile participants through unknown foreign networks, maintaining connections with unknown unauthorized hosts and also intercepting packets raises a lot of security-related questions. We should take into consideration that the introduced authentication methods (see Chapter 2.3) are not meant to provide any data security but to authenticate the registering MN at its Home Agent. Even this authentication is not sufficient. If no security association exists between the HA and the actual FA, the FA is not able to check the authentication and has to trust an unknown pair of hosts which are going to use its services. On the other side, the HA can't authenticate this FA since it can't verify its origin. Understanding that the Mobile IP standard just tries to add mobility to the IPv4 and not to extend or improve the protocol itself, we should nevertheless think about the theoretical possibility of much better security in this context. Both the registration information and the tunneled data-packets are clear-text messages and therefore readable for all the local neighbors and trespassed routers. The data-packets are even modifiable. Noticing that the registration messages never occur in common IP-mode and that Mobile IP's triangular routing typically involves more routers in the packet delivery compared to the common IP, we could even talk about a certain impairment of the security by Mobile IP. E.g. one could trace the movements of mobile nodes by reading the registration traffic by some means or even participate at their data traffic by modifying the tunneled packets.

Another kind of security measure, which has become very popular, especially in the years after the release of the Mobile IP standard, is the *packet filtering*. Packet filter enable the network administrator to control the correctness of the used IP-addresses in- and outside of his own network by refusing all the packets which have improper sender/receiver values on the network entry point. E.g. a packet filter would refuse each packet coming from outside of the local network with the sender having some inner IP-address. Those measures prevent a lot of security problems (see also RFC2267) caused e.g. by IP-spoofing or wrong configuration and can help to reduce the amount of traffic on the LAN to Internet link. Since this mechanism is pretty simple and fast (there are a lot of hardware solutions for this), it's becoming increasingly widespread. Hardly any network today is completely open or unprotected. A more complex but also more mighty solution is a *firewall* enabling the administrator to control the passing packets on almost every layer (i.e. particularly on the transport layer in the case of TCP/IP). In this way a firewall can explicitly reject some services - or even more strictly - explicitly accept some services simultaneously rejecting all the others. Due to the nature of its definition, Mobile IP doesn't work with any of these two measures since it relies on the assumption that IP unicast datagrams are routed based on the destination address in the datagram header (and not e.g. by source address). Using one of the mentioned mechanisms we break this assumption and make a lot of Mobile IP functions unworkable. So, the only possible firewall solution in this scope would be a loose firewall, which lets pass all the packets from and to FAs. This solution causes another problem. In this case we have to replicate the firewall functionality at every available FA since our agents would be left unprotected otherwise. This measure distributes the security over the whole network. For this reason it obliterates the main advantage of the firewall - the centralized security.

2.5 Improvements to RFC2002

Motivated by the multitude of weaknesses presented in the last chapter, a lot of improvements have been recently proposed for Mobile IP. Due to the complexity and diversity of the existent Mobile IP standard extensions it will not be possible to discuss each system in detail within this scope. Instead, we want to shortly present the respective propositions and to illustrate their main motivation and objectives.

As already explained, the mobile node's IP is the only possibility to globally identify this host in the Internet. As a matter of fact, each change of this IP has to be treated as the change of the host, which foils every intention for dynamic IP assignment. Almost the same problem occurs while identifying the network – we have to work with the network masks and the IP addresses of the agents to recognize a handover although it is possible that an enterprise distributed over several locations uses some different network addresses. This complicates building conceivable Mobile IP infrastructures and limits the possibilities for the host identification and the address assignment to the basic IPv4 methods, which cannot always be used in the modern networks. Motivated by these considerations, global host and network identification methods by cryptology means have been introduced. Either based on the global network of the Authentication-Authorization-Accounting servers (*AAA* →[Perk&Calh 99] server systems, *DIAMETER* →[Calhoun 98], *RADIUS* →[RFC2138]) or being configured by other means, these systems use the Mobile IP Extension mechanism to add a so-called *Network Access Identifier* (NAI) →[RFC2486] to the registration and advertisement messages. This NAI is exactly the additional identification possibility, which is missing when using the IP-address itself. So, it becomes possible to dynamically assign an IP-address to a mobile node. At the same time one obtains a reliable network identification method.

Other extensions of the standard try to improve the Mobile IP routing. That is done by defining some better approaches for the packet forwarding and more generally for the MN communication →[Perk&John 99]. Packet forwarding after a handover is the ability to re-send the packets from the predecessor to the actually responsible foreign agent in order to minimize the packet loss. MN communication concerns the potential correspondent nodes. By adding so-called *binding caches* to the CNs these could learn the actual location of the respective mobile node. Knowing the new location the CNs would be able to send the packets directly to the MN avoiding the sub-optimal triangular routing. To achieve this, several new control messages are defined which are sent by the HA to each CN which is still sending packets to the home (or the previously visited) network when the MN is already registered at some other location. Such a message is called a *binding update message* (BUM). Apparently, each CN has to understand the new control messages in order to support this optimization. If so, it answers with a *binding acknowledge message* (BAM). The realization of this idea obliges to make changes to every potential CN, i.e. in principle to every host, which is of course somewhat unreal. But installing such binding caches at least in the networks already using Mobile IP would avoid sending packets from the foreign to the home network in cases where a foreign mobile node is communicating with a CN located in this foreign network.

2.6 Reverse Tunneling (RFC2344)

The most prominent and also standardized representative in the big zoo of Mobile IP improvements is beyond doubt the so-called *Reverse Tunneling* defined in [RFC2344]. The improvements, which were presented in Chapter 2.5, either try to add new functionality and possibilities to the standard or to optimize the existing features. The Reverse Tunneling-approach shows a way to make a Mobile IP system work in some very popular environment: it enables the usage with the most common packet filters and simple firewalls.

That is achieved by a relatively simple method. Almost in the same manner, in which the Home Agent tunnels the packets to the new location of an MN, represented by the CoA (*forward tunnel*), the answers of the MN are tunneled back to the Home Agent (*reverse tunnel*) by the Foreign Agent - or by the MN itself in case where no FA is available. This simple measure illustrated in Figure 2.6-1 is an effective work-around against at least two problems, which have been mentioned before. First of all, since the address of the real sender, the MN, is hidden within the tunneled packet, a usual packet filter would let it pass. A packet from FA to HA sent from the visited network is topologically correct and causes no problems. Only the transport protocol ID (usually TCP or UDP) within the original IP header changes to one of the possible encapsulation transport protocol IDs (usually IPIP →[RFC2003]). Secondly, a tunnel always represents exactly one hop from the point of view of the encapsulated packet. Thus, the discrepancy between the TTL-values on the way from and to the MN, which can cause packet loss on the way from MN to his home network is remedied. The TTL is decreased by exactly one hop for each direction.

Using the built-in extensibility of Mobile IP like e.g. reserved bits in the control messages and the presented Extension mechanism enabled the integration of this additional feature into the existing Mobile IP standard.

By adding an additional bit to the flag-field of the Agent Advertisement message, [RFC2344] defines a possibility to announce the support of new features. If a Mobile Node understands this new flag, it can now demand its usage in its Registration Request message. By using a new Mobile IP extension, the Mobile Node can also demand the usage of a so-called *Encapsulating Delivery Style extension*. This extension may only be used between the MN and the FA, i.e. each FA has to consume this extension and must not relay it to the HA. If this extension is added then the data from FA to MN and vice versa is transported in an additional tunnel. That is called *encapsulating delivery style*. Otherwise, the packets are sent directly. That is called *direct delivery style*. This additional tunneling is necessary in order to be able to optimize the data traffic between the MN in the foreign network and the other hosts in the same network. Without any measures, those packets would be tunneled back to the HA, which would then send them back to the foreign network by common IP routing means. E.g. a MN could want to communicate with a local printer in the foreign network. This kind of routing would be almost unacceptable. That is why the *selective reverse tunneling* has been defined. If a Mobile Node which has demanded an encapsulation delivery style by adding the extension sends a not encapsulated packet to the FA, then the FA must not tunnel it back to the HA. Instead, it decapsulates and re-sends this packet using the standard IP routing mechanism. In this fashion each MN can influence the encapsulation behavior of its responsible FA: the packets meant for the local network are sent directly to the destination hosts, the others are sent through the tunnel as originally intended.

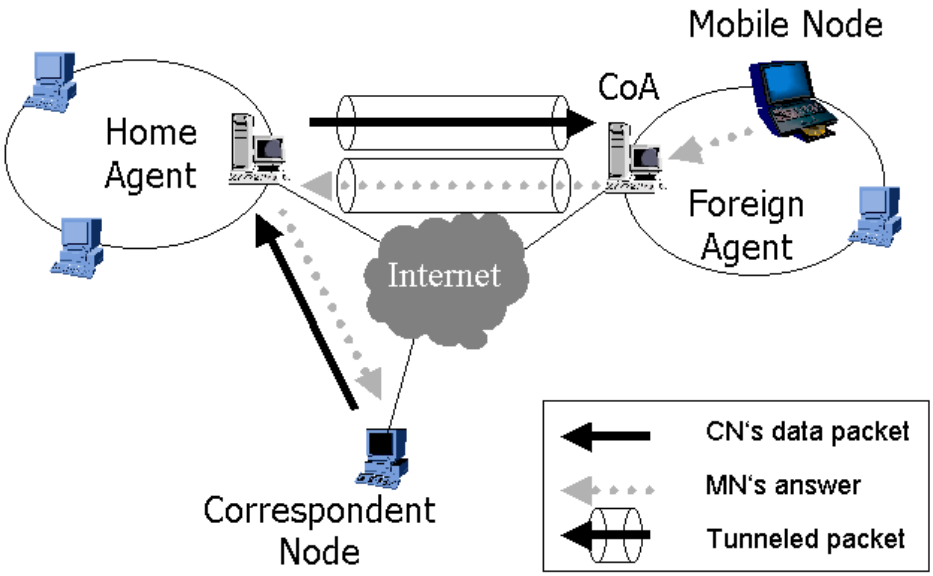

Figure 2.6-1 Data traffic according to Mobile IP with Reverse Tunneling

Hence, this quite simple but powerful improvement has to be explicitly supported by the MN; otherwise it is not used. However, the MNs coming from some network, which is not using a packet filter, would be probably unprepared. But we can hope that in the modern Internet such networks are rare and that being an Internet Standard, the [RFC2344] will spread quickly.

So, the main disadvantage of this improvement is the even worse routing behavior than in Mobile IP. Now, every packet between the two involved networks is transported in a tunnel increasing the load at the FA and the protocol overhead from FA to HA compared to the triangular routing. Additionally analyzing the behavior of Reverse Tunneling in combination with routing optimization, we notice a paradox sometimes referred to as *reverse triangular routing*. Using routing optimization the exterior CNs send the packets in a tunnel directly to the visited network. MN's answers are then tunneled back to the HA which finally re-sends them to the CN.

2.7 Other improvements to Mobile IP

Several complete solutions have been issued giving a structure to the internal components and trying to solve the main presented problems occurring mainly in the complex Mobile IP networks. The main representatives are *Cellular IP* →[Valko et al. 99] primarily improving the micro-mobility, i.e. the handovers within the network, and introducing central configuration for the involved inner-network components and *HAWAII* →[Ramj et al. 99] also aiming the improvement of the micro-mobility but apart from that introducing Quality of Services-support and increasing the reliability of the system. For more detailed descriptions we would like to refer to →[FATIMA].

However, the problem of both solutions is the obligatory changes to the MNs, which cause an incompatibility with conventional Mobile IP components.

Some other similar Mobile IP improvements, which define a complete system enabling better micro-mobility, usage of private IP-addresses, central system configuration and more features are described in [FATIMA]. Usually introducing a hierarchy on the internal components and one central element, these systems typically use one system-wide CoA and hide both the real inner structure and the control traffic in the network to outside as far as possible. Thus, the gateway is normally the only part of the system communicating with the external hosts. The essential changes in the control traffic sometimes influence the mobile nodes breaking the compatibility but usually achieving better theoretical results in this manner. Almost every such system makes great use of the Mobile IP extension mechanisms, which preserves the compatibility but simultaneously loses any effect if the respective extension is not supported by the partner network.

A whole family of extensions and improvements deals with the security in Mobile IP. *Secure Scaleable Authentication* enables the usage of certificates and of both symmetric and asymmetric cryptography in the Mobile IP authentication extension making the usage of pre-shared keys obsolete, which results in less administrational work. Unfortunately the format of the standard authentication extension has to be changed which causes an incompatibility with the Mobile IP standard. *Rapid Authentication* decreases the effort needed for the authentication every time when a MN triggers a local handover. For this purpose it defines a new extension, which simplifies a re-authentication within the same network. The missing accounting and authorization support, i.e. resource usage depending on the local policy and the current authentication, are meant to be solved by *DIAMETER extension for Mobile IP* →[Calhoun 98]. This extension integrates the DIAMETER-protocol in a Mobile IP system enabling automatic key distribution and thus a MN-FA-authentication and FA-authorization for offering certain services. To secure the Mobile IP data traffic two extensions have been defined. Added to the control messages they support *IPSec* →[RFC2401] *within Mobile IP*. With its help it becomes possible to use IPSec's ESP-tunnels →[RFC2406], which provide data message encryption and thus integrity, authenticity and confidentiality of the transported data.

Please refer to [FATIMA], [Schiller 99] and [MIP Web] for further information and references.

3 New proposal: The FATIMA system

3.1 Overview of the existing draft

As described in the chapters above, the standard extensions and improvements, which exist at the moment, either cannot solve all known problems or they do this at the cost of compatibility. One can easily see that a set of control message format extensions isn't sufficient for a general solution since a lot of problems can be reduced to the missing infrastructure. The management of the infrastructure and the establishment of the internal connections are exactly the point where the most Mobile IP-like systems shortly presented in Chapter 2.7 require some additional work to be done at the side of the mobile node simultaneously causing the incompatibility.

At the ITM, University of Karlsruhe, Germany a new draft has been released promising theoretical possibility of a solution for the most problems and keeping the interoperability with [RFC2002] networks by leaving the Mobile Node untouched. Additionally, the transparency to the exterior hosts is guaranteed. Every change of the Mobile IP specification only deals with the system's inner nodes and no changes in external FAs or HAs are needed. Probably the most interesting point in the new approach is the conceptual integration of a firewall from the beginning on. In following the draft will be shortly described in its status quo. For a complete documentation please refer to [FATIMA].

[FATIMA] (*Firewall Aware Transparent Internet Mobility Architecture*) defines a tree-structure in the local network interconnecting the inner nodes. The edges of this tree are FATIMA-connections as defined in the same draft and have nothing to do with the real existing Level 2-links. It is intended to cryptographically encode the inner connections using e.g. the ESP-tunnels →[RFC2406]. The tree nodes are the FATIMA-components. There are several kinds of nodes, which will be discussed in detail later. The *Exterior Mobile IP Gateway* is the only FATIMA-node maintaining access to the exterior hosts. There can be only one such gateway per FATIMA-system always situated in the root of the tree. Two other important FATIMA-nodes are the *FA-Proxy* (FAP) and *HA-Proxy* (HAP) replacing their respective RFC2002-counterparts and roughly executing the same tasks. There has to be at least one of each proxy type in the FATIMA-tree. The complete FATIMA-tree is called *Virtual Mobile IP Network* (VMN). The tree is built automatically in the initialization phase using the *VMN Management Protocol,* which is defined as a framework within the draft →[FATIMA] at this moment. The same protocol is intended to maintain the tree and e.g. to integrate new components, etc. An example of the smallest FATIMA-tree is illustrated in the Figure 3.2-1 and a more sophisticated one in the Figure 3.2-2. Other inner nodes are supposed to separate the tree into logical sub-trees, i.e. FATIMA-sub-networks (*Interior Mobile Gateway, IGW*), or to route the FATIMA-related packets within complex trees (*Routing Agents, RA*). These types of agents never deal directly with the Mobile Nodes. They only forward the received packets within the VMN along the optimal paths and enable greater trees.

3.2 Control traffic: registration, de-registration

In general, the control messages are always sent along the paths in the FATIMA-tree. For this mean, every FATIMA-node participates in the control traffic, recording from which address the message came from for every MN in its database. These entries are used in order to provide the address translation since the control messages are often forwarded within the FATIMA-tree from one FATIMA-node to the next responsible node. Additionally, FATIMA-nodes are capable of distinguishing between *temporary* and *permanent* (or *active*) entries for the MNs. Temporary entries are used for the MNs, which are currently registering and permanent for the registered MNs. The temporary entries are deleted automatically after some reasonable period of time.

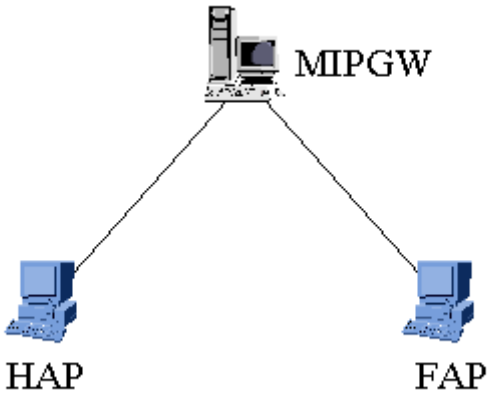

Figure 3.2-1 The smallest FATIMA tree

To demonstrate the main idea of a FATIMA-system we could take a look at a simple FATIMA-network consisting of one MIPGW, one HAP and one FAP. Being configured by the hereinbefore mentioned protocol, these nodes build a simple tree structure as shown in Figure 3.2-1. A foreign node, which has arrived at this network, hears the advertisements periodically sent by the FAP. However, these advertisements carry the CoA address of the central FATIMA-gateway and not of the foreign agent itself. Thus, the MN sends a [RFC2002] conformable registration request to the sender of the advertisement with its pre-configured Home Agent and the obtained central CoA. The message arrives at FAP, which creates a temporary entry for this MN and forwards the message over the FATIMA-tree connections to its parent-node in the tree, which is the central gateway in our case. The gateway checks the included MN-FA authentication (if any), creates a temporary entry for this MN recording the sender of the registration request (i.e. the FAP in our case) and forwards this registration request to the included HA-address if possible. All the possible nodes on the way from FA to MIPGW would proceed in the same manner, always recording the sender for this MN.

After the common [RFC2002] procedure the Registration Reply message finally arrives at the central gateway (since the HA always answers to the sender). The gateway checks the reply, then its temporary entries and having found the MN, changes the entry to a permanent one if the response was positive or deletes the entry otherwise. In any case it forwards the message to the appropriate FAP found in the temporary entry. It proceeds in the same way deleting or changing the entry to a permanent one. It forwards the answer to the respective MN so it gets the result of his registration try.

The own Mobile Nodes are pre-configured with the address of the MIPGW, which acts as a V*irtual Home Agent* for all absent MNs. If a MIPGW receives a Registration Request of one of its own Mobile Nodes coming from the exterior link, it checks the included AEs, creates a temporary entry for this MN and forwards the request to the responsible HAP (the GW could decide this e.g. due to the subnet division of the IP network, in this context see also Chapter 6.4). Before doing so, it changes the HA-field in the registration request (see Chapter 2.3, Figure 2.3-5) to the address of this HAP. The HAP gets the request from his FATIMA-tree parent and activates ProxyARP at the respective physical link, from now on intercepting all the messages destined to the MN. The HAP creates a permanent entry for the MN and generates the RFC2002-conform Registration Reply as described in Figure 2.3-6. Hereafter, it sends the message along the tree path. The reply message arrives at the MIPGW, being routed within the FATIMA-tree. The MIPGW changes its MN-database according to the reply. Having changed the HA-field (see Figure 2.3-6) in the reply message to its own address, it forwards the reply to the FA from which the request originally came due to the temporary entry. By this mean it lets its MN know that MNs current location has been registered.

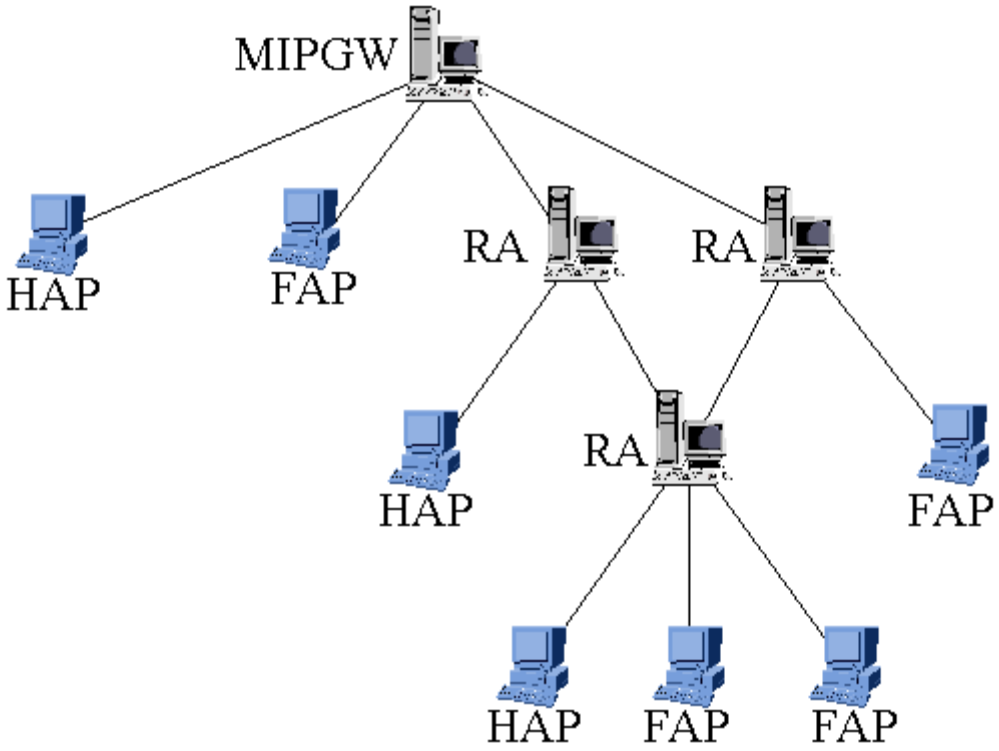

Figure 3.2-2 An example for a possible FATIMA network

Returning home an MN sends a de-registration message (as customary in Mobile IP, see Chapter 2.3) to its HAP. The HAP just forwards the message to the central gateway, which checks the MN-HA-AE and deletes the registered MN from it registration database if the authentication is right. Then the MIPGW generates the Registration Reply and replies with it to the HAP, which now deactivates the ProxyARP, deletes the permanent entry from its registration database and gives the message to the MN.

3.3 Data traffic: packet routing

The usage of the FATIMA system promises great routing improvement. Having received a data packet each FATIMA-node checks its registration tables for permanent entries and sends the packets destined to the registered MNs over the appropriate link. E.g. the MIPGW wouldn't ever send the messages, which are destined to the local network back to the home network of the foreign MN, and the FAP wouldn't forward the messages from a registered MN to the MIPGW if these messages were destined to another registered MN. The same is consequently done by the Routing Agents and of course by the IGWs. That is what we would normally call "per-host routing". This usually causes performance leaks, however in this case it seems to be well usable since the number of hosts per agent is not supposed to become very high and - if that happens - can be decreased by adding new agents.

[FATIMA] supports Reverse Tunneling →[RFC2344]. I.e. having received a data packet destined to a CN in the home network of a MN, the central gateway tunnels the packet back to the MN's Home Agent if possible.

Usually, the data messages of the own absent Mobile Nodes don't arrive at our FATIMA network. However, if the reverse tunneling is used, these packets arrive at the MIPGW, which then re-sends them to the respective direction – to the CN outside or to the appropriate host in the own network using standard IP in both cases.

3.4 The agent types in FATIMA

In this chapter we will give a small summary of the main tasks of all the FATIMA-agents corresponding to the methods explained in the Chapters 3.1 - 3.3:

- Exterior Mobile IP Gateway (MIPGW):
    - provides an unique CoA advertised by all FAPs, i.e. MIPGW is the end of all Mobile IP related tunnels)
    - acts as a (virtual) HA whose address is given in the Registration Requests by the own MNs in foreign networks
    - does the whole work needed for authentication purposes (checks of HA-MN-AE, FA-MN-AE, HA-FA-AE)
    - acts as a main controlling instance of all Mobile IP activities in the own and the foreign networks taking track of the registered and known MNs. Provides control of own HAPs and FAPs by the mean of the VMN Protocol and registration traffic
    - is responsible for any routing optimization methods
    - avoids reverse triangular routing as described in Chapter 2.6. For this purpose this host is authorized to send topologically incorrect packets through the outer packet filter

- HAP:
    - is still responsible for ProxyARP, i.e. for packet intercepting after having received the forwarded Registration Request from the MIPGW
    - generates Registration Reply messages (except for de-registration)
    - forwards the locally sent data packets to the parent in the FATIMA-tree
    - is no more involved in authentication, Mobile IP tunneling and route optimization
- FAP:
    - advertises the MIPGW in his own advertisement broadcasts
    - forwards the Registration Request messages received from the visiting MNs to the FATIMA-tree parent
    - limits authentication work to AE-checking after local handovers
    - Replay-attack-protection by supporting a new extension
- RA:
    - comparable to a usual router in an IP network, each RA tries to send the received packets to the receiver over the shortest available and working path. Each RA can have several child-nodes and several parent nodes. The parent and child nodes can be given a priority.
- IGW:
    - has the complete RA-functionality
    - represents a "small" MIPGW and opens a sub-tree within the main FATIMA-tree, separating the subnet from the main VMN in the same way in which the MIPGW separates the exterior networks from its own net

3.5 Security in FATIMA

The security aspect can be divided into at least three more or less independent parts. The passive security can be achieved by firewalls, packet filters and other measures preventing the potentially insecure connections of ever being installed →[Schmeh 98]. The active security can be used at the same time enforcing all the connection channels by cryptographic means, as e.g. by securing the packet integrity and achieving data confidentiality →[Beth et al. 98]. And finally, explicit defense measures against some (due to the used protocol) possible attack types can be installed to prevent host crashes or long time non-accessibility like often caused by Denial-of-Service attacks. All three types of security can be used in a FATIMA system.

As already mentioned before, the [FATIMA]-draft explicitly supports firewalls. This is done by integrating the central FATIMA-gateway into the firewall structure, e.g. as a *Bastion Host* →[Wack&Car 94]. Since the gateway has been defined as the only host to support exterior connections within the FATIMA-system and all the hosts in the screened subnet are permitted to maintain connections to the backbone (Internet) this configuration doesn't cause any problems (see Figure 3.5-1) to the connectivity. Since the internal screening router (packet filter) is defined to let the packets from the own network to the screened subnet and vice versa, the defined communication with the other FATIMA agents is not touched either. Apart from FATIMA's central gateway the only type of involved hosts, which could ever try to communicate through the external screening router (see Figure 3.5-1) are the own Mobile Nodes visiting foreign networks. Due to the definition of [FATIMA], the complete control traffic will be addressed to the central gateway, which is fully accessible from outside. The *authorized firewall traversal* as described in [FATIMA] has been proposed for the data communication.

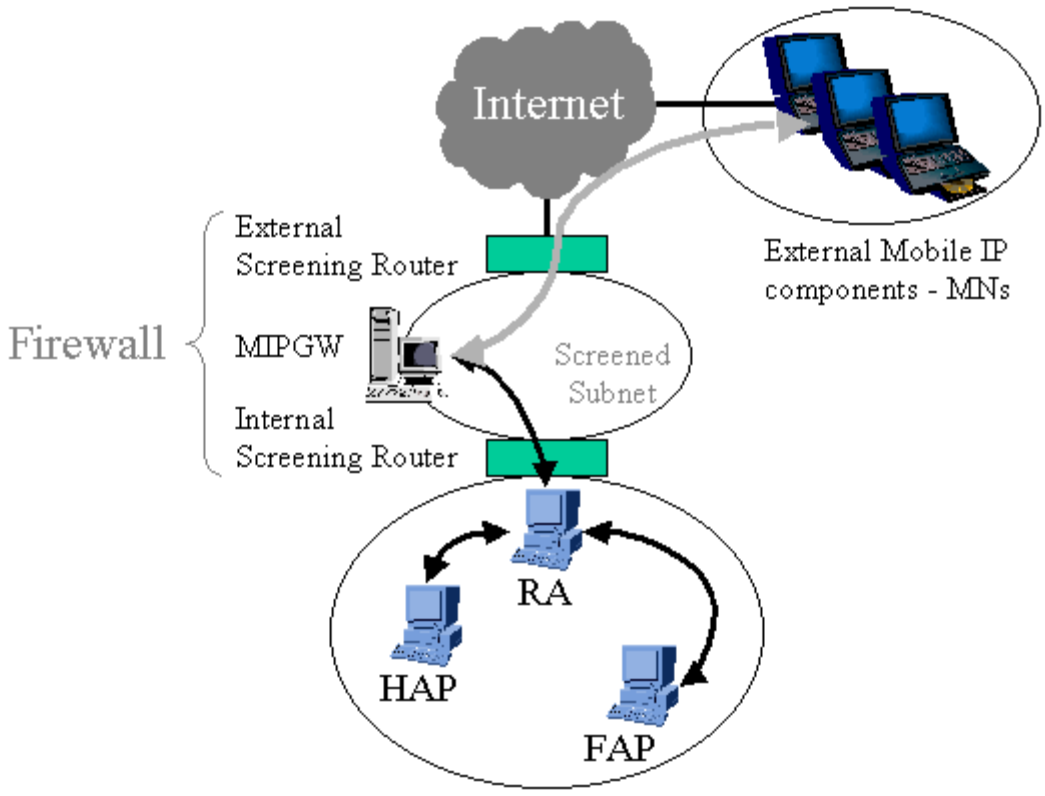

Figure 3.5-1 Integration of FATIMA in the firewall

An example for the firewall rules with this configuration is given in the Table 3.5-1. This example supposes the concerned network at 129.13.*.* and its Screened Subnet at 129.13.98.*. The traffic direction is always meant from the point of view of the protected network, i.e. for the external router from the point of view of the Screened Subnet and for the internal router from the point of view of the local network itself. To avoid the reverse tri-angular routing as described in Chapter 2.6 the external screening router (→Figure 3.5-1) could be configured to let out the topologically incorrect packets (→Table 3.5-1, rule 3) physically sent by the central gateway's network adapter. In this way, the MIPGW acting as FA for the foreign mobile nodes could forward their packets directly to the correspondent nodes, without the detour via the respective home agent, as it would be using the reverse tunneling option. This configuration does not cause any

security problems since the hosts in the screened subnet are separately supervised and no additional packets are let in into the network.

Outer screening router				**Rule N°**
	Incoming	From:	!129.13.*.*	**1**
		To: *	129.13.98.*	**2**
	Outgoing	From:	*.*.*.*	**3**
		To:	!129.13.*.*	**4**
Inner screening router				
	Incoming	From:	129.13.*.*	**5**
		To:	129.13.*.*	**6**
	Outgoing	From:	129.13.*.*, !129.13.98.*	**7**
		To:	129.13.98.*	**8**

Table 3.5-1 Firewall rules used in FATIMA

The active security will be discussed here separated into two parts, the external messages used for the communication with mobile nodes and the FATIMA internal traffic. The control messages for registration purposes are secured by AE due to the [RFC2002], which remains unchanged for compatibility reasons. However, the number of necessary Mobile Secure Associations (see Chapter 2.3) can be dramatically decreased using a FATIMA system, since only the FATIMA-gateway has to share the necessary MSAs and not each present agent as usual. E.g. two networks with FATIMA just need one MSA while two standard Mobile IP networks have to install an MSA between each HA and FA. Since FATIMA will support route optimization →[Perk&John 99], it is intended to secure route optimization messages as well in order to prevent an attacker on detouring the MN-traffic to itself. The data traffic with the MNs is not encrypted because this is supposed to be the responsibility of each MN.

All internal FATIMA-nodes are to be equipped with *X.509-certificates* →[Geiselmann 97] signed by a local CA. It is intended that with the help of these certificates each FATIMA-node will use AH or ESP-Tunnel →[RFC2401] to provide control message integrity.

Finally, under the presumption that the own network is secure, the central gateway and the FAPs are obviously the only possible points for different types of attacks, since these have to process packets from unknown hosts. From the point of view of a potential attacker reply, DoS and flood attacks are suitable in this case. Especially at the central gateway which is the only entry point to the network a lot of work should and can be done to prevent or at least to make these types of attacks harder. The advantage of FATIMA in this case is the administration of just one node, which, in addition, is part

of a firewall. A lot of ideas like packet filtering of involved packets (i.e. tunneled and registration packets), intelligent mobile node treatment strategies and even clustering have been proposed for these purposes, as mentioned in [FATIMA].

3.6 Fast Handoff Extension

[FATIMA] shows a Mobile IP based approach to achieve local FA-changes, so called fast handoffs, without interacting with the responsible home agent. This mechanism is realized with the help of the general Mobile IP extension mechanism presented in Chapter 2.3. Here, the idea will be presented. For detailed instructions, please refer to [FATIMA], Chapter 13, Fast Handoff Extension.

The motivation for such a mechanism is plausible. Since FATIMA-networks hide the local FA-changes to outside using one central CoA it would be a reasonable wish to re-register the MN locally without first contacting its HA. This can always be done if the registration lifetime affirmed by the HA for the past registration request hasn't yet expired at the moment when a new registration request is sent by the MN. The new registration request passes the central gateway and so the gateway could generate the corresponding positive registration request message itself, i.e. without relaying the message to the home agent. If the MN frequently changes its location within the FATIMA-tree, this would result in

- decreasing the amount of control traffic
- decreasing the load at the home agent
- decreasing the overall registration time from the point of view of the MN

In [FATIMA], the Fast Handoff Extension (FHE) can be used if the involved authorities, both the MN and the FATIMA central gateway like to use it. This extension in the common Mobile IP extension format with the extension number higher than 127 (→Chapter 2.3) is added by the active FAP to the last registration reply message. If the MN supports this extension, it can demand a fast handoff in its next registration. If it doesn't support the FHE, it can drop this extension without dropping the other included extensions, in accordance with its high type number.

The rest of the mechanism is pretty simple. Since everything is managed by the FATIMA-tree the MN doesn't even need to know the content of the received FHE. It's enough to save it and to add it in an unchanged form to the next Registration Request message. Because of this fact, [FATIMA] doesn't even dictate the content of the data-field of the extension. It can be freely chosen depending on the implementation. It probably has to hold at least the IP-address of the MN, the expiration time of the current registration and MN's authentication data. The Type field has to be set to an unused value higher than 127 and the Length field has to hold the length of the extension data. The position of the FHE in the extension sequence is more important and is explained in [FATIMA], Chapter 13.4.5, p224.

4 Basic ideas for FATIMA concept verification and evaluation

4.1 Real implementation vs. simulation

Since the main task was to evaluate the possible performance within mobile networks, the first question was to decide what is the best way to do it. Basically, there are two main strategies:

- Real implementation on some suitable free OS
- Simulation of the involved networks with the help of some simulation software

The first possibility includes the completion of the new draft where it is necessary and the OS-based implementation of at least each agent and feature defined mandatory by the draft. However, there is no need to implement standard Mobile IP agents, since free implementations exist for the most common free OS like e.g. [Linux] or [FreeBSD]. On the other hand, we have to supply the necessary number of computers, i.e. above all to buy and install the hardware and the OS. We have to build a network. The qualitative results will be probably more accurate in that case since we can't guarantee that the simulation software provides the real network environment with all details and unpredictable problems (we could even say that it surely doesn't). Additionally, a simulation as a software running on some computer gives the possibilities to exchange the data between the involved simulated nodes in a way, which is not available in the reality, making the results unreliable by doing so.

Summarizing, we could say that a real implementation would probably help a lot in finding the concept errors. We would obtain a reliable concept and more precise qualitative results. We would even gain a running software prototype, which could be used in a slightly modified form after the research. However, in the mobile environment it would be nearly impossible to make measurements on the mobile node movements. We can't really afford hundreds mobile participants moving around. Additionally, we would like them to move in some special way, imitating an enterprise network or perhaps a small one. Obtaining quantitative results by these means would take an extraordinary long period of time, which is, after all, the main reason against the real implementation at this step of draft development.

4.2 Possible approaches and final decision

So, we've come to the decision to simulate networks with mobile nodes gathering required samples. To do so, we could write an own program, use some commercial simulation software or try to find an alternative.

Obviously, writing the own software results in more work compared with the second solution. First of all, we have to provide the time control engine, i.e. to create a simulation core, which is an outstanding task. Secondly, commercial packets are usually supplied with plenty of functions and simulation models, which can be reused for your

own purposes saving time, which would otherwise be needed for the creation of the own models. Particularly such widespread communication models as a TCP/IP network are very likely to be available.

However, the commercial packets have a great disadvantage. This is the extraordinary high price of the packages and, to be more precise, especially the typically used payment models in this relatively small market. Usually, some time-limited licensing is used forcing the customer to pay a fee every new period of time in order to have the right to use the software. These fees are sometimes too high for educational authorities. Nevertheless, in the beginning we decided to use a commercial package. Unfortunately, both the production of this software and any support have suddenly been stopped by the software producer. All in all, this is probably always the same risk when using a software package provided by somebody else.

So, we had to find an alternative. The solution was a free simulation package. After some search, we finally found OMNeT++ at [OMNeT Web] which already had a user community and a structured documentation. A list of other free simulation packages and a comparison can be found in [OMNeT Man].

4.3 OMNeT++: discrete event simulation tool

OMNeT++ →[OMNeT Web] is a C++-based discrete simulator for modeling communication networks, multiprocessors and other distributed systems. It's open-source and free for non-profit use. OMNeT++ provides exchangeable interfaces, among other a graphical user interface as shown in Figure 4.3-1 which includes monitoring of the involved components, simulation progress display and even debugging aids →[OMNeT Man]

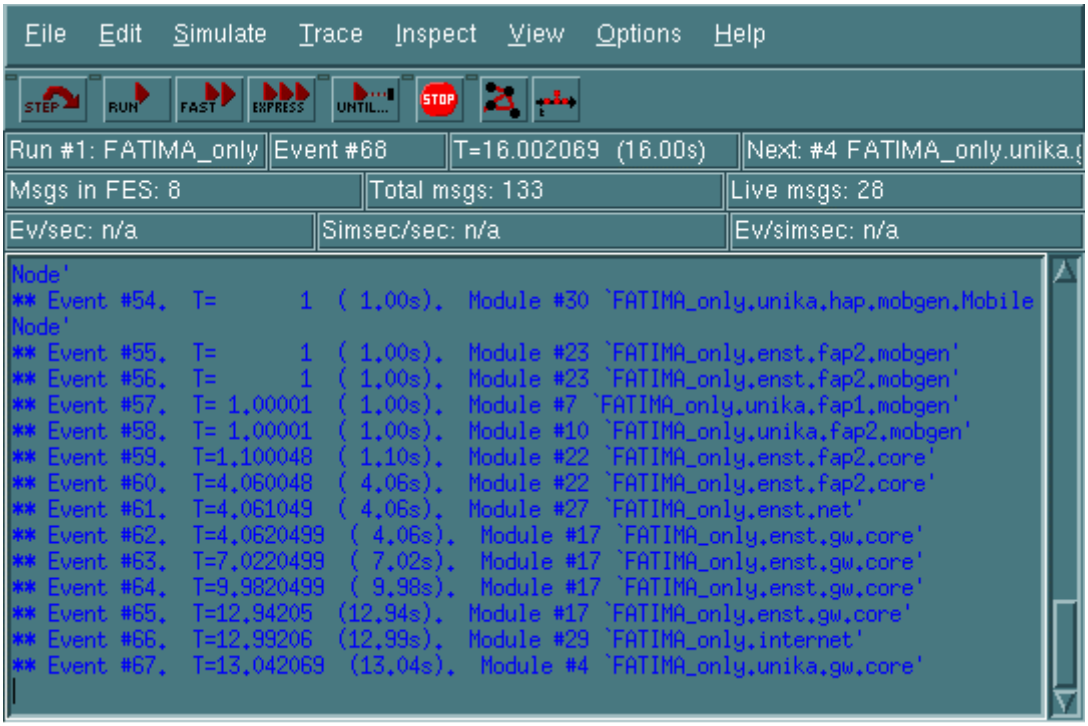

Figure 4.3-1 OMNeT++ GUI

An OMNeT++ model consists of so-called *modules* that communicate by exchanging *messages*. During the network layout, i.e. the existing modules and connections between them, can be defined by the mean of the integrated *Network Description Language (NED)*, the actual behavior of the modules is defined as C++ code using the OMNeT++ simulation library. In this way, the interface defined by NED is separated from the behavior (implementation) defined as C++ code. This provides reusability of module interfaces defined by the NED code. There are two kinds of modules, *simple modules,* which are the real active parts of the model and *compound modules,* which represent more complicated elements built out of several simple modules. Simple modules are executed quasi in parallel by the simulation core. Additionally, there is a possibility to create modules during the execution of the simulation. These modules are called *dynamic modules*. Independent of the module type, a message can be destined either directly to a certain module or travel through a given module gate. Messages can carry arbitrary data. Being sent through a gate a message follows the predefined path beginning at this gate due to the NED code. Of course, the connections can also be created during a simulation run.

For the implementation of the simple modules OMNeT++ offers an API as shown in Figure 4.3-2 including a simple module interface, a message interface and a rich simulation library providing support for essential functions, as a lot of routines for the simulation purposes as e.g. I/O-functions, statistics-classes for gathering the achieved results, etc. but also more general stuff like statistical distributions, random numbers generators and even container classes like queues, stacks, containers, etc.

Basically, a new simple module is created in two steps. First, its interface is defined in the topology description. Then, its corresponding class is created by sub-classing the offered API-interface (see Figure 4.3-2). The interface defines functions which have to be implemented in the new class (as far as enforceable in C++) in order to act as a "real" module in the view representation created automatically by OMNeT++. Apart from that, if the possibilities proposed by the basic message class aren't satisfactory to the programmer, it's also possible to sub-class the message class adding new functions, attributes, etc.

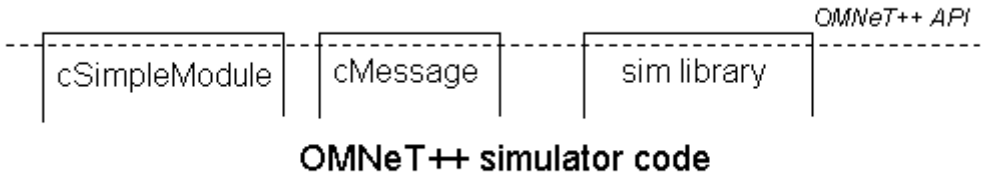

Figure 4.3-2 OMNeT++ advanced programmer interface

Finally, wanted results can be collected in the modules by calling routines of the statistics classes, transparently saving all given values to output vector and scalar files, depending on the type of gathered data. These pure text files designed to use a reasonable and simple format can be statistically analyzed afterwards by using suitable statistics packages.

In this manner, OMNeT++ represents a pure simulation engine, taking track of events and simulated time. It takes care of message sending guaranteeing correct message

delivery i.e. at the right place and to the right point of time. OMNeT++ also provides plenty of functions supporting a user (simulation programmer) in his work. However, OMNeT++ lacks support for all kinds of multicast messages, in particular for broadcast messages. The support of this feature is not planned for the near future. And as a pure simulation engine, OMNeT++ in its basic distribution does not know anything about TCP/IP[1].

[1] In the meantime, OMNeT++ has a CVS tree providing access to existing models, among others also for TCP/IP. Please refer to [OMNeT Web] for the current address of the tree and further information.

5 Conceptual design of the simulation

5.1 Overview

Having chosen the simulation system one has to think about the concept for the software to be built. The most important question here is to define exactly what we really want to be implemented. Naturally, this depends on definition of the given work. Basically, to fulfill the formulated task, it's an obvious duty to implement both Mobile IP agents according to the [RFC2002] and the agents according to the [FATIMA] system, which is not yet completely specified. Some common extensions to [RFC2002] as e.g. Reverse Tunneling (see Chapter 2.6) have to be implemented as well. Due to the nature of the chosen simulation engine (see Chapter 4.3) the scope has to be extended. Some additional implementational work has to be invested into the basic mechanisms, which are not natively supported by OMNeT++ but needed for Mobile IP. Evidently, we can't talk about Mobile IP without having a TCP/IP environment. Of course, we don't need to implement the whole TCP/IP suite, but at least the basic routing behavior and some ICMP commands have to be recognized by our system, since they are used by [RFC2002].

The analysis of the specifications and of the given task finally resulted in the following rough volume of work:

- **Network and Transport Layers:**
  TCP/IP-compatible basic routing, IP-addressing, basic protocol distinction, broadcast support and other functions necessary to provide support for Mobile IP. No transport layer functionality like congestion control, handshaking, connections, etc.

- **Mobile IP:**
  FA-, HA- and MN implementations with Agent Advertisements, control traffic exchange, one sort of data packets. Time control of registrations, only basic IPIP-encapsulation support, no flag support for optional features.

- **Reverse Tunneling:**
  This has to be supported in the agents and in the mobile node. Moreover, it should be switchable on/off by the user in the implementation of the simulation.

- **FATIMA:**
  Conception and implementation of the central gateway and FAP and HAP. Since the RAs just provide routing within the FATIMA networks, they are not indispensable to obtain the necessary answers. For the same reason and since they represent a local MIPGW, an IGW is not needed neither. The central gateway should be probably paid more attention. A possible subdivision into several independent modules should be proposed and supported.

Having completed this work, several different networks can be built with the help of the simulation engine in order to prove the workability of the concept and to collect statistical data on the behavior of the respective systems afterwards.

5.2 Necessary minimal network

In this chapter we will discuss the dilemma, which always arises when talking about simulations in general. On one hand, to obtain reliable results the simulation has to support a great number of possible scenarios. The greater this number, the better the simulation since it can provide a great amount of predictions making the overall result "more realistic". Typically, the subject of the simulation is an innovation, e.g. a new technology or mechanism to be researched. Hence, we always have to keep in mind that a simulation programmer doesn't necessarily know about all the possible scenarios. On the other hand, the complexity of the implementation usually directly depends on the number of the supported situations or with other words on the proximity to the reality. So, a programmer has to deal with the evaluation of the possible scenarios trying to find out and to implement the ones known to be indispensable to let out redundant situations, from the point of view of the given task.

In the case discussed here the obvious question is the number of simultaneously active networks, which are needed to simulate a FATIMA system and to obtain the requested results. Well, since we have to provide a comparative statement, we also have to support a standard Mobile IP system. Overall, the simulation program should be able to simulate the most common scenarios:

For the control traffic:

- Location recognition
- Registration of the own mobiles from a foreign net
- De-registration of the own mobiles returning home
- Registration of the foreign mobiles in the local network

Data traffic: the data exchange between

- An own MN outside of the own network and a CN in the own network
- An own MN outside of the own network and a CN in the visited network
- An own MN outside of the own network and a CN somewhere in the Internet
- A foreign MN in our network and CN in our network
- A foreign MN in our network and CN in its home network
- A foreign MN in our network and CN somewhere in the Internet

Analyzing the upper scheme one can see that we need at least two networks of each type in order to be able to emulate these situations and to obtain comparative results (i.e. we would have to implement four networks connected to the same backbone in order to simulate and evaluate MIP←→MIP, MIP←→ FATIMA and FATIMA←→FATIMA scenarios). However, for the upper scenarios only two networks will be really active at the same time, since we do not have situations involving three different networks, except for data exchange with a CN, which is neither in the own nor in the visited network. This case is negligible since it can be more easily simulated without a mobility supporting network using a standard internet host connected to the Internet backbone.

Using the NED language of OMNeT++, it is possible to easily build a new topology for a new simulation run involving pre-configured and implemented modules. In this manner, we have the possibility to implement and define a FATIMA system and a Mobile IP system and to build a network of e.g. two Mobile IP systems or a FATIMA – Mobile IP simulation. The top-view on the whole network is independent of the involved members as shown in Figure 5.2-1, where e.g. "`unika`" could be a FATIMA network and "`enst`" a standard Mobile IP network or visa versa.

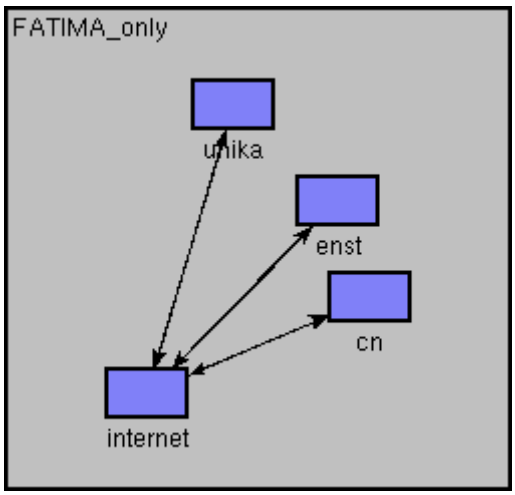

Figure 5.2-1 Networks, the Internet module and CN

Thanks to this favorable feature, it is not necessary to build a simulation program using four networks at the same time as mentioned before. Instead of this, we introduce a new instance representing the Internet backbone ("`internet`" in Figure 5.2-1), to which we will connect the two involved networks and a correspondent node ("`cn`"). This instance will just provide the basic routing functionality delivering a received packet to the respective network due to the packet address.

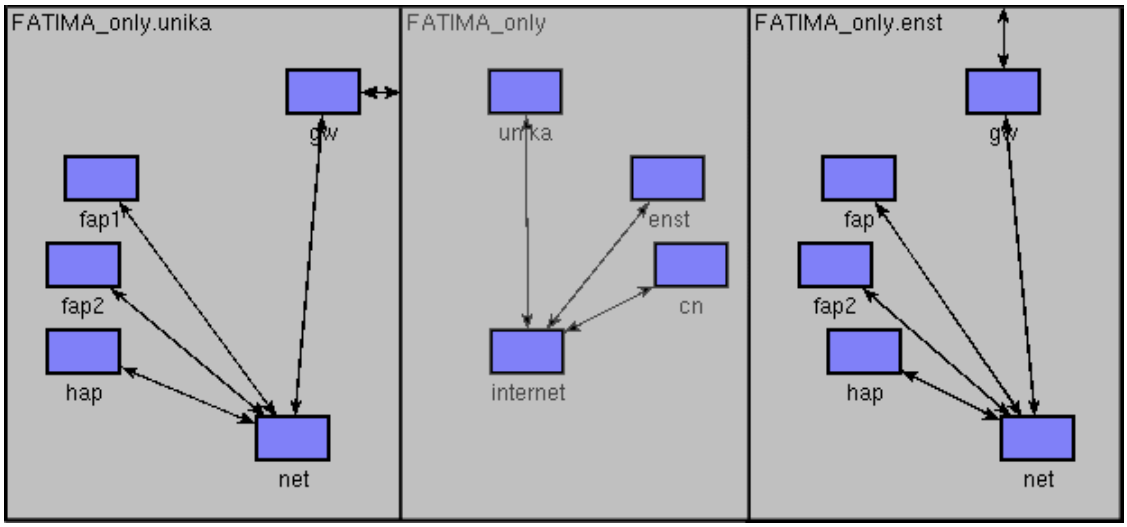

Figure 5.2-2 An example network simulating two FATIMA systems

In this manner we can obtain the results for two Mobile IP networks, the mixed scenario and the pure FATIMA/FATIMA connection by building a new simulation topology using OMNeT++'s NED language and recompiling the source code. The built

simulations can then be started successively gathering the requested results for the respective mode. This approach is explicitly proposed and supported by OMNeT++ as a direct consequence of the separation of module's interface and implementation as explained in Chapter 4.3 and [OMNeT Man].

Of course, the needed scenarios additionally depend on proposed protocol changes / extensions to be analyzed. E.g. for a Fast Handoff Extension (see Chapter 3.6), we need at least two foreign agent instances in the foreign network. In contrast, no necessity of a scenario with several home agent instances could be identified according to the available propositions. For that reason, the decision has been taken to support several FAPs / FAs and just one HAP / HA per network. Within the FATIMA system, no RAs and IGWs will be supported since these do not influence the performance results and just help to divide the FATIMA tree into organizational entities. An example for a complete simulation topology using two FATIMA-systems is shown in Figure 5.2-2.

In order to keep the implementation work in the given time range, the routing optimization and packet forwarding from the old FA to the new FA will not be used, since it would involve three networks.

5.3 Independency and potential code portability

At the moment of the conception of this work, it was not sure that the final implementation would base on OMNeT++. Although the open-source software, like e.g. freeware or GPL compliant software, has many well-known advantages, it is hardly ever guaranteed that the product development will go on. The high quality of the OMNeT++ package and the remarkably fast and qualified support by the author were not known to us at that point of time. Therefore, the independence from OMNeT++ – as far as possible using it as the simulation engine – was one of the conceptual intentions. Additionally, this would yield the possibility of reusing parts of the code for the real implementation or other simulations.

For that reason, the OMNeT++ API as shown in Figure 4.3-2, was extended to some higher Abstraction API, which was then used in the source code. Only the parts of the code, which were really OMNeT++ typical, like module handling of OMNeT++ modules etc., still use the pure OMNeT++ API.

This was achieved by sub-classing the offered OMNeT classes. Basically, each used class was replaced by the own classes and the used helper-functions available in the simulation library were wrapped by own function calls, usually calling the real library function. However, not only the independency of the function calls and class names was the intention. OMNeT++ uses the `cSimpleModule` class (see Figure 4.3-2) to implement both the module itself ant its needed network functionality like sending and receiving packets. In reality, each node can have some drivers representing whichever physical adapters. These drivers can have completely different meanings like network adapter or inter process communication driver. The drivers deal with different types of messages, like e.g. network messages and signals. On the other hand, the same message sending mechanism is also used in OMNeT++ internally for starting modules, etc.

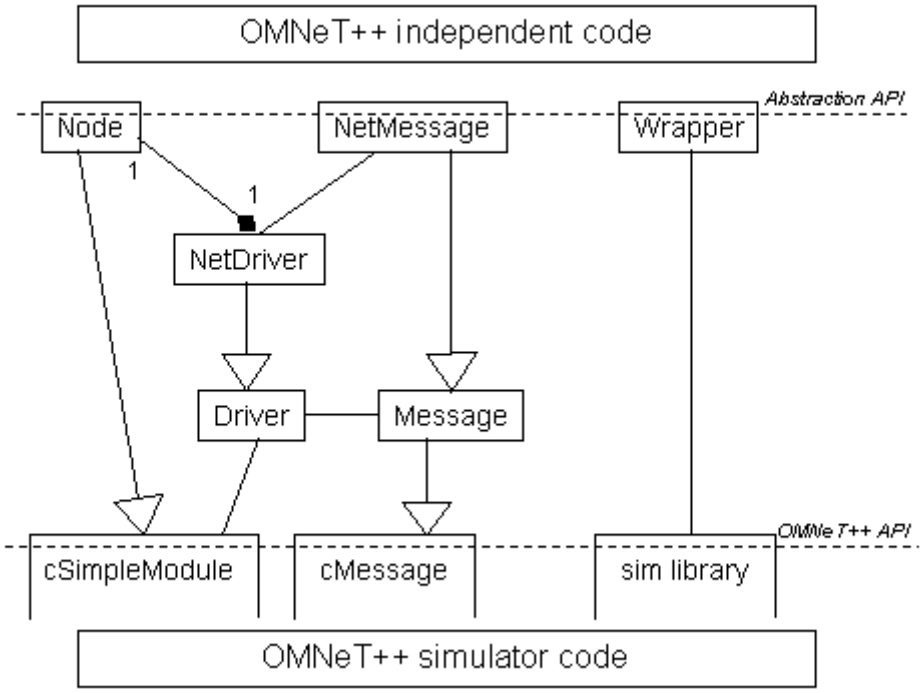

Figure 5.3-1 OMNeT++ Independency concept

That's why the representation of the node and its drivers was separated by sub-classing means (generalization) into two interfaces used for the OMNeT++ Abstraction API: into a module itself and a generic driver interface called `Driver` as shown in Figure 5.3-1. Since the message type depends on the driver the native `cMessage` class was sub-classed to a generic simulation message class called `Message`. Each `Driver` class uses that `Message` class. The active modules we are talking about in the Abstraction API are of the type `Node` representing a network host and therefore adding the needed functionality to `cSimpleModule`. The effective network driver used by `Node` is the `NetDriver`. Each `Node` *has* exactly one `NetDriver` combining all the really available network adapters. The `NetDriver` itself uses `NetMessage`, which represents IP-datagrams in a simplified form used in the simulation.

5.4 Class model for Mobile IP and FATIMA agents

Since Mobile IP agents are extended classical Internet nodes they should have all the functionality of the standard node extended by needed functions. So, it is a logical step to completely inherit the `Node` class proposed by the Abstraction API building the new `MobileIPNode` class like shown in Figure 5.4-1. From this class we can then define a specializations like `FA` or `HA` classes representing the foreign and the home agent respectively.

Almost in the same manner in which Mobile IP hosts extend the usual hosts the VMN hosts extend the common Mobile IP hosts. It is particularly true since we will not use VMN agents which are only used for the internal VMN purposes and do not participate in the Mobile IP traffic in the same way.

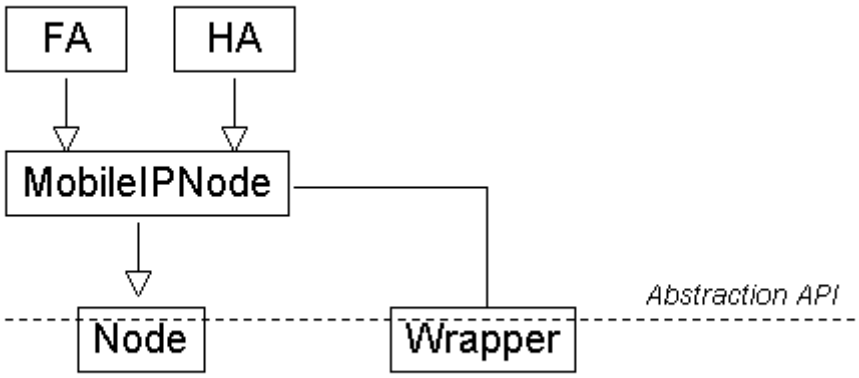

Figure 5.4-1 The `MobileIPNode` class as specialization of the `Node` class

The extensions we are talking about in both cases concern certain registration databases used by both Mobile IP and FATIMA agents. These databases, i.e. own mobile node register and visiting mobile node register, are to be represented by some class. They are used by the `MobileIPNode` class and can be used unchanged by the VMN agents. In the simulation the databases are instances of the classes `HomeRegister` and `VisitorRegister` as shown in Figure 5.4-2. These instances are always built and destroyed by the concrete instances of the agents.

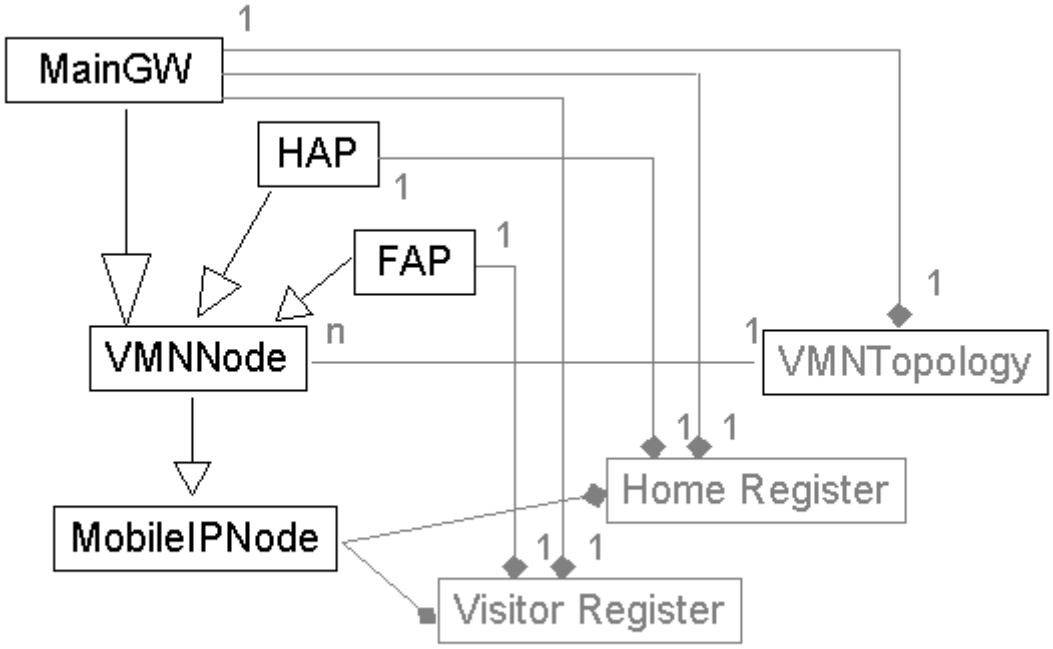

Figure 5.4-2 FATIMA agents as an extension of standard `MobileIPNode` class

The predecessor class – `MobileIPNode` – only has a relationship to it (e.g. just holding a common variable of this type in the real implementation). Each agent instance like FAP or HAP, but also FA and HA *has* exactly one instance of each type of the register classes (see Figure 5.4-2).

However, the VMN nodes maintain an additional internal structure representing the FATIMA-tree, which does not exist in the standard RFC2002-definition. This structure is managed explicitly by the central element in the FATIMA tree, i.e. it can be built, changed and destroyed only by the main gateway represented here by the `MainGW` class. This topology class, `VMNTopology`, is used by every FATIMA-node during its existence, i.e. by `VMNNode` class and its successors, `FAP` and `HAP` classes. In fact, as shown in Figure 5.4-2 the same instance of this class is used by all the FATIMA nodes in the simulation, which is of course not directly possible in reality.

This class model gives us the possibility of reusing the written code to a considerable degree. Additionally, the common parts of code are grouped in reasonable entities facilitating code maintenance. Finally, using this model we assure that the FATIMA agents based on standard Mobile IP use the same functionality and even the same data structures.

5.5 Datagram and broadcast sending with OMNeT++

As already explained in Chapter 4.3 OMNeT++ doesn't natively support any broadcast messages. However, this type of message sending is essential for IP and Mobile IP, particularly for Mobile Agent Advertisement sent by ICMP as explained in Chapter 2.3. Hence, a concept for broadcast message sending had to be found. Additionally we also need a possibility to logically interconnect all the available hosts within a simulated local network.

Of course, we could interconnect the involved hosts by OMNeT++ topology language means as shown in Figure 5.5-1, i.e. defining a pair of gates (input gate and output gate, as usual in OMNeT++) for every host connected to the concerned host. Though this concept seems pretty simple, it does not allow using the same interface for every module representing an "IP-host" since the gates have to be defined in the interface. Obviously, the number of gates doesn't scale well with the number of present hosts. Each time we change the network e.g. by adding an FA, we would have to add connections to *every* involved node i.e. to change all the already configured instances.

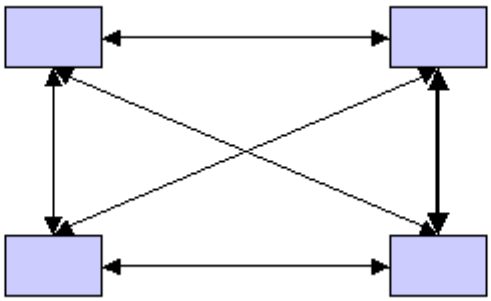

Figure 5.5-1 Broadcast in OMNeT++: interconnected modules

Another possible approach would be to build a chain of modules as shown in Figure 5.5-2. So, each module would be configured with two pairs of gates and its behavior would be extended to forward the received messages to the neighbor if the message is not destined to itself according to its IP-address. In this case, we could leave the interface untouched, however with unconnected gates at the respective end of the chain (or by building a circle). We would have to implement some reasonable simple routing algorithm to each host for re-sending the packages in the right direction. This approach allows us to add a host by changing only two other involved hosts in the worst case. The broadcast messages could be simulated by copying the message and resending it afterwards. Nevertheless, this approach does not represent the message flow in a real network rather imitating a physical layer functionality of a Token Ring or Token Bus systems than an appropriate network.

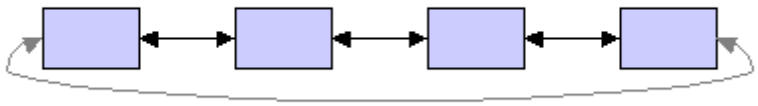

Figure 5.5-2 Broadcast in OMNeT++: chaining the modules

So, another concept was found introducing an additional module type symbolically representing the network itself. As shown in Figure 5.5-3, this module is connected to all the hosts like all the hosts are to be connected to the network in reality. The standard hosts do not do any routing. They just send messages over the only available output gate. The network module re-sends the messages over the right gate. In order to be able to do so, it maintains a kind of routing table linking the gate number with the IP-address of the host connected to this gate. Still, this is completely transparent for the hosts. Since the hosts are eventually representing the instances to be simulated, it seems to be very important.

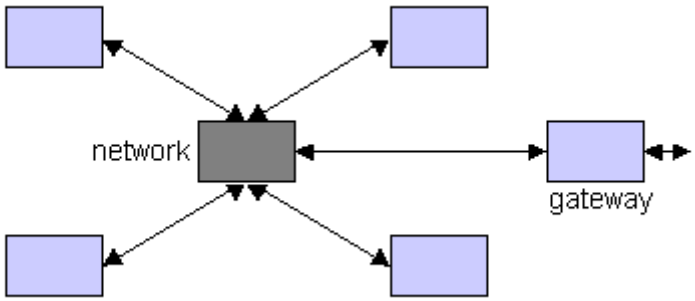

Figure 5.5-3 Concept finally used for the network simulation

The network instance itself can easily simulate broadcast by re-sending the broadcast message to all the connected hosts. Even a multicast concept is easily possible although not used in our simulation. Adding a new host just implies changes to the network module itself, namely an addition of a new pair of gates and a new routing entry. Though in this manner the network modules do not have a common interface, we have to consider that, as explained in the Chapter 5.2, there will be just two connected networks each using one module of this type. Apart from that, we will be able to re-use

the same module type to simulate the Internet backbone. Our network module will be interconnecting and routing between the two connected networks and optionally an independent correspondent node. So, we have a total of only three network modules – a number, which will hardly ever be changed and is easily manageable.

To interconnect the so built local networks we use a slightly extended host module type supporting two complete gate pairs for network communication (Figure 5.5-3). Just like in reality, these modules which we call gateways typically will not forward received broadcast messages, i.e. they will not copy such message from the input gate of the first pair to the output gate of the second pair of gates. So we can avoid the spreading of local broadcasts over the local network limits by the same means like it is done in real networks. Additionally, such gateway modules could e.g. check TTL values in the messages or simulate packet filters since they represent the only entry point to the network.

Having chosen a suitable concept for broadcast, we can implement some simple IP-routing into our network type modules and thus begin sending messages to IP addresses. Due to the chosen concept, in the simulation no ARP is needed. The messages always arrive at the wanted module's gate, routed by the network module on the basis of the included IP-address to the right link and from there on determined by the simulation engine on the basis of the topology file.

After these considerations we can now think about the message content. To determine it we will summarize the needed message types by claiming some facts:

- all sent network messages are (simulated) IP messages
- control traffic uses ICMP with Agent Advertisement and Mobile Router Advertisement for agent discovery
- control traffic uses UDP/434 for registrations
- data traffic could use any of possible data transport protocols defined by IP

First of all, we will use TCP as the only data traffic protocol. Of course, no real data will be sent between the simulated hosts, so the chosen protocol doesn't matter. The different transport protocols don't influence the comparative performance of the mobility support system since they represent the same overhead/work for every analyzed underlying mobility support layer. So, no data UDP packets will ever be sent in the simulation. The ICMP protocol will not be used either since there are no connection errors or non-existent hosts, etc. in the simulation. In this way, we could use the number assigned to the "UDP protocol" for the registration traffic and "ICMP" for the agent discovery explicitly. The protocol implementation itself isn't necessary.

According to that explanation, there are only three basic message types to be supported. Fully assembled they will look like shown in Figure 2.1-1. Their real content will vary due to their final purpose. The `Data` field of the data message will not be used to transport any data since the number of bytes to be transported serves the same purpose. The `MobControl` field in the UDP package represents one of the two possible registration messages types as explained in Chapter 2.3. The `MobExt` field represents one of the defined Mobile IP extensions. Since there can be an arbitrary number of such

extensions, they can be chained. The Agent Advertisement according to [RFC1256] and the Mobile Router Advertisement are combined in one field including all the necessary fields according to the mentioned standards.

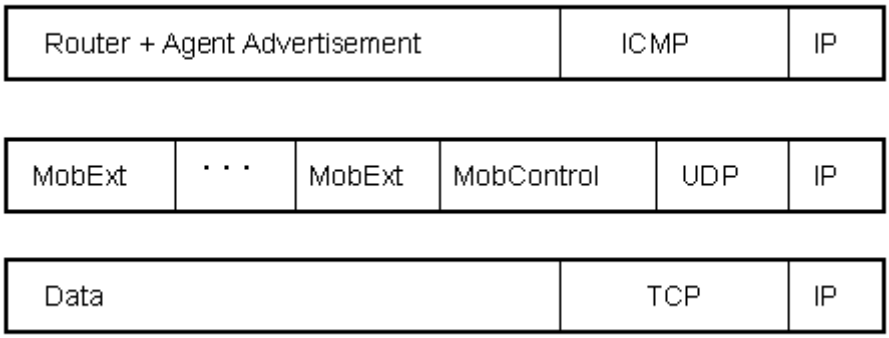

Figure 5.5-4 The format of the three basic message types used in the simulation

To simulate the structure of the messages, we will use the encapsulation mechanism provided by OMNeT++. This meets the conceptual criteria for the simulation and provides a lot of advantages from the point of view of the implementation. Since all messages will be sub-classed from the general `Message` class as already explained in Chapter 5.3 the encapsulation functions will be inherited. Hence, they can be directly used in the successor classes. The formation of the messages can then be done by building the deeper layer message, filling out the needed fields, building the upper layer message and encapsulating the higher layer message into the deeper one. The encapsulation provides automatic deletion of all encapsulated messages so we could easily implement the needed chaining for the Mobile IP Extensions. The corresponding UML diagram is shown in Figure 5.5-5.

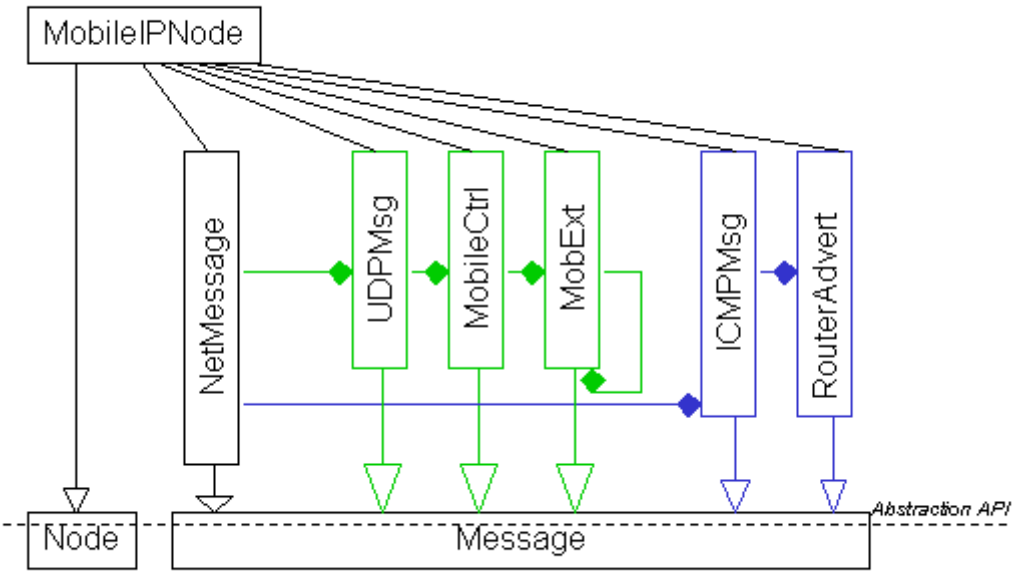

Figure 5.5-5 Simulation messages in the class model

Every message discussed here is meant to be used by the instances of `MobileIPNode` class. Obviously, all used messages are successors of the `Message` class. `NetMessage` class represents the IP datagram and has to carry its most important fields as shown in Table 5.5-1. The other fields may be implemented but are not essential for the simulation. The IP header options will not be used, so the header length is always 20 byte, which is important for performance measurements.

An instance of class `NetMessage` can compose any other instance of class `Message`. In our case it will always be either an instance of `UDPMsg` class representing the UDP payload or an instance of the `ICMPMsg` class representing ICMP payload. The `size` field of the `NetMessage` class itself and the correspondent TCP protocol number in the protocol field represent the TCP payload, so no explicit class representation is required.

The `UDPMsg` class defines some additional essential fields (→Table 5.5-1). An `UDPMsg` instance always carries an instance of the `MobCtrlMsg` class which fields can be found in the same Table 5.5-1. Apart from those fields, `MobileCtrl` maintains a reference to a `MobExt` class, which adds the necessary fields (Table 5.5-1) for the Mobile IP Extension mechanism as defined in Chapter 2.3. Apart from that it is capable of building a chain by carrying a reference to itself.

NetMessage	UDPMsg	MobCtrlMsg	MobExt	ICMPMsg	RouterAdvert
source IP	source port	Type	type	code	Lifetime
dest. IP	dest. port	Code	data		sequence num.
protocol ID	length	life time			type
length		own IP			length
		HA IP			registration life
		CoA IP			flags
					CoA

Table 5.5-1 Fields used in the simulated message classes

In the same way, the instances of the `ICMPMsg` class carry instances of the `RouterAdvert` class representing both Router Advertisement and Mobile Agent Advertisement messages with all important fields combined in one class. All additionally defined fields can be found in the Table 5.5-1.

The meanings of the fields in Table 5.5-1 correspond to the explanations given to the respective standard in Chapter 2.3.

5.6 Mobility simulation and handover

After some suitable concepts for the broadcast messages, standard IP-networking and the message formats have been found one has to begin to think about a mobility concept within the simulation. This concerns several things at the same time as will be explained in the following. Most of the topics discussed in this scope only affect the simulation and do not exist in reality as conceptual problems. Thus, these topics depend on the chosen simulation engine, OMNeT++, and its possibilities to a great extent.

First of all, it should be clarified how it is planned to bind a potentially mobile node to a given network. Evidently, we cannot connect it as a usual host as explained in Chapter 5.5. On one side, this would force the user to give an exact number of mobile nodes in advance (in the topology file) and apart from that the changes to the so defined modules are not yet represented by the OMNeT++ GUI, which however would not change the simulation results. On the other side, the possibility would have to be found to intercept the packets destined to that mobile node to simulate ProxyARP behavior as explained in Chapter 2.3. This cannot be easily achieved if the mobile nodes are connected directly to the network.

Secondly we have to think about the simulated movements. A mobile node has to be able to change its binding from his own network to the foreign network. If the foreign network maintains several foreign agents the mobile node has to be able to change its local binding within the foreign network. This topic should be discussed under three different aspects. The aspect concerning OMNeT++ would be to find the possibility to fulfill the hereinbefore mentioned requirements, i.e. to move a module or to change its current binding with the simulation engine. The aspect concerning the simulation-work itself would be to take care of the accessibility of the MN, the routing and the interception of packets after the handover with the same implementation. The mathematical aspect would be to find a (parameterizable) function describing some typical handover behavior for certain networks.

In the following we will show how all these requirements could be fulfilled within this simulation.

OMNeT++ supports dynamically created modules and connections. These are not shown in the available GUI but apart from that limitation these modules and connections act as usual once they have been created by some other module. Additionally a message does not necessarily have to be sent over a defined connection. OMNeT++ supports so-called direct message sending →[OMNeT Man], i.e. knowing the unique module ID number it's possible to deliver a new message to an appropriate module's input gate.

Using these OMNeT++ features we define a new module type, which will remain transparent for the rest of the simulation almost like a network module is transparent to the connected hosts. This module type is called `MobileGen`, i.e. *Mobile Generator*. Each host acting as an agent will get a more complicated topological structure as shown in Figure 5.6-1. Through the `agent gate` each Mobile Generator will be connected to the agent core module, i.e. one MG to each HA, FA, HAP and FAP. The last four modules types act as usual hosts, however supporting the respective mobility system and being connected to the two other modules, a module of type `Network` and a module of type `MobileGen`. As soon as these modules receive a packet, which is not

destined for their own IP address they forward the received packet to the second gate pair, sometimes being forced to do more work (like decapsulation, decryption/encryption, address translation etc.). One could say that these modules have two network interfaces, an interface connecting the agents to the mobile participants and an interface connecting them to the local network.

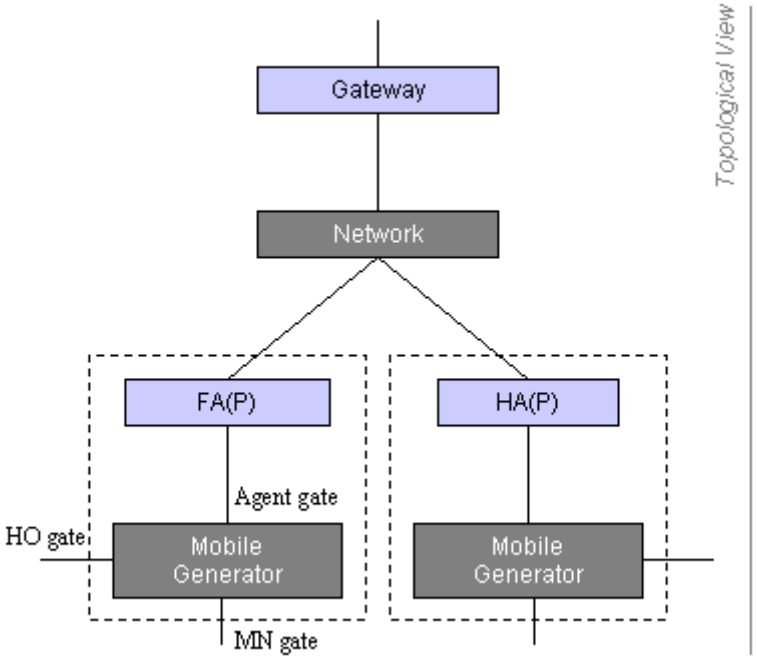

Figure 5.6-1 Mobile Generator in the topology

The Mobile Generator being connected in this manner is the representation of all mobile nodes from the point of view of a mobility agent connected to this MG. However, it remains transparent for the agents. All the messages sent by the mobile nodes arrive at its MN gate as shown in Figure 5.6-1. The Mobile Generator silently relays any traffic between all available MNs on one side (`MN gate`) and the agent on the other (`agent gate`). That includes distributing broadcast messages, which are copied by the Mobile Generator to each currently available mobile node module. One could say that the Mobile Generator emulates a wireless connection to the MNs in vicinity. The messages to the MNs are exchanged by an OMNeT++ mechanism called *direct message sending*. To use that mechanism each Mobile Generator has to know the OMNeT++ internal module IDs of the available MNs by maintaining a table of them. This is achieved by consequent MN handling from the beginning on as will be explained later.

The Mobile Generator (MG) instances are all the same and do not depend neither on the type of the connected agent (like e.g. FA / HA) nor on the type of the simulated system (like e.g. FATIMA FAP / Standard FA). However, they can be configured in a certain way e.g. obtaining the number of the MNs to be created and the IP-addresses for these nodes. One Mobile Generator per network creates the necessary number of mobile nodes according to these configuration parameters. Having created an MN instance it configures it. The MN configuration consists of the IP-configuration parameters used by every host i.e. the node's IP addresses, the network masks and the IP address of the responsible HA. Additionally, a parameter holding the currently responsible Mobile Generator ID for communication purposes is passed to every MN. This parameter is

only used by the network driver of a mobile node, which is different from a usual host's network driver. This driver uses direct message sending to send messages to the Mobile Generator. Though each MN is configured in this way this parameter remains completely transparent for the IP-part of the simulation of the Mobile Node. I.e. being done by the driver, message sending looks exactly the same like in the general host. Every MN-module created in this manner is inserted in the shared central list holding all available MNs and maintained by the MGs (see Figure 5.6-2).

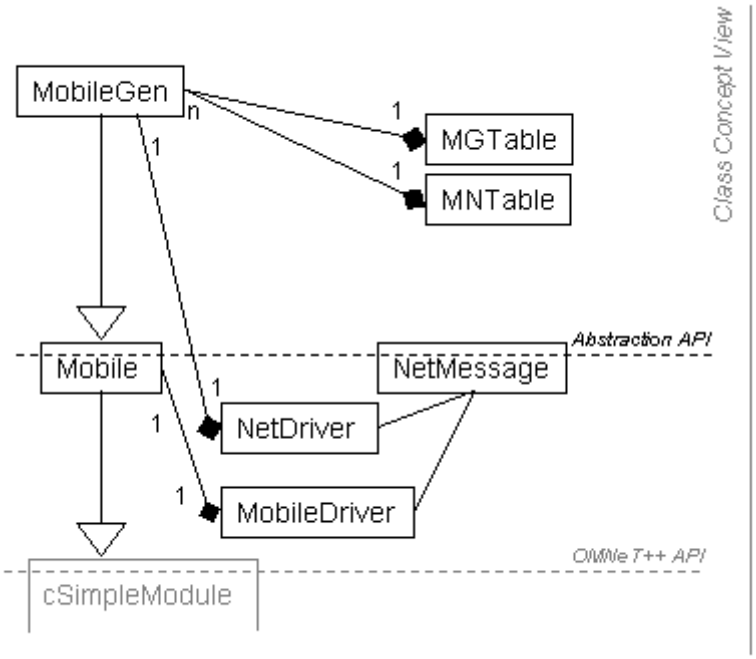

Figure 5.6-2 Mobile Generator in the class concept

In the class model the mobile generators are represented by the `MobileGen` class as shown in Figure 5.6-2. All instances of this class share two common databases. The `MGTable` holds the IDs and types of all available mobile generators while the `MNTable` is used to store all the built mobile nodes with the respective currently responsible mobile generator and some organizational information. The `MobileGen` class itself is the successor of the class `Mobile`, which has not yet been presented. This class is a member of the Abstraction API and is used by classes representing mobility supporting modules i.e. by mobile nodes and mobile generators. This class is a direct successor of the `cSimpleModule` class from the OMNeT++ API. It uses a `MobileDriver` instead of the `NetDriver` used by the classical hosts represented by the `Node` class as described in Chapter 5.3 (see Figure 5.3-1). Since a `MobileGen` represents a Mobile Generator, whose topology is shown in Figure 5.6-1, it uses both direct message sending and usual message sending through the topological connections. Hence, the `MobileGen` class makes use of both `NetDriver` and `MobileDriver`. Both drivers are successors of the common `Driver` interface class and use the `NetMessage` class.

Only the MG which is connected to the HA / HAP instance of some network creates mobile nodes for this network. Since there is only one home agent instance in our simulation there is also only one MG creating mobile nodes. Having created the MNs

and configured their IP-configuration, this MG chooses a random MG, which this MN will be connected to in the future. Obviously, it doesn't make any sense to bind mobile nodes of some network to the foreign agent instances of the same network. For that reason, all MGs have to share some MG-table as shown in Figure 5.6-2. This table holds the module IDs of all available MGs grouped by their respective IP-network bindings. E.g. MGs connected to the foreign agent instances in one topological network obviously belong to the MG-group of the other IP-network since they provide the communication with the mobile nodes coming from that net. Since our simulation only uses two networks at the same time, all MGs can be divided into two groups including the home agent from the first network and all foreign agents from the other network respectively. Thus, each creator MG chooses one of the MGs of the same group and assigns the ID of this MG to the MN. After that, the creator informs the newly assigned MG and starts the MN module.

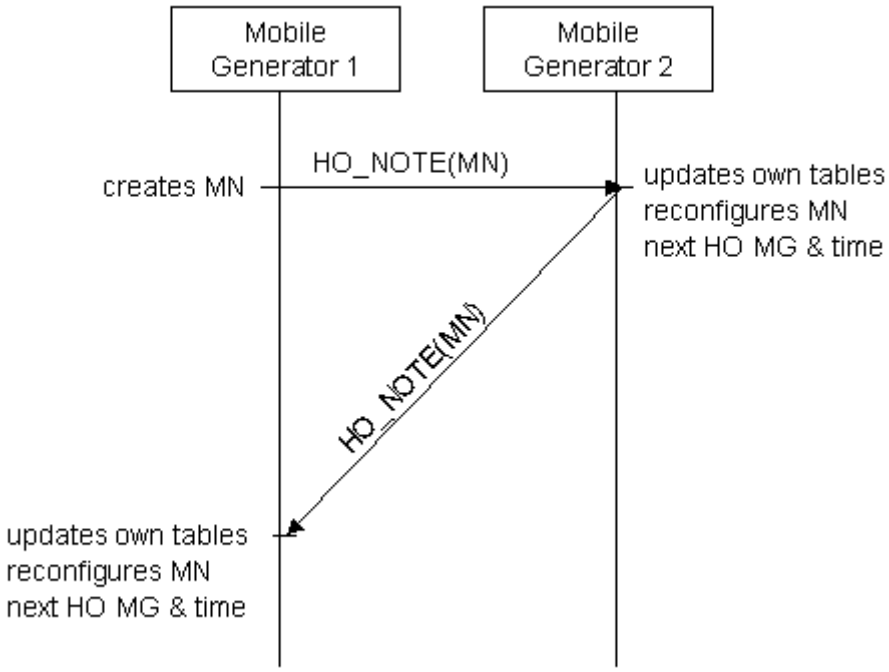

Figure 5.6-3 Creation of Mobile Nodes and their distribution over the networks

In Figure 5.6-3 we can see a scenario demonstrating the handover concept in the simulation where the `MobileGenerator1` is the creator-MG and the `MobileGenerator2` is one of the MGs connected to a foreign agent instance in the second network. The time axes show the simulation time tracked by the simulation engine. This explains that the handover notification message for the created MN-module (`HO_NOTE(MN)`) sent after the creation arrives immediately at the MG2, i.e. in zero time. This message is sent using the direct message sending mechanism. The MG2 receives the HO notification and adds the MN ID to its MN-table. It reconfigures the parameter of this MN holding the ID of the responsible MG adjusting it to its own ID. From that moment on the MN sends all its messages over the new mobile generator since the MN's driver (an instance of `MobileDriver` class) always re-reads this parameter before sending. Right after this reconfiguration, the MG2 determines the values needed for the next handover, which will represent the first simulated movement. The values computed with the help of some probability distribution functions are the ID

of the next responsible MG and the point of time at which this handover will occur. Having computed these two values, the MG2 sends a notification message to the respective computed MG, i.e. to the MG1 in the case shown in the Figure 5.6-3. That represents a global handover back to the home network. However this message is sent using the so-called delayed message sending. Its delay corresponds exactly to the point of time for the next handover computed before. MG1 receives this notification message at some time where this handover has to be processed, i.e. when the MG1 has to take over the mobile node. Thus, MG1 repeats the procedure described above updating the MN-table, reconfiguring the MN and computing the next handover event parameters. This is repeated till the end of the simulation.

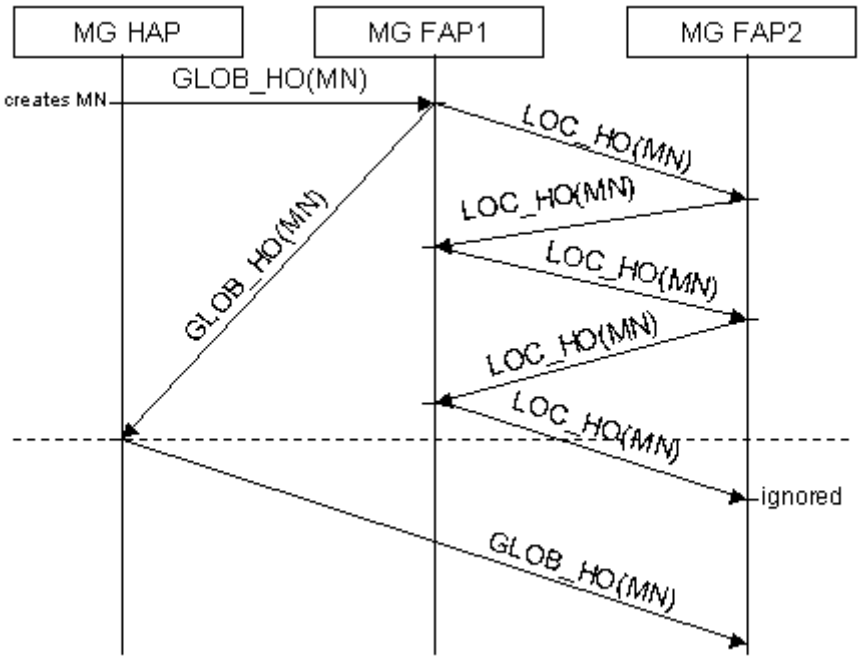

Figure 5.6-4 Handover sequence example

To complete the brief explanation above we have to state more precisely which probability distributions are used and in which way the different kinds of handovers occur. First of all, having received a notification message announcing a global handover, each MG has to compute the parameters for the *two* following handovers: a *global* handover back to the other net which is supposed to occur later in the average case and a *local* handover occurring more often in reality because of the geographical proximity of the available FA-instances. A possible handover sequence is illustrated in Figure 5.6-4. The interesting point in this sequence is the correction of a late arrival of a local handover announcement. Due to the fact that two announcements are sent for the same mobile node, there is a possibility that an announcement for a local handover for some mobile node (MN) arrives at one MG (MG FAP2) later than an announcement for a global handover for the same mobile node arrives at some other MG (MG HAP). In this case, the local handover at the second MG (MG FAP2) should be ignored, since the mobile node has already moved into the other network. However, the recognition of those cases can be easily implemented using the shared table of mobile nodes and setting some control flags. Additionally, the MGs connected to HA(P) instances never send announcements for local handovers since there can be only one HA(P) per network.

As described above, the available MGs are grouped into two groups according to the MN network addresses their connected agents are dealing with. Recalling that our concept uses only one HA(P) per network we can say that choosing an MG from the same group but of another agent type represents a global handover. Similarly, choosing an MG from the same group and of the same type represents a local handover. And finally, choosing the same MG by chance represents the situation where an MN stays at the same location. In the last case the notification message is sent by the currently responsible MG to itself, with the delay computed in the same manner as usual.

Whether the handover occurs or not is decided with the preset probability constants. Random values generated with the help of the uniform distribution function provided by OMNeT++ are checked against these values used as upper limits. If the drawn number is smaller, the handover occur after a random period of time computed later:

$$\varpi_{local} = 0.7$$

for a local handover and

$$\varpi_{global} = 0.3$$

for a global handover. This can be easily changed by two provided parameters.

Once having decided that the handover should occur, the random delay time till the next local handover is computed using the exponential distribution with a preset mean *T*. The computation of the delay time for the next global handover uses the same exponential distribution, however with a higher mean parameter:

$$d_{HO} \sim \exp(k \cdot T) \text{ , where}$$

$$k = 1 \text{ for local handovers,}$$
$$k = 6 \text{ for global handovers,}$$
$$T = 1000s$$

Please refer to [Ross 00] or [Law et al. 99] for general information on simulation models and to [Forsb et al. 99] for the particular case of the distribution of mobiles. See also Chapter “References/Simulation” in this document for more literature.

5.7 Mobile Node

As already explained in Chapter 5.6 the mobile nodes are created dynamically during the execution of the simulation. Nevertheless, the mobile nodes have a topology definition, basically defining them as usual hosts but with an additional parameter holding the actually responsible mobile generator.

According to the goals of this work two mobility supporting networks have to be compared. Because of that structural accentuation, the developed model for the simulated mobile node completely ignores all cases in which a mobile node acts as a stand-alone mobility supporting host, i.e. without relying on a present mobility supporting system. Thus, there always has to be at least one foreign agent instance per network in the simulation. On the other hand, in this manner the mobile node does not

need to know anything about e.g. IPIP encapsulation / decapsulation or dynamic address assignment as necessary for obtaining a CoA. This helps to simplify the implementation of the simulation.

Thus, in the simulation the mobile node always sends data in the same way. Analyzing the remaining functionality one can see that a mobile node principally has to support two different modes, a classic mode without any mobility support and a mobile mode being registered at some CoA. However, apart from that the simulation has to support an additional mode, which represents the initial situation when the mobile node is not yet sure about its position. Further analysis shows that the mobile node actually can be completely described by a finite state machine (FSM) with four states as shown in Figure 5.7-1. The MN has to reply to the received messages depending on the current state. The fourth state represents the case in which an MN has already sent a registration request message but has not yet received a reply.

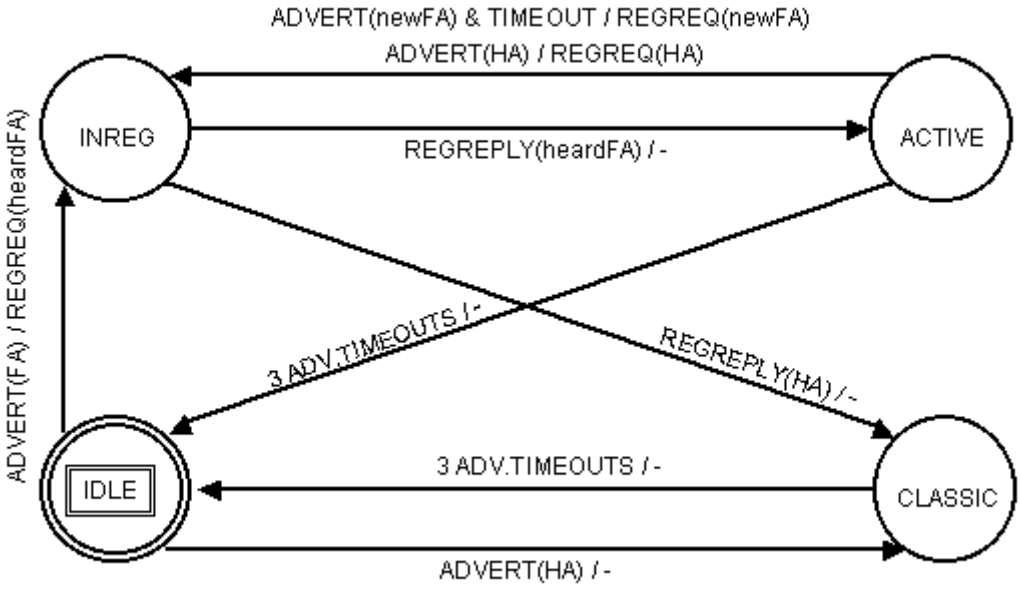

Figure 5.7-1 Simulated Mobile Node as a Final State Machine (FSM)

In Figure 5.7-1 the circles symbolize the states and the arrows the possible state transitions. The used action / reaction format is INPUT MESSAGE / OUTPUT MESSAGE. Any messages not explicitly mentioned are ignored. (To be more precise, such messages cause reflexive transitions without any output. However, these arrows are not drawn for clarity reasons). The IDLE state is the initial state as described above. Each MN begins in this state right after its creation. Having received an advertisement message from an HA, the MN does nothing and changes it state to CLASSIC which represents the Internet host mode without mobility support. If the next expected advertisement message from the home agent is missed at least three times the MN assumes that it has lost the connection with its home network and changes into an IDLE state. Having received an advertisement message from an FA, the MN emits a Registration Request message for the obtained CoA to the heard agent and changes into the INREG state representing the state during the registration. From this state the MN changes into the full operational ACTIVE state after having received a Registration

Reply from the contacted FA. ACTIVE and CLASSIC states are the only states which allow to send data packets. On three timeouts of the active FA, the MN returns from the ACTIVE state back to the IDLE state. However, having heard an advertisement of some other agent after just one timeout of the old agent advertisement, it changes its state back to INREG and tries to register with the new agent emitting the corresponding Registration Request. This case also includes the case of de-registration, in which the new heard agent is the home agent. Depending on this, the MN waits for the Registration Reply and changes to the ACTIVE or to the CLASSIC mode respectively.

In the class concept, the class `MobileNode` represents the Mobile Node, which is basically a direct successor of the class `Mobile`. This Abstraction API class is shown in Figure 5.6-2. It is used with the Mobile Generator but since both modules use direct message sending they are sub-classed from the same base.

6 Identified conceptual problems in FATIMA

During the implementational work on the FATIMA network components several conceptual problems and misunderstandings have been identified within the draft. This is not remarkable since there is still no official paper to this concept except of [FATIMA] which is just giving an idea about how the concept could work theoretically. This implementation gives only a possible realization of some basic ideas and therefore does not try to solve all the conceptual problems. However, the identification of these problems is an important contribution to the completion of the work on FATIMA. Where it was directly possible a suitable solution for the problem found has been proposed as described hereafter. In other cases, a workaround within the simulation temporarily solved the problem, which, however, remains unchanged in reality. The nature of the remaining problems finally forced us to renounce the whole related feature, as in the cases where a work-around within the simulation would undergo the problem and therefore be unessential.

6.1 Insufficient database entries

This problem described in this chapter is very close to the implementation and can be attributed to the new address substitution and more complicated detour schemes in FATIMA. While in the traditional RFC2002-system only one level address substitution occurs, like e.g. HA→FA→MN, in FATIMA we have to deal with multi-level address substitution, like e.g. HA→FMIPGW→RA→RA→FAP→MN, and completely new detours. Normally this does not cause any trouble, because already [RFC2002] defines temporary entries holding the needed information during the registration process. However, at least in one case, the registration entries of the involved hosts have to be extended.

If the standard HA receives a registration request message from one of its MNs trying to register from a foreign network, it checks the registration information and generates the Registration Reply message. To do so, it swaps the IP source and destination addresses and the UDP source and destination ports, includes the needed UDP payload, i.e. the reply information, its extensions and sends the message. No temporary entries are needed in this case. However, when the FATIMA central gateway receives a Registration Request message from one of its own MNs from outside of the network, it forwards the Registration Request to the responsible HAP (also see Chapter 6.4 in this context). As explained in Chapter 3.2 it changes the HA address field to the address of the HAP. The central gateway knows the structure of the tree and can therefore choose the right FATIMA sub-tree, i.e. a connected RA or IGW, at which this HAP will finally be reached, perhaps over some additional forwards. That's a standard task of RAs and IGWs, so no problem will occur here. The HAP can now generate the Registration Reply in the same manner as a usual HA swapping the source and destination fields of the received message. The RAs and IGWs have to lead the message back to the central gateway. Even if we assume that RAs and IGWs have some special features in order to be able to do so, the central gateway which does not have any temporary entry will not be able to relay this reply to the MN. The correspoding case is described in [FATIMA], p175, Chapter 11.1.2.

So, the solution is obvious: before forwarding the message the central gateway has to make a temporary entry for the sender-MN which holds at least the fields presented here in the following in detailed form. For each field its name, its corresponding data type for the implementation and a short explanation are given.

Temporary entry for the registration of the own mobiles from some foreign network should at least consist of:

- IP-address of the original sender of the request message
  Type: `unsigned integer`
  This is necessary to be able to send the message back to the original sender

- UDP port of the sender
  Type: `unsigned short integer`
  This is necessary to be able to send the message back to the right port

- CoA
  Type: `unsigned integer`
  This is necessary since in the Reply message no CoA is included and so, no permanent entry could be built for the related MN

We assume here that the destination IP-address and UDP-port originally present in the Request message can be derived by other means, i.e. the corresponding process knows at which coordinates it is reachable from outside. Additionally, we propose to include these two fields:

- Lifetime in seconds
  Type: `unsigned short integer`
  To be able to delete this temporary entry if no reply is sent for which reason ever. The same type as proposed in [RFC2002] is used.

- Temporary flag
  Type: `boolean`
  By this means, this entry could easily be transformed into a permanent entry. However, this depends on the chosen organization of the corresponding entry database.

These temporary entries have to be made in every FATIMA-node on the path from the central gateway to the selected HAP in order to give them the possibility to relay the answer to the respective sender in the tree.

6.2 Firewall problem

The firewall rules defined in the FATIMA draft correspond to the well-known firewall configuration with two packet filters and a screened subnet between those. The central gateway (MIPGW) is then installed as a *bastion host* →[Ches&Bell 94]. This configuration is explained in Chapter 3.5 and illustrated in Figure 3.5-1.

On the consequential way to the strictest possible firewall rules setup only one exception has been tolerated. That is the missing ingress filtering on the outer packet

filter (Table 3.5-1, rule 3), which is a work-around against the reverse triangular routing phenomenon as explained in Chapter 3.5. The FATIMA developers suggest installing the remaining rules due to the Table 3.5-1. In the meantime, more research has been done for the security of the Mobile IP systems →[MIPSEC] and the firewall integration →[FATREP]. Moreover, the system has been given exact firewall allow rules for the necessary protocols which can be found in [FATFW].

However, we should think about the possible implementation of a FATIMA process in this scope. If the FATIMA implementation should be done in the style of the implementation →[Dyn HUT] of standard RFC2002 Mobile IP system, then it should be possible to implement the FATIMA-software as a user-mode process. If doing so we should assume - as it is assumed in Mobile IP - that each reconfiguration of the system itself and of the other co-existent user-processes is done through a general proposed API. Taking an exact look at the proposed firewall configuration one can see immediately that the connectivity of the Mobile Nodes being at their home network and *not* using any mobility support is highly limited. In fact, each connection request to the servers outside of the local network has to be made over the proxies within the screened subnet. Logically, each co-existent software installation, perhaps even the operation system itself, uses the proxies if any connectivity is provided, as e.g. HTTP and FTP proxies or more general proxy configurations as possible e.g. with [RFC1928].

Having left the home network, the connectivity of the Mobile Node depends on the used data-sending mode. First of all, it is obvious that no problems with the registration can occur because of these firewall restrictions since the firewall traversals both to and from the central gateway are explicitly granted. Secondly, Mobile Node has to use Reverse Tunneling option as described in Chapter 2.6 in order to be able to send data to its own network members (as e.g. to other MNs being at home). Obviously, if an MN is using the reverse tunneling, its encapsulated packets will arrive at the central gateway. (Provided that the outer packet filter doesn't drop the encapsulated packets). Thus, the gateway can decide if the decapsulated packets will be forwarded to the respective receiver. Although it gives the possibility to restrict the Mobile Node connectivity in the same way as if it were at home, anyone will agree that principally a higher connectivity is given to each Mobile Node being *not* at home. This is somewhat confusing. However, even supposing that an MN has been given the full connectivity out of some foreign network, it will hardly ever be able to use it. Instead of doing so it will continue to address all its packets to the proxies of the firewall due to its software configuration being set-up for the home network. To be more precise, the MN will encapsulate its proxy requests and send these packets in a reverse tunnel or an AH or ESP tunnel →[RFC2406] (if using the authorized firewall traversal) back to the central gateway which can then decapsulate and deliver them locally, within the screened subnet.

Evidently, this would undergo any routing optimizations and finally result in a highly sub-optimal "routing". Till now, no reasonable automatic and generally usable possibility exists which permits the dynamic re-configuration of the installed software by some user process. Even assuming that the concerned OS uses central proxy configuration, we will hardly ever find a reasonable API to re-configure it from a user-space process.

We understand that this problem is neither a native FATIMA problem nor provoked by FATIMA itself. But since FATIMA claims to be able to install a reasonable firewall support within a Mobile IP compatible system, it has to be solved first. For this reason,

the firewall configuration was not inserted in the simulation program yet. The packet filters are not implemented and not installed. However, a concept for such filtering exists and can be added later.

6.3 Mobile Node incompatibility

Another FATIMA-concept problem could be detected with the help of this implementational work. A functionality definition within [FATIMA] causes an incompatibility of the FATIMA Mobile Node to the [RFC2002] Mobile Node. This would be a severe problem since one of the most important advantages of FATIMA should be the unchanged Mobile Node implementation and with it the downwards system compatibility. Till now, no reasonable concept work-around has been proposed although a solution seems possible. However, this problem doesn't concern the simulation much since we can easily adapt the implementation of the Mobile Node to support any requirements. Evidently, since we do not know the work-around yet the MN changes within the simulation could affect the reliability of its results.

A classical [RFC2002] compliant mobile node is pre-configured with the IP-address of its Home Agent. This address is used both for the primary Move Detection algorithm as defined in [RFC2002] (see there chapter 2 and 2.4.2) and as the Home Agent address in the Registration Request messages as shown in Chapter 2.3 of this document. Receiving the Mobile Router Advertisements usually sent as broadcast in the local network, an MN compares the sender IP-address with the pre-configured one and decides if it is currently in its home network. It works since each Home Agent sends packets from its own IP-address as a default. Missing those advertisements but receiving an advertisement from some FA, a Mobile Node decides that it is not at home and sends a Registration Request message to exactly the same pre-configured IP-address.

However, [FATIMA] changes this situation by introducing a structure of available HAPs and the central gateway. In the case of FATIMA, only the central gateway is reachable from outside the network, so it acts as a Virtual HA as described in Chapter 3.2. Obviously, because of this access restriction each MN has to be pre-configured with the address of the central gateway as its Home Agent address. However, returning or beginning in the own network the MNs have to recognize their HAP as the home agent (→[FATIMA], p175). I.e. the Move Detection algorithm suffers under this configuration, because the Mobile Router Advertisement messages are sent from the address of the respective HAP, which therefore cannot be identified as the responsible Home Agent since the MN is pre-configured with only one IP-address.

A possible work-around would be to change the behavior of the HAPs forcing them to advertise the address of the central gateway. However, the de-registration messages would then be sent to the central gateway directly and not to the HAPs first as defined so far (see Chapter 3.2). Additionally, each available HAP would be recognized as the responsible Home Agent by each MN of this network. Originally, it was planned to configure only one HAP as the responsible proxy per mobile node. In particular, the network segmentation by the IGWs would not make much sense if the MNs did not try to register within the FATIMA structure either.

6.4 Selection of the responsible HAP

This issue is rather a deficiency than a real problem. In [FATIMA] in its actual version there is a case for which no behavior has yet been defined. It occurs within a FATIMA network using at least two HAPs. Hence, it does not concern the simulation in its actual conceptual form. However, originally it was one of the reasons to limit the simulation to use only one HAP per VMN.

When the central gateway of a FATIMA network using several HAPs receives packets from its own absent Mobile Nodes trying to register from some foreign network the gateway has to decide to which HAP the request will be forwarded to. Unfortunately, no basis for this decision has been defined yet. This situation is not even mentioned within the FATIMA draft.

However, this question arises and some guideline should be defined in any case. Possible approaches are e.g. the decision based on the sub-network segmentation of the own network or perhaps even a load-balancing mechanism dynamically assigning a free HAP to each MN. The final decision in this scope mostly depends on the intentions of the developer since each approach can theoretically increase the load at the central gateway simultaneously decreasing the load at the HAPs and vice versa.

7 Implementation with OMNeT++

The implementation was completely done in C++ using OMNeT++ simulation library and its NEDC language. The NEDC language defines the topology of the networks to be tested. The C++ classes define the module behavior. Due to the nature of the NEDC language, there is no necessity for a complicated concept. The NEDC definition has been split into two definition files, the first one holding the basic module and connection types but also the more complicated modules built according to the concepts described in Chapter 5 and the second one defining the involved networks and the topologies to be simulated. The C++ files hold the corresponding classes. As customary, the implementation and the headers are separately stored in the corresponding `cc-` and `h-`files. There are some basic concepts, which are used in that simulation besides the ones already described in the chapters before.

7.1 NED nomenclature

All the modules defined within the NED files use the same names for the basic necessary parameters. These obligatory parameters have been defined as NED-strings and given a nomenclature. The complete list can be found in the Appendix B:.

The modules gates have also been given some general names. All the gates are named using the following scheme: depending on the direction from the point of view of the related module, one of the two possible prefixes "`to_`" or "`from_`" is used. The identifier appended to this prefix describes the nature of the gate. "`net`" is used to describe gates representing network adapters.

See the corresponding NED files for more information.

7.2 Root class concept

The basic class for almost each class involved in the simulation is the `RootClass`. Only few classes representing independent data structures or small unimportant objects are not sub-classed from that class.

The `RootClass` is completely virtual and is basically used to define global constants. With the help of these constants one can change e.g. the obligatory association with the NEDC-definitions as mentioned in Chapter 7.1. Additionally, the simulation parameters can be set here as e.g. the probability values, delays, message lengths, etc. The main advantage is, however, that all the involved classes are `RootClass` descendants and therefore have some common behavior basis.

Because this class is always used, its header file is imported everywhere in the simulation. Thus, this header file `RootClass.h` is used to define the pre-compiler macros and types, some C++ enumerations, general C++ I/O-functions and string operators for classified simulation error and warning reporting. This makes that file the right place to set global definitions.

7.3 Inner structure of nodes

The `Node` class is the basic class for all IP-modules. As a basic class, it defines the needed elements and structures, among others the references to the `NetDriver` and the `RouteTable` classes. Additionally, it provides a new API, which can then be used in all objects of this class and its descendants. Deriving classes from the `Node` class saves the IP-initialization in every descendant, since everything needed for the simulated IP-layer is set up in Node, i.e. the added structures are initialized correctly and all variables are set according to the provided parameters.

As a direct descendant of `cSimpleModule`, `Node` itself has to implement its virtual API functions, `activity()` and `finish()`. These two functions are started automatically by OMNeT++ for each module. The first, `activity()`, is started after the creation of each module and the second, `finish()`, before its destruction. Each module is destructed automatically by OMNeT++ when its `activity()` function returns. Hence, user has to program an infinite loop and to put the permanent behavior of this module into this loop. The `Node` class is intended to take care of this. It provides three new member-routines named `start()`, `process()` and `stop()`. Each descendant of `Node` should implement these three procedures. `start()` should be used to set up own variables and initialize the needed structures; `process()` represents the main behavior and will be repeated automatically until the end of the simulation. Finally, `stop()` should be used to clean up the structures built before in `start()`.

This is done by `Node` as shown in Figure 7.3-1. In the beginning of `activity()`, the IP initialization is started, then `Node` calls the (still) virtual routine `start()` and jumps into the infinite loop of `process()`-calls. OMNeT++ will interrupt the infinite loop once the simulation time is over and the `finish()` function will be started automatically. So, `Node` implements this function first calling its own `stop()` routine and cleaning up the built structures afterwards.

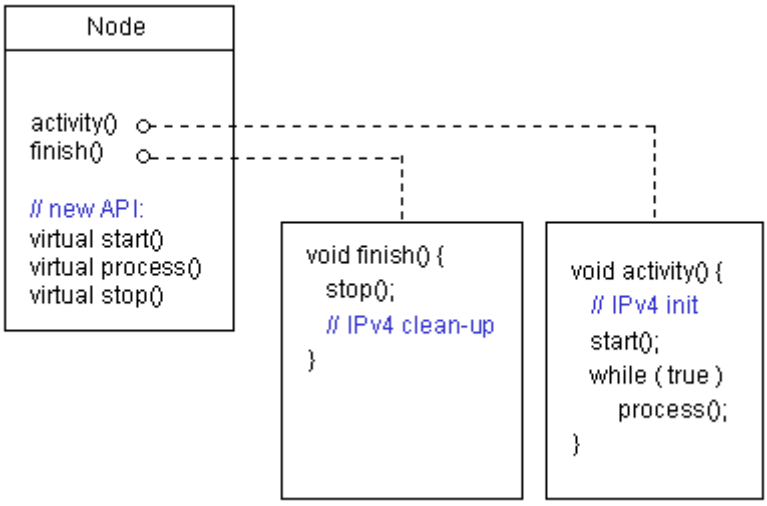

Figure 7.3-1 The new API as proposed by the Node class

This scheme is used in the same manner in the `MobileIPNode` class, which is a direct descendant of `Node` and then, right in the same way, by the `VMNNode` class which is the direct descendant of `MobileIPNode`. `MobileIPNode` prepares the Mobile IP structures while `VMNNode` takes care of VMN typical structures as explained in Chapter 5.4. Therefore, these two provide their own interfaces, `mip_start()` (`vmn_start()`), `mip_process()` (`vmn_process()`) and `mip_stop()` (`vmn_stop()`). Similarly, the corresponding structures are set up automatically in the previewed places and then the right function is started automatically in the predecessor classes.

7.4 Message flow and types in agents

The message processing of received messages in the agent-modules is always done according to some common flow path as illustrated in Figure 7.4-1. First of all, the message is received from the respective `Driver` instance using its `receive()` call. The messages marked as UDP are then given to the control process while the messages having some other type are given to the data process. (Please note that the logical control flow in the program is meant; the processes do not have to be necessarily extracted as stand-alone routines). The control process is supposed to process and to generate answers to the received Mobile IP messages like Registration Request or Reply. Data process is supposed to extract the possible encapsulated messages and to re-send them. Both processes can generate and return an answer message. The return value is then validated and the original message is deleted if no return value is produced. Otherwise the message is sent using the `send()` call of the respective `Driver` instance. In this way, the received message is re-used as proposed by [OMNeT Man].

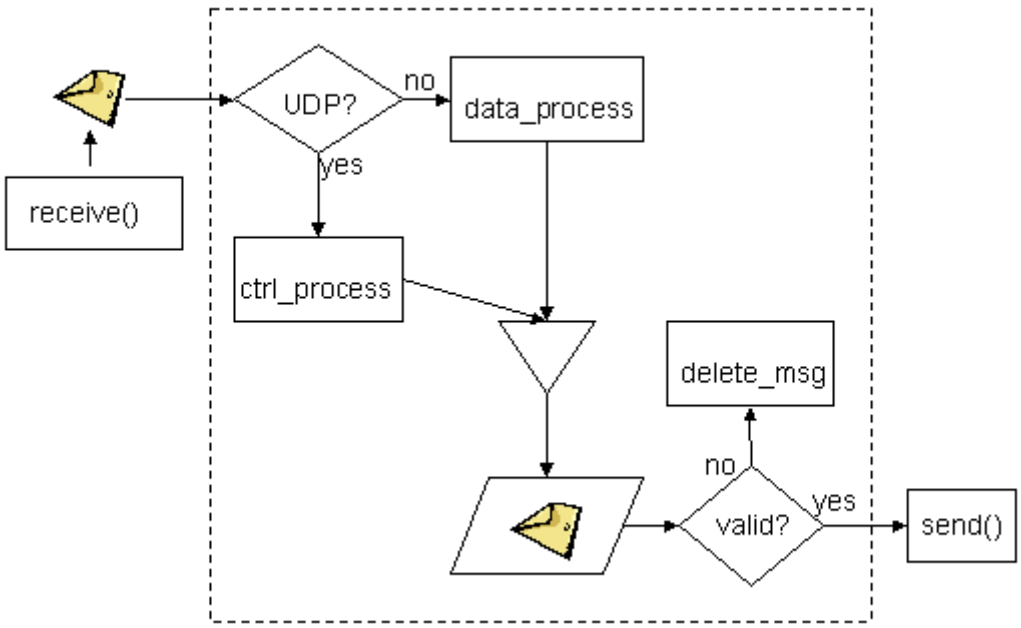

Figure 7.4-1 Message flow and processing in modules

The possible control messages received by the modules depend on their type. For the possible messages concerning the standard Mobile IP agent, please refer to [RFC2002]. For the VMN-agents, the possible messages, which are to be processes by the control process, are presented in Figure 7.4-2.

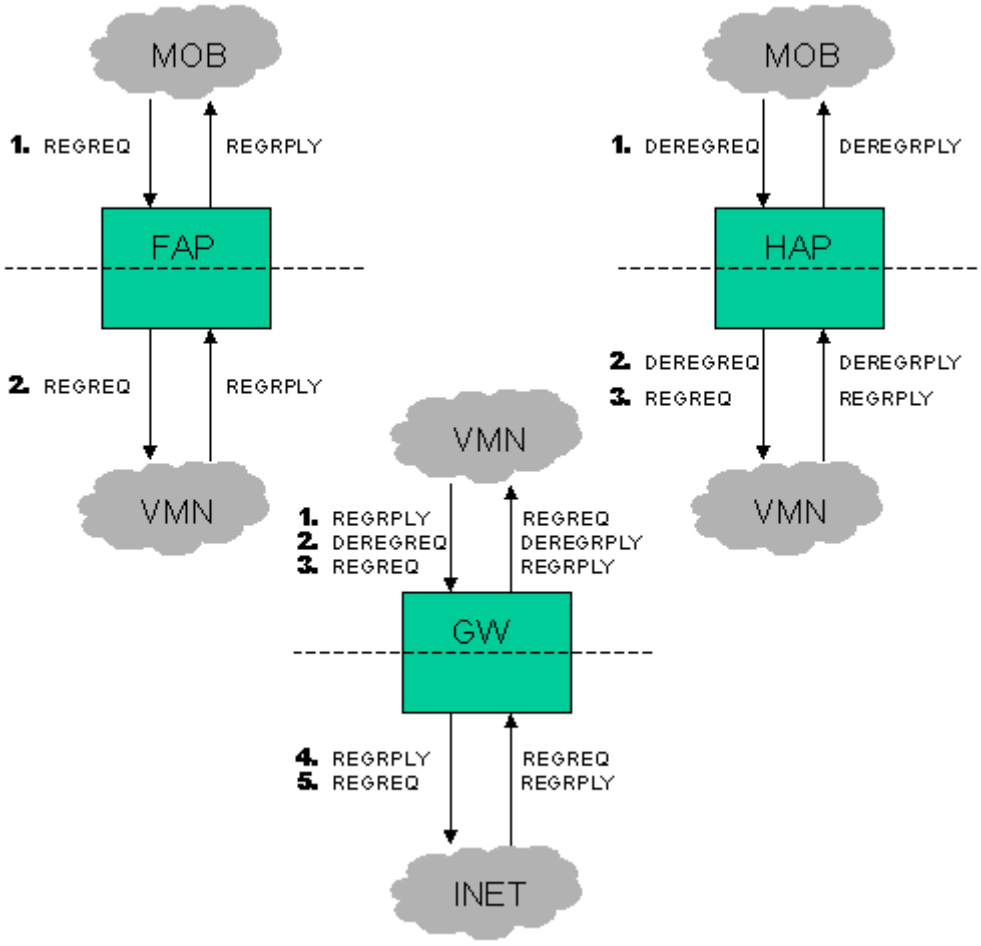

Figure 7.4-2 Possible control messages for VMN agents

Figure 7.4-2 shows the three types of VMN agents in their typical topological configurations. Each FAP instance is connected to both VMN, i.e. the local network containing other VMN-agents, and to the mobile nodes (MOB) represented by the mobile generator. In the same manner, each HAP is relaying messages between the mobiles (MOB) and the local network (VMN). The central gateway (GW) is an exchange point between the Internet backbone (INET) and the local network (VMN).

The exact scenario explanation for the cases in which these messages occur can be found in Table 7.4-1. In this table, REGREQ is the abbreviation for a Registration Request Mobile IP control message, REGRPLY – for the Registration Reply Mobile IP control message as explained in Chapter 2.3. DEREGREQ and DEREGRPLY are usual request / reply messages with a requested lifetime of zero. These represent de-registration requests and replies of the MNs returning home according to [RFC2002].

FAP	1	REGREQ of the foreign MN and corresponding REGRPLY of the HA
	2	Forwarded REGREQ and received REGRPLY from the HA for a foreign MN
HAP	1	DEREGREQ of an own MN and corresponding DEREGRPLY from the GW
	2	Forwarded DEREGREQ and corresponding DEREGRPLY of the GW
	3	REGREQ of an own absent MN from the GW and the generated REGRPLY
GW	1	REGRPLY from an HAP for an own absent MN and the prior REGREQ
	2	DEREGREQ of an own MN returning home and the generated DEREGRPLY
	3	REGREQ of a foreign MN and the REGRPLY received later from its HA
	4	REGRPLY of a HAP forwarded to the own absent MN and its prior REGREQ
	5	The REGREQ of a foreign MN forwarded to its HA and its REGRPLY

Table 7.4-1 Message s received and sent by the VMN-agents as shown in Figure 7.4-2

7.5 Databases in the simulation

As already mentioned in Chapter 5.4 there are some data structures in the simulation, which represent MN registration databases in the Mobile IP nodes. Those databases have to fulfill some requirements like dynamic size adjustment, fast explicit element lookup and element search / add / remove.

The elements are the mobile nodes, which are represented by their database records. These records basically consist of the home IP-address of the mobile node (`MobEntry::IP`) and some management information like lifetime of the registration (`MobEntry::lifetime`) and the point of time of the last registration. Since it was finally decided to store all mobile nodes in one structure instead of managing two different databases, the records include a temporary flag, which is set for the MNs in registration and unset for the registered ones. In this manner, we do not need to copy the entries after a successful registration. However, the exact record type depends on the kind of this record from the point of view of the managing instance. So, we have to save different additional information for the own and for the foreign mobile nodes. For the network's own absent MNs in the simulation, the most important information is the current CoA (`HMobEntry::CoA`). For the foreign mobile nodes we have to save the source IP and port from which the last registration request came and the address of the responsible home agent. Apart from that, in the VMN we also have to save the source coordinates due to the new detours as explained in Chapter 6.1.

For that reason, the class model shown in Figure 7.5-1 has been used within the simulation, using a base entry class `MobEntry` and sub-classing the remaining classes from this common basis. `FMobEntry` represents the entry used in the foreign agent instances to save foreign mobile nodes. Similarly, `HMobEntry` represents the database

entry for the own absent mobile nodes and the `VMNHMobEntry` the same entry in a FATIMA network. The base class overloads the equality operator making the mobile node IP address the primary key. So, two records are considered the same if they include the same MN address (`MobEntry::IP`).

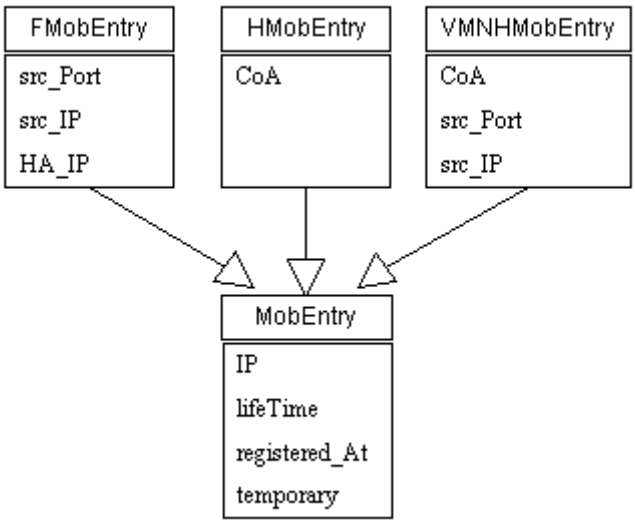

Figure 7.5-1 MN database entries

The most frequent queries for these databases are the following:

- Add an element to the database
- Find and return the data record for the given IP
- Is this IP in the database?
- Remove the element from the database

Analyzing these requirements one can see that a hash table hashing IPv4-addresses would be perfectly suited to process these queries, since all these operations would run in *O(1)*. Finally, the hash table template presented in Chapter 7.6 was used.

A hash function for the IPv4-addresses was a little problem. Basically, an address in this format is an unsigned 32bit-integer, so we could use a generic modulo operation. However, all the addresses within one network are likely to come from the same range (being divided into sub-nets) and using two different networks with exactly the same host address parts, we could be unlucky and always meet the same remainder for the corresponding host addresses in these networks. So, it's better to accentuate the last byte of the IP address always pointing to the host and perhaps the second byte pointing to the used (sub-)network. Finally, the so-called Van Jacobson hash function for IPv4-addresses from the [FreeBSD] TCP/IP stack implementation was used. After slight modifications for our purposes the function has finally been implemented in that form:

$$hash(ip) = ip \otimes (ip \cdot 2^{23}) \otimes (ip \cdot 2^{17}) \wedge \neg(size - 1) \bmod (size - 1),$$

where:

ip: IP address to be hashed
size: size of the hash table (should be a power of 2 value)

7.6 Generic hash table template

To meet the requirements described in Chapter 7.5, an independent data type, a generic hash table C++-template for the elements of some class T, has been developed. This template implements a hash table with *linear chaining*[1]. Being implemented as a combination of an array and a list internally, it dynamically adjusts its user-defined initial size if needed. The user can also freely specify the hash-function. Additionally, the hash table supports a so-called handler function, which will be automatically called for the both related elements each time when the user tries to insert a new element, which is considered equal to an already contained element. Apart from that, an element stream and a small statistics function returning results about the occupation situation are provided.

The template is represented by the `template <class T> class Hashtab` and is shown in Figure 7.6-1. The constructor needs the initial size of the new table and a pointer to a hash function. The hash function could be any function returning an integer hash value for an instance of the element of type `T`. The value range will be proven and truncated as needed (by *'modulo table_size*'). The elements are inserted using the overloaded `operator<<`. The `T* search(T& a)` function will return the pointer to the element `T` or `NULL`, if an element equal to `a` (i.e. `element == a`) cannot be found.

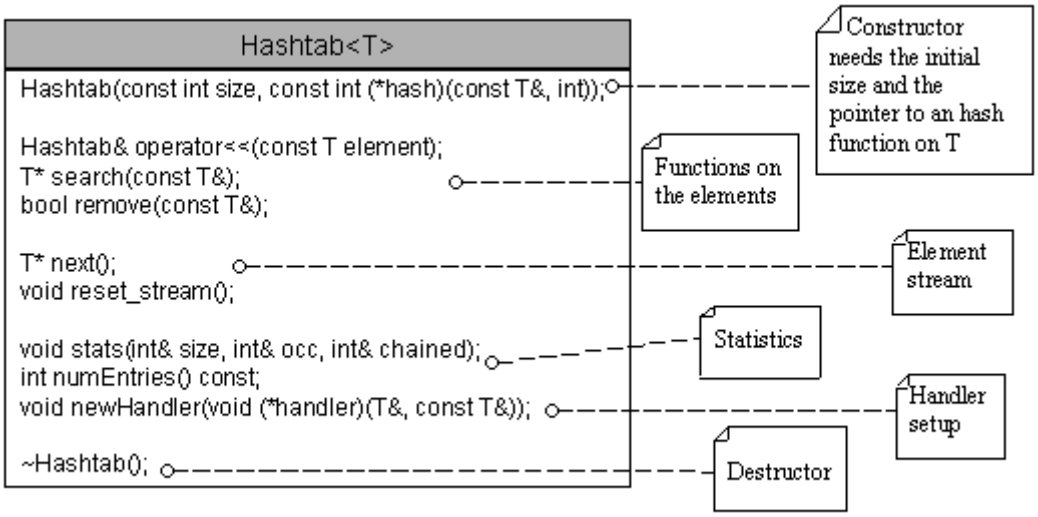

Figure 7.6-1 Public routines in `Hashtab<T>`

A default constructor for class `T` is required; this is used for building new instances within the table. For some more complicated type `T` the user should overload the equality operator ('=='), which is always used to determine whether two elements are equal. The assignment operator ('=') should be overloaded if the normal C++ operation (just copy the members) is not suitable. The '=' will be used to copy the element-

[1] *Linear Chaining* is a conflict resolution method which is used when two different elements have the same hash value

contents into the table structure. Nevertheless, for basic types the existing definitions work just fine. User can specify the handler-function as explained above using the `newHandler(…)`–call.

7.7 Summary of main classes

As already explained in Chapter 7.2, most main classes are sub-classed from the `RootClass`. The complete hierarchy of the `RootClass` descendants is illustrated in Figure 7.7-1. You can easily recognize the main sub-tree beginning with `Node` which has already been explained in Chapter 5.4. The only new class in this sub-tree is `SimpleHost` which represents a classical Internet host without mobility support and which is used to simulate free correspondent nodes. In the same way, the sub-tree beginning with `Mobile` class has been explained in Chapters 5.6 and 5.7.

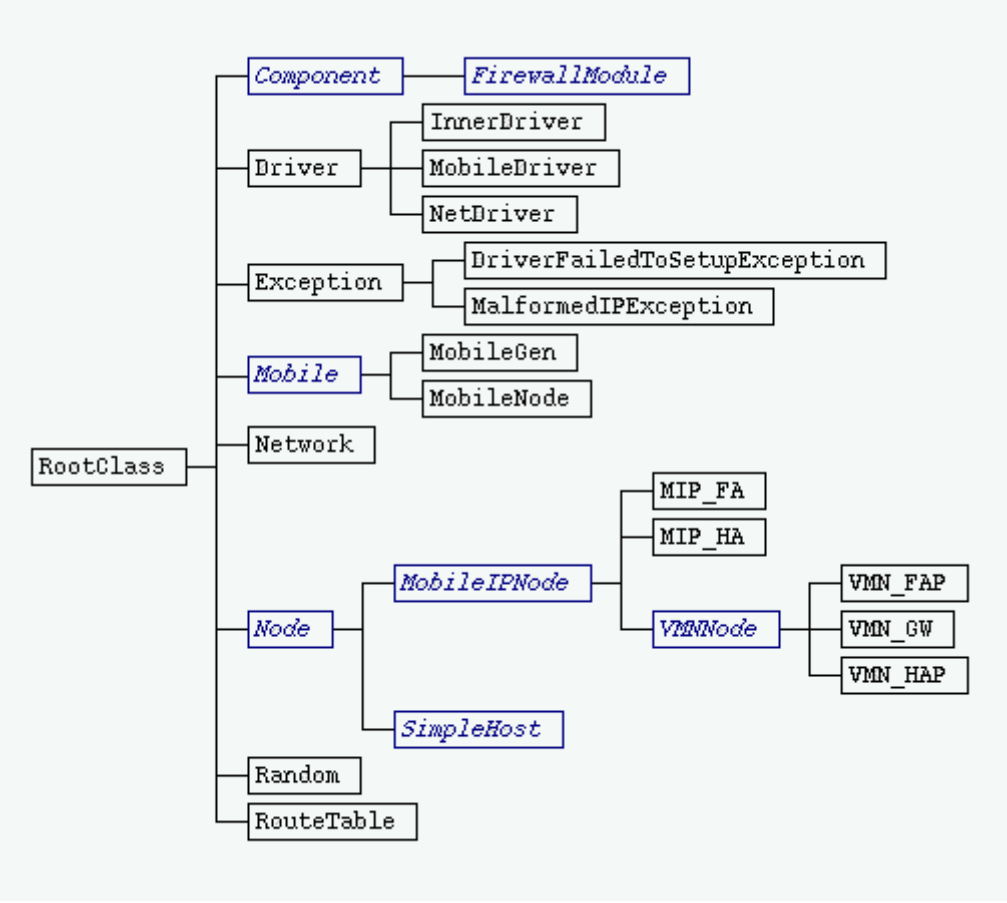

Figure 7.7-1 Complete hierarchy of the main classes

The module types represented by `Mobile` and `Node` classes use the respective driver from the `Driver` sub-tree. The `InnerDriver` is used by the so-called `Component`-modules. The `Component` class has been developed in order to be able to distinguish between the modules connected to the network and the inner sub-modules of the more sophisticated hosts. In fact a component could represent any module, which is not participating in the simulated networking. E.g. the central gateway could be designed as a combination of three sub-modules, a cryptographic sub-module, a traffic shaper and the core process. To be able to record the simulation results per inner module i.e.

component, this concept has been developed. The components can also send messages (for visualization or simulation purposes). These messages are instances of `InternMessage` class and can be sent through the `InnerDriver`. An example for a component interface (`FirewallModule`) has been written but not used.

The `Exception` sub-tree includes the `Exception` interface and some instantiations for it used by `Driver` and `RouteTable` classes. The `RouteTable` class represents a routing table used by every network node. It also defines the functions to transform an IP formatted string type into a numeric IP value and vice versa which is used in modules for reading their parameters. The `Random` class is a pure wrapper class using the OMNeT++ random generators and distributions internally. It is a part of the *Abstraction API* as described in Chapter 5.3. Finally, the `Network` class implements the concept described in Chapter 5.5. It additionally uses the OMNeT++ API directly, i.e. no driver concept is applied here. Instead of that it implements the `cSimpleModule` interface. That is comprehensible because this class is explicitly designed to solve the problems concerning OMNeT++ as broadcast emulation and IP-routing.

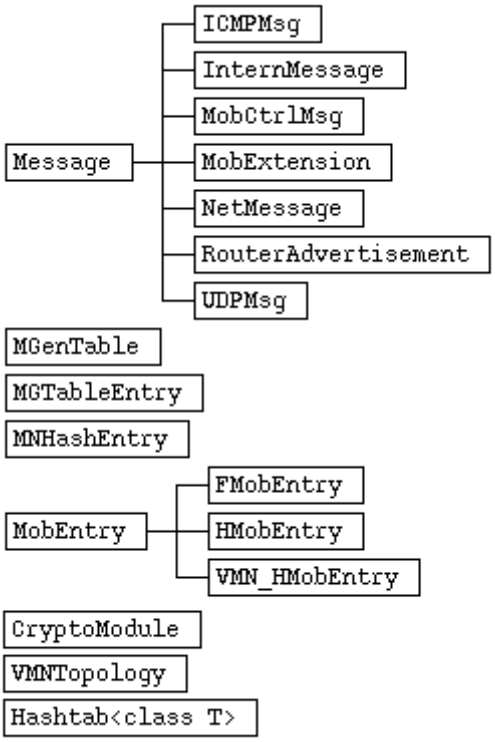

Figure 7.7-2 Additional used standalone classes

The additional classes are either `Message`-descendants or simple database records or independent containers or databases as instantiations of these containers with one of the records as shown in Figure 7.7-2. `Message` descendants and database records were not sub-classed from the `RootClass` in order to keep them lighter to improve overall simulation performance. The implemented `Hashtab` container was not sub-classed from the `RootClass` because it can be used stand-alone as e.g. the common STL containers. The `CryptoModule` class which has not yet been mentioned will be explained in Chapter 7.8.

7.8 The `CryptoModule` class

The only exception to the scheme explained in Chapter 7.7 is the `CryptoModule` class (→Figure 7.7-2), which was initially intended to implement the `Component`-interface. However, it was finally implemented as an independent class whose constructor takes the pointer to the owning simple module. This class represents the abstraction of the cryptographic functions used in the real host causing the delay in the caller-module, which would occur during the computation if the cryptographic functions had really been called.

This module in its current form is much less complicated than it could be as a real component (representing e.g. a hardware unit in a FATIMA gateway). It does not gather any statistical information itself. Instead it proposes several functions, which simulate the required load (i.e. the delay). The delay times for the simulated functions have been measured under the test environment presented in Table 7.8-1.

CPU	PII, 300MHz
OS	SuSE Linux 6.4
Library	OpenSSL 0.9.5 (SuSE-Paket openssl-0.9.5-31)

Table 7.8-1 Test environment for the delay times of cryptographic functions

All used delay values have been obtained with the Unix-command "`time`" measuring the user-mode CPU time. The delays for the encryption functions were measured as a result of the encryption of an overall amount of 100 MB with the respective function, in portions of 100/200/300/400/500 byte. As constant keys were used, no key scheduling is included in the obtained delay times. For the hash functions, an overall amount of 1000 MB of data was used, divided into the same data portions. All delay times were measured and kindly provided by Frank Pählke from Institute for Telematics, University of Karlsruhe.

8 Simulation results

The results presented here should be understood as the proof of the feasibility of the FATIMA project once the conceptual problems described in Chapter 6 have been solved. On the other hand they also provide the proof for the usability of the implementation developed during this work.

In order to obtain reliable results delivering insights in the comparative performance of FATIMA vs. standard Mobile IP, a lot of additional runs and scenarios had to be tested. Apart from that, the used simulation constants and handover timings and distributions should be understood as propositions since their values describe special situations. Anyway, there are hardly any successful approaches to simulate Internet or complex mobility schemes in general →[Paxs et al. 97]. So, it is possible that the usage of other values could provide contradictory results.

Several different parameter types influence the simulation behavior. Most of these parameters are combined in the `RootClass.h` file where they can easily be changed. This includes values for handover timings and distributions of the mobile nodes, the data flow parameters for the data traffic and the used typical message lengths. Moreover, certain additional features can be activated and deactivated there, like e.g. Reverse Tunneling (→Chapter 2.6) or Fast Handoff Extension (→Chapter 3.6).

The next important place for changes of the simulation behavior is the topology description. Certainly, the timings and paths of the packets and therefore the recorded results depend on the types of the involved networks (FATIMA, Mobile IP), on the defined channel types (bandwidth, delay, etc.) and the number and membership of the mobile nodes. All these settings can be made in the NED files. The NED parameters, which will be changed more often, can be set in the INI file (`omnetpp.ini`). For further information on how to configure and use OMNeT++ based simulation programs, please refer to [OMNeT Man].

Finally, another important place which influences the simulation timings, is the `CryptoModule.cc` file explained in Chapter 7.8. In this file you can change all the values concerning cryptographic functions and their performance.

8.1 Terms and explanation of the results

All results presented here have been gathered using the same test environment, a common value basis as presented in Table 8.1-1. If these values are changed in certain scenarios, we will explicitly mention it in the scenario description. The start seed value was changed for every run with the help of the "`seedtool`" program, which is part of the OMNeT++ distribution.

In the following we will talk about a *symmetric configuration*, if every involved network is configured with the same number of mobile nodes. Otherwise we will talk about an *asymmetric configuration*.

Usually, at least three possible main network topologies are tested, i.e. FATIMA-FATIMA, FATIMA-MobileIP and MobileIP-MobileIP. The fourth configuration (MobileIP-FATIMA) is used where it matters, as e.g. with an asymmetric configuration.

The typical configuration within the networks consists of one HAP / HA and two FAP / FA instances. This allows simulating both the local and global handovers. The number of mobile nodes is changed according to the wanted results.

Channel types			
	Local link	100 MBit/s	1ms
	Backbone link	800 kBit/s	200ms
	Mobile link	1 MBit/s	50ms
Mobile IP control settings			
	Registration Lifetime	5000s	
	Reregister safety	60s	
	Advertisement Interval	300s	
Handover values			
	Global HO probability	0.3	
	Local HO base probability	0.7	
	Average visit time (exp mean)	1000s	
	Global HO mean factor	6	
Data flow values			
	Send data	ON	
	Destination host	-	
	Maximum burst (max range)	100	
	Wait between bursts (exp mean)	2000	
Features			
	FHE	OFF	
	Reverse Tunneling	ON	

Table 8.1-1 Test environment usually used to gather simulation results

8.2 Scenario 1: Several MNs sending to a CN in the Internet

This scenario represents the case, in which different numbers of mobile nodes send packets to a fixed CN installed somewhere in the Internet. The CN answers with a simple answer packet. The length of the messages is randomly chosen. Having received the reply data message, the MN records statistical information on the overall time period passed since message sending. In this way it records the round trip delay (RTD) to this CN.

The test environment was exactly as illustrated in Table 8.1-1. We decided to use the asymmetric Mobile Node configuration, in the meaning that all MNs were members of one analyzed network. The other network was configured with no mobile nodes. That is why all four topology possibilities have been tested (as explained in Chapter 8.1).

The simulation time for each run was 14 days. In the first step we tested 10 mobile nodes per topology, in the second 60 and in the last step 150 mobile nodes. However we decided to randomly choose some mobile nodes and not to try to represent the behavior of every involved mobile node.

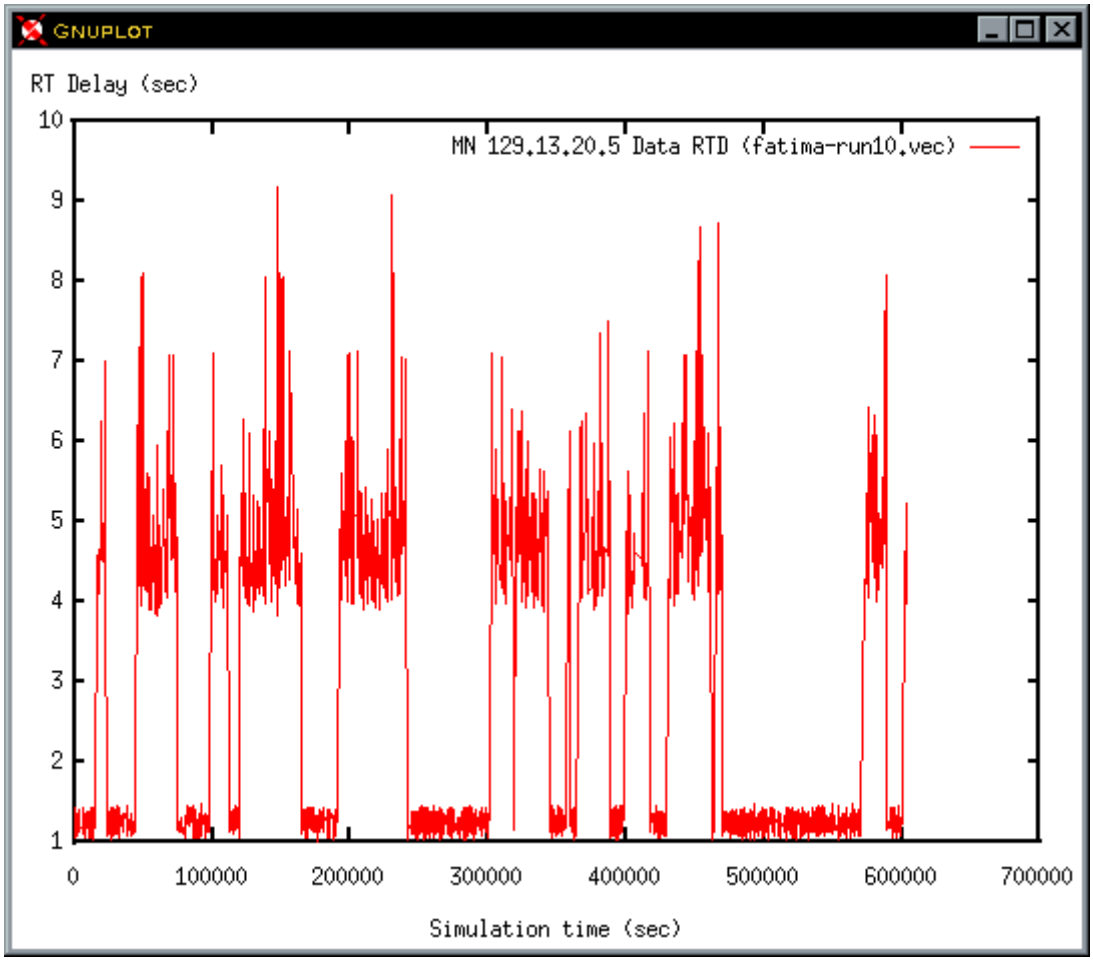

Figure 8.2-1 RTD: MN-neutral CN, FATIMA-FATIMA, 10 MNs (MN@FAT)

Figure 8.2-1 illustrates the RTD measured by a mobile node originating from a FATIMA network in a FATIMA-FATIMA configuration with 10 MNs. The two axes shown in Figure 8.2-1 are simulation time in seconds (horizontally) and the round trip delay in seconds (vertically). One can easily recognize the typical oscillating run of the

curve. At the beginning, when the MN is at home and does not use any mobility support, the delay remains low. The FATIMA system is not involved in delivering the packets from or to the mobile node. Then, after a global handover, the delay rises rapidly since the mobility support is switched on. The exact value of the delay is not that important since it highly depends on the used test environment. Quantitatively, only the comparative results are considered interesting. Being at the foreign network, the MN sends the packets to the CN using Reverse Tunneling. Additionally, the FATIMA system obliges to make several cryptographic operations, both for own and visiting Mobile Nodes, so the delay rises. Within the FATIMA system, packets are sent through ESP channels; that increases the overall delay one more time. This delay remains at the high position during MNs absence at home and falls back to the lower state after its return.

In the following we can see the same situation from the point of view of a mixed topology and a standard Mobile IP system. Please note, that the Mobile IP system does not use any cryptographic checks for the data packets at the moment. This results in lower measured delays.

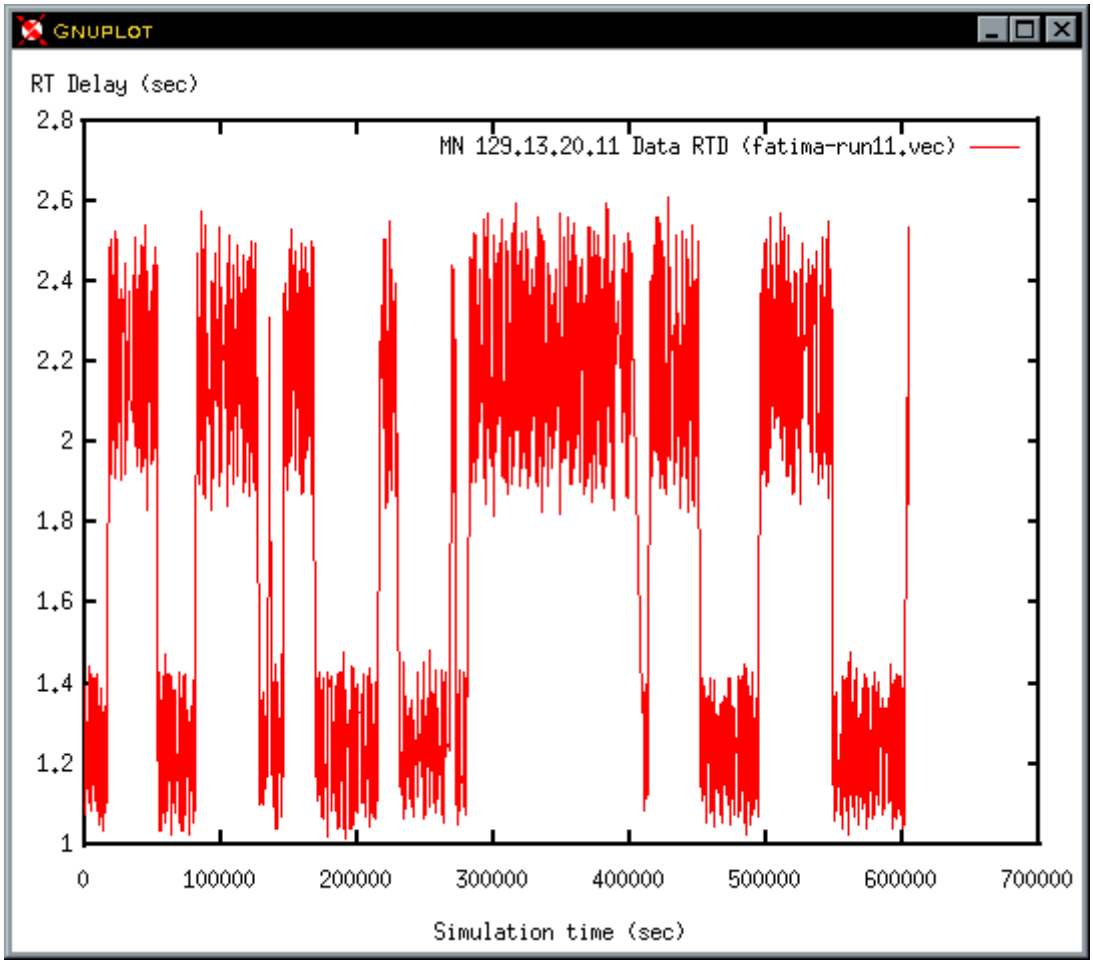

Figure 8.2-2 RTD: MN-neutral CN, FATIMA-MobileIP, 10 MNs (MN@FAT)

Figure 8.2-2 represents this situation for the MN from a FATIMA network in the mixed topology while Figure 8.2-3 represents the same situation for the MN from a Mobile IP network in the mixed topology.

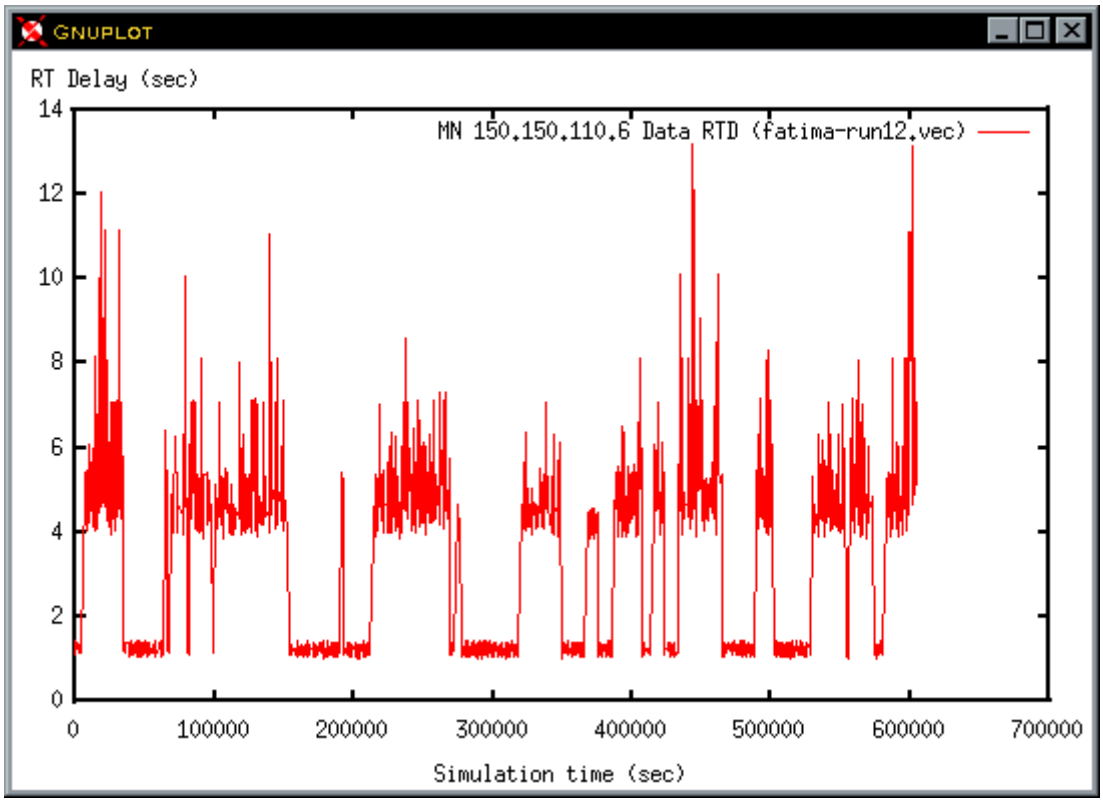

Figure 8.2-3 RTD: MN-neutral CN, MobileIP-FATIMA, 10 MNs (MN@MIP)

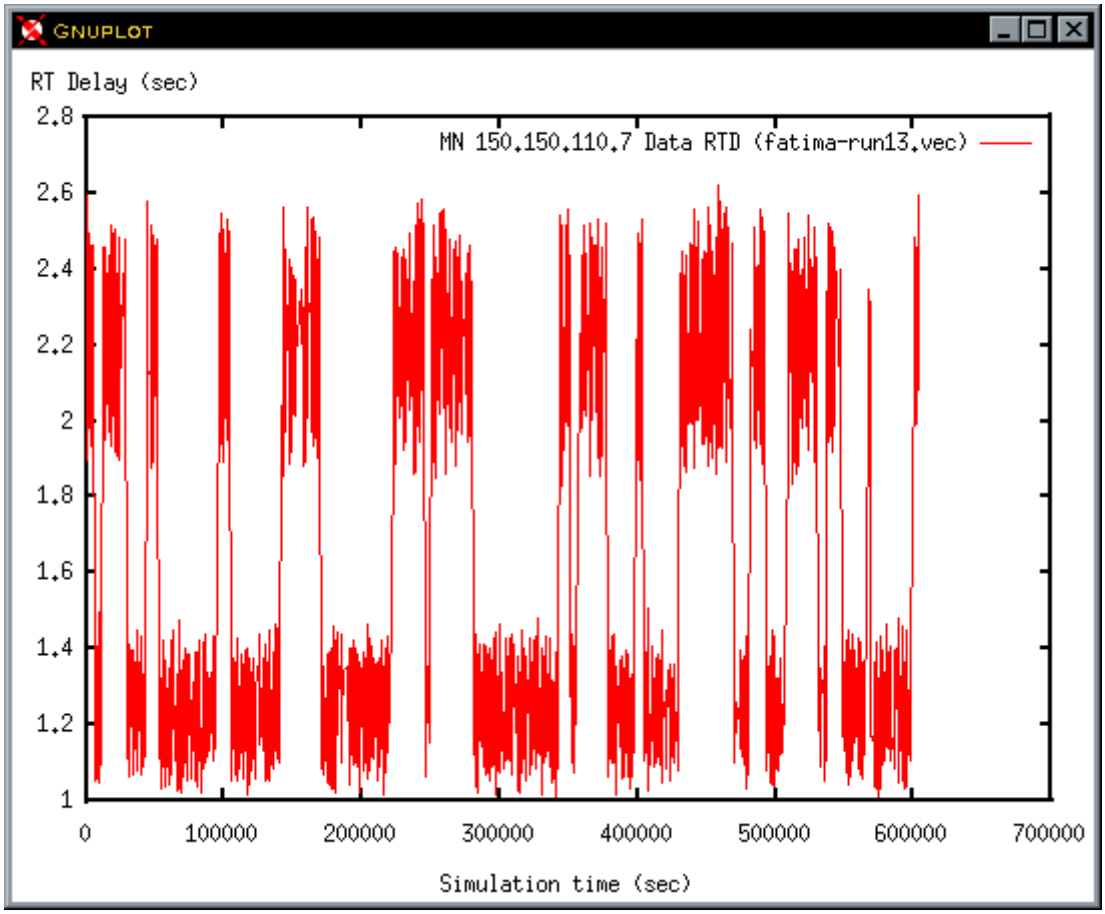

Figure 8.2-4 RTD: MN-neutral CN, MobileIP-MobileIP, 10 MNs (MN@MIP)

In the following illustrations we can see the same situation presented for 60 MNs.

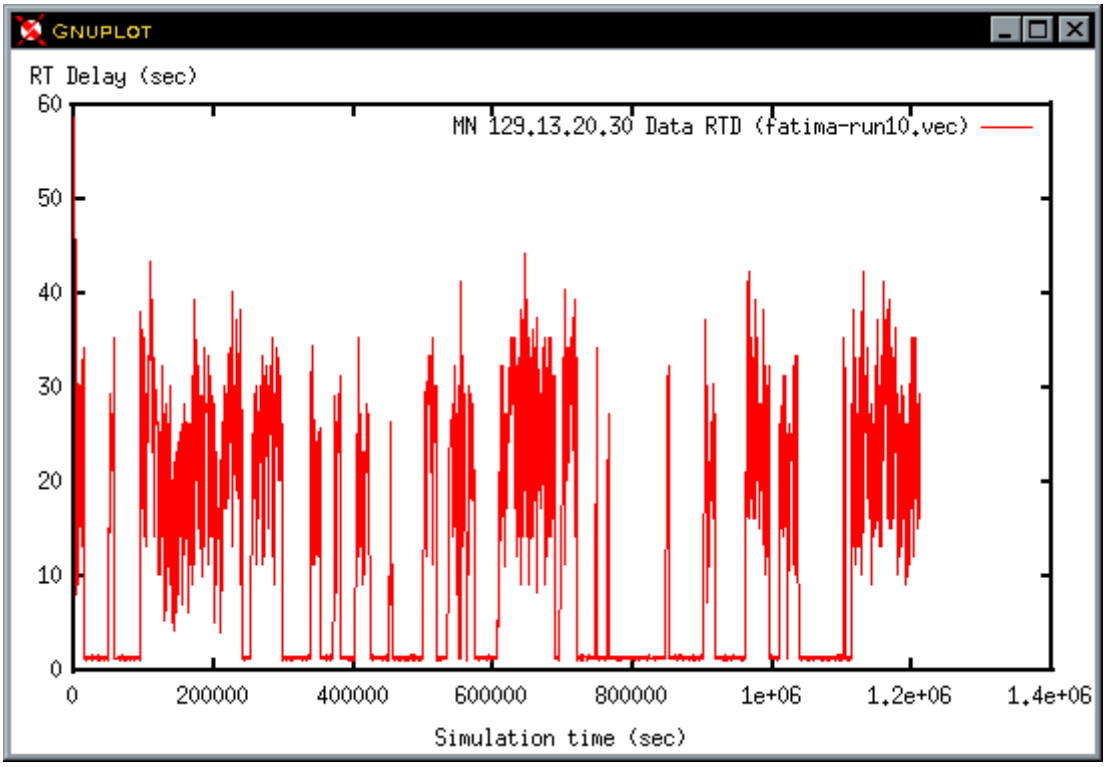

Figure 8.2-5 RTD: MN-neutral CN, FATIMA-FATIMA, 60 MNs (MN@FAT)

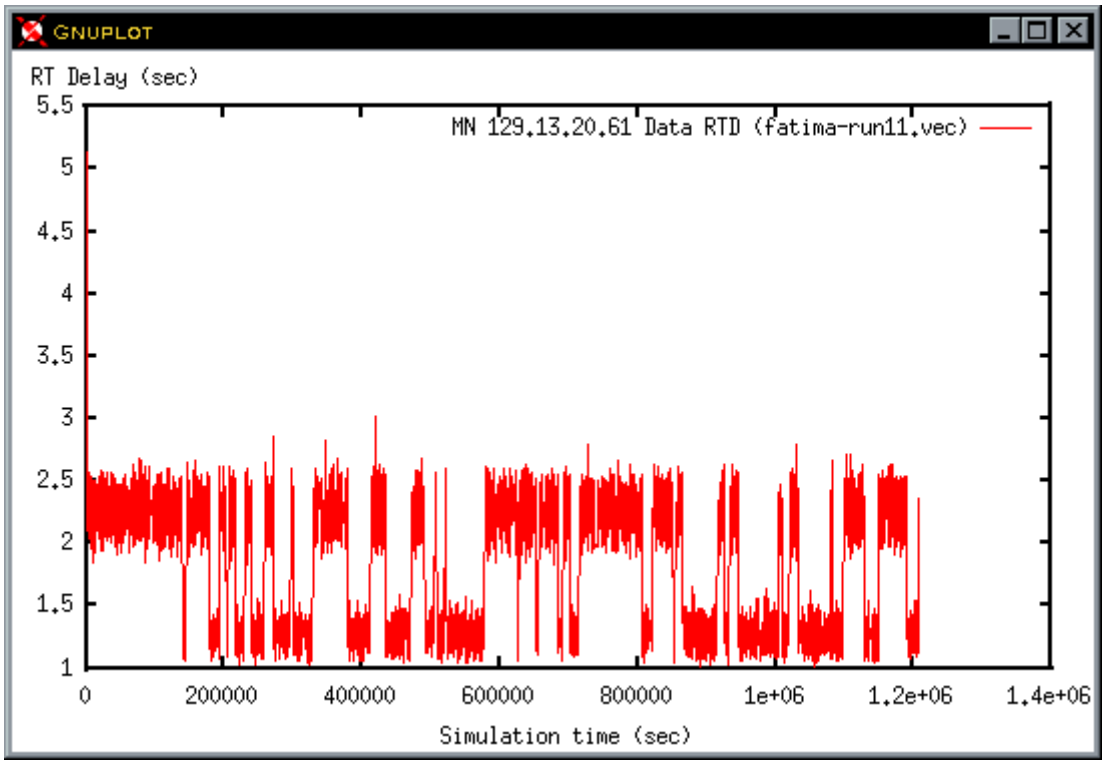

Figure 8.2-6 RTD: MN-neutral CN, FATIMA-MobileIP, 60 MNs (MN@FAT)

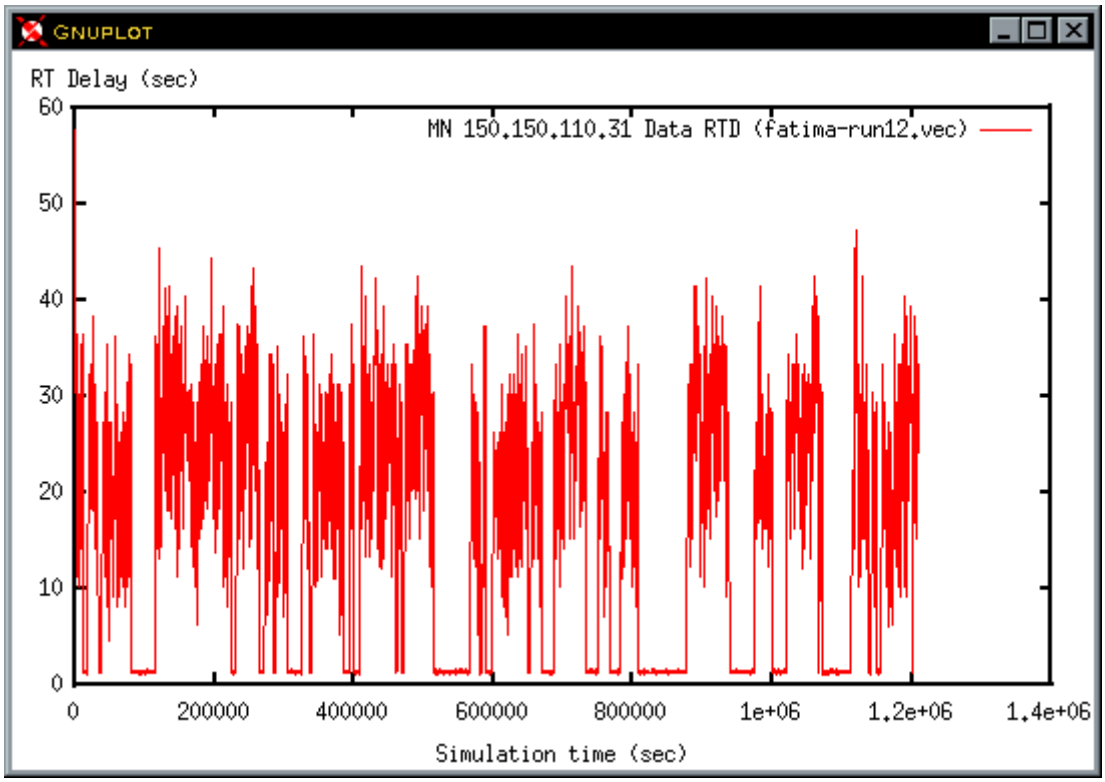

Figure 8.2-7 RTD: MN-neutral CN, MobileIP-FATIMA, 60 MNs (MN@MIP)

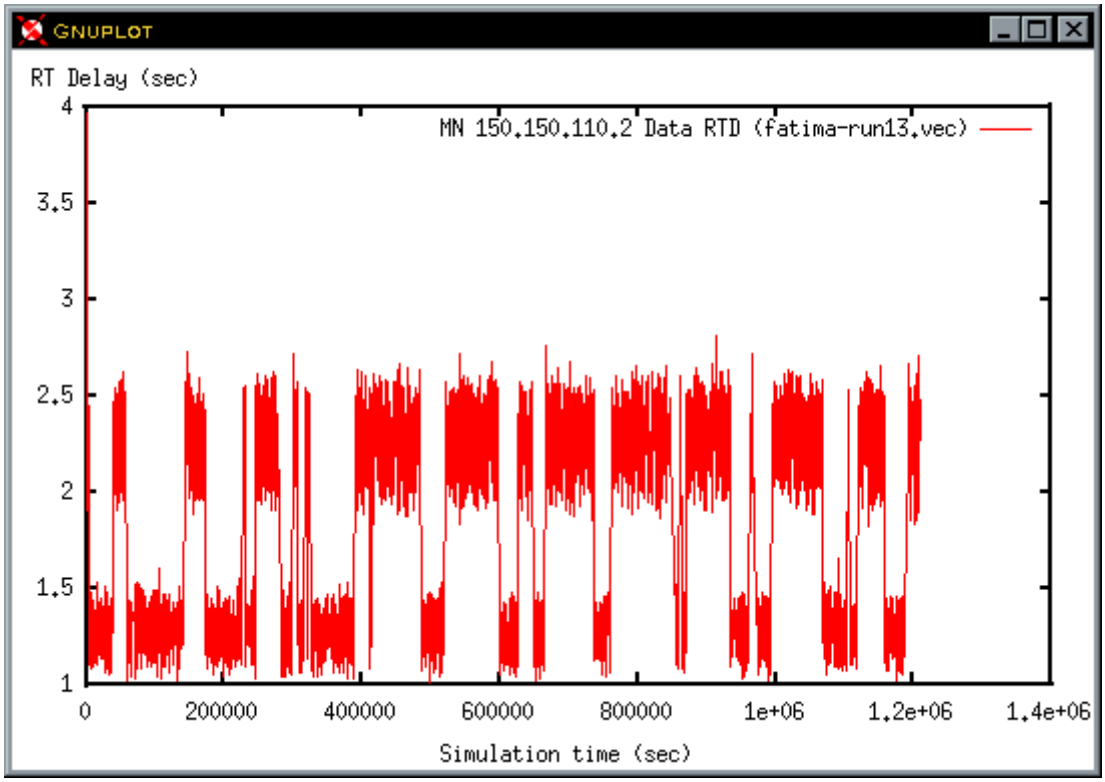

Figure 8.2-8 RTD: MN-neutral CN, using MobileIP-MobileIP, 60 MNs (MN@MIP)

And finally, here are the results for the same situation with 150 MNs per topology.

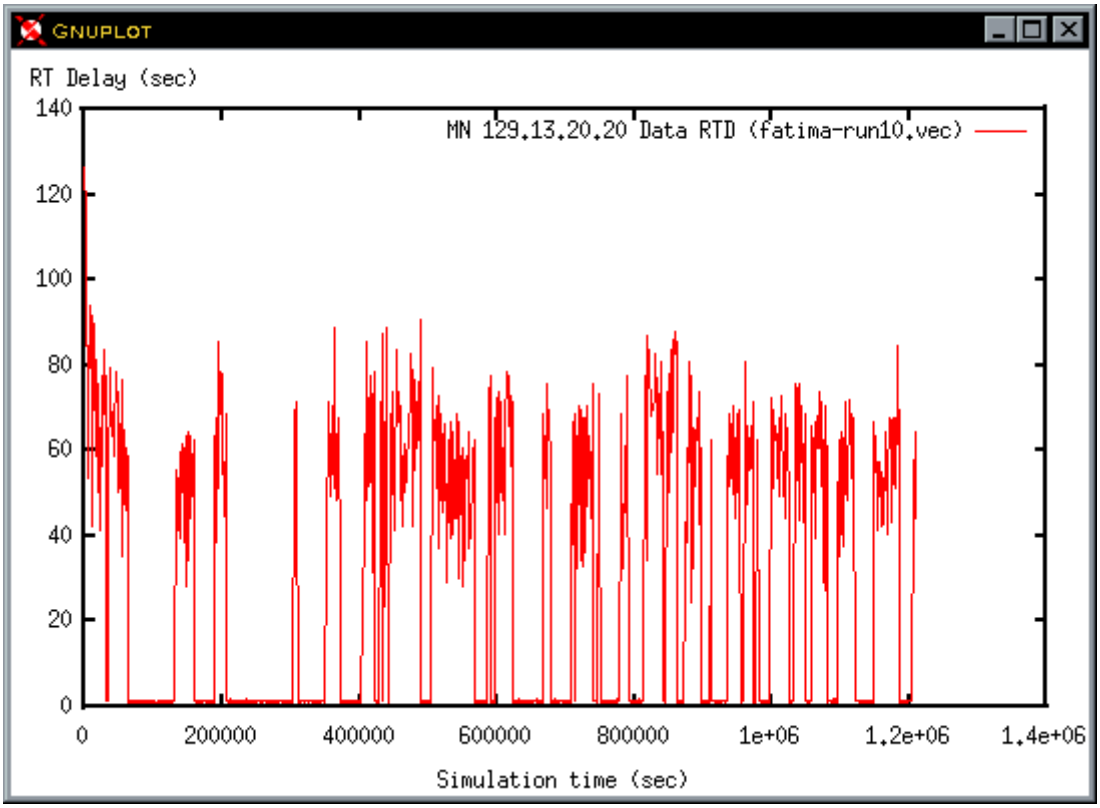

Figure 8.2-9 RTD: MN-neutral CN, using FATIMA-FATIMA, 150 MNs (MN@FAT)

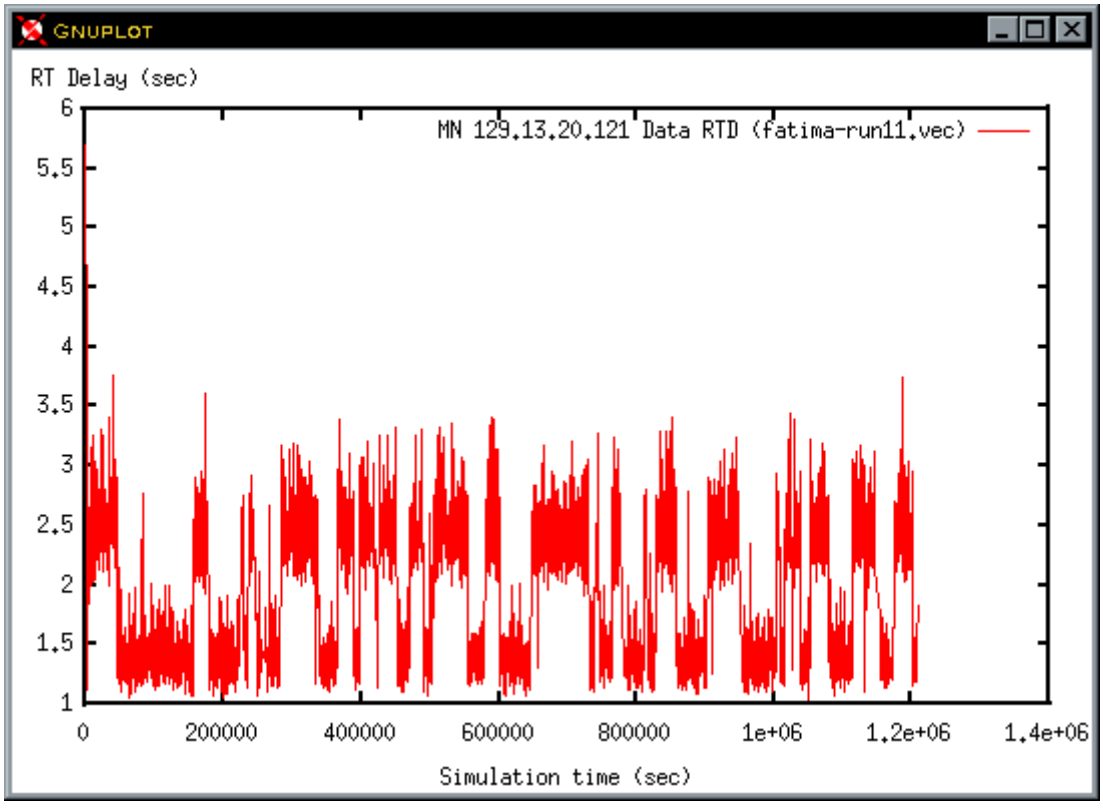

Figure 8.2-10 RTD: MN-neutral CN, FATIMA-MobileIP, 150 MNs (MN@FAT)

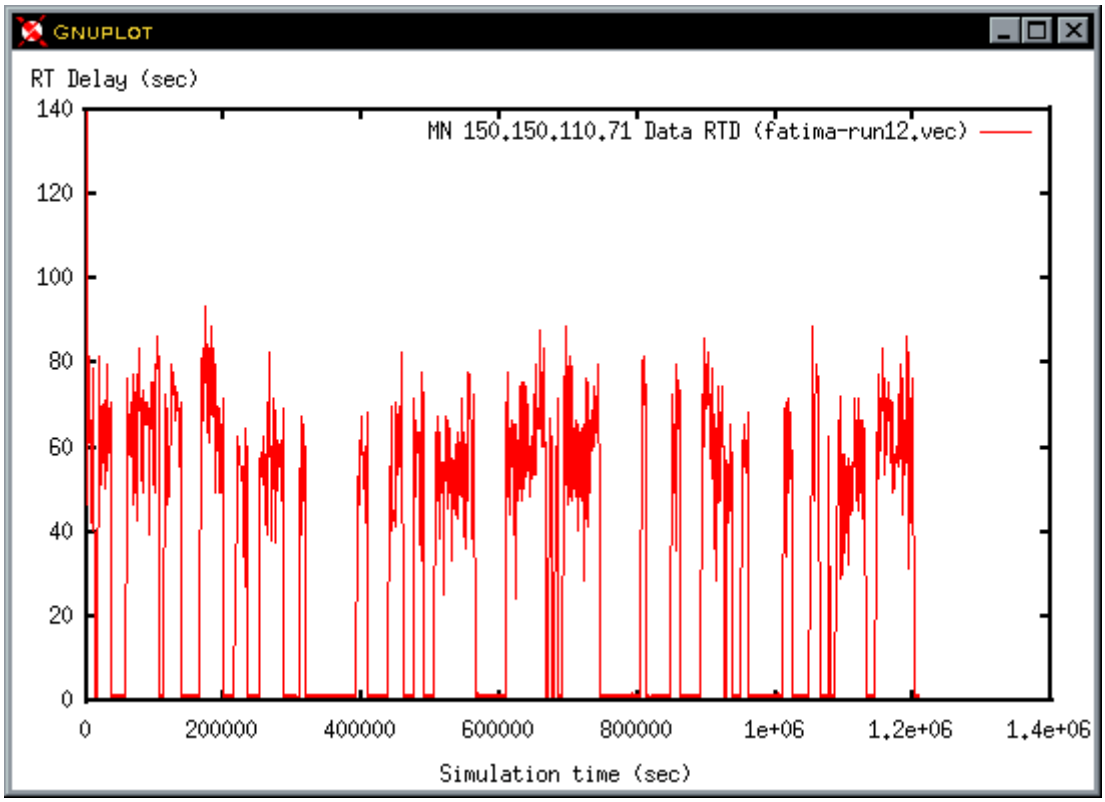

Figure 8.2-11 RTD: MN-neutral CN, MobileIP-FATIMA, 150 MNs (MN@MIP)

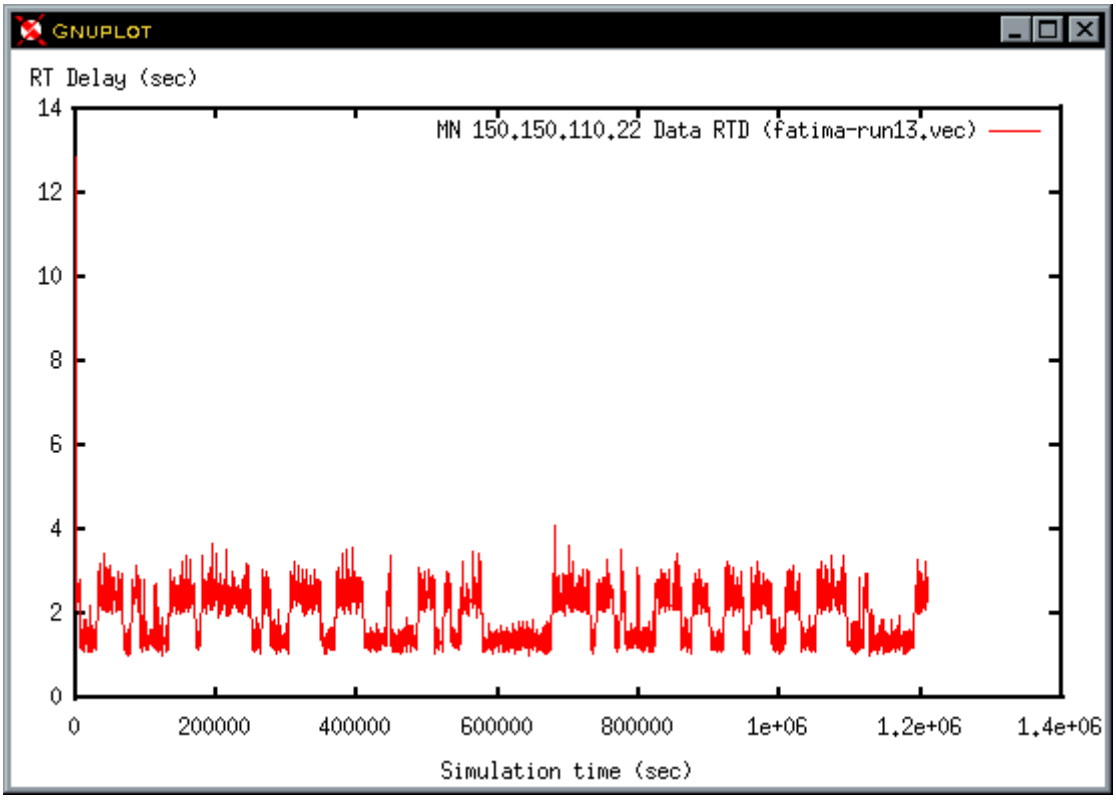

Figure 8.2-12 RTD: MN-neutral CN, MobileIP-MobileIP, 150 MNs (MN@MIP)

Comparing Figure 8.2-1, Figure 8.2-5 and Figure 8.2-9, we can easily see how the round trip delay arises when using a FATIMA system according to the additional load produced by the increased number of the mobile nodes.

A similar effect can be illustrated in the mixed topologies. In those mixed topologies where the MNs home is a standard Mobile IP network (Figure 8.2-7, Figure 8.2-11), the round trip delay falls back to the minimum. However, in the ones where MNs home is the FATIMA-equipped network (Figure 8.2-6, Figure 8.2-10), after the MNs returns home, the round trip delay decreases noticeably but still remains much higher than in the same situation with a Mobile IP network (compare Figure 8.2-6 and Figure 8.2-7 for 60 mobile nodes or Figure 8.2-10 and Figure 8.2-11 for 150 mobile nodes). That is due to the high load on the central gateway, which has to forward the messages of the MNs even if they are not using any mobility support.

In the pure Mobile IP topologies, this problem does not exist since the load on the HA hardly noticeably increases because of the data traffic (it's not encrypted and therefore not measured). Only the different routing is responsible for the higher delay here. The overall RTD remains at almost the same level of about 2.5 – 3 sec, see Figure 8.2-4, Figure 8.2-8 and Figure 8.2-12.

8.3 Scenario 2: MN sending to a CN in its home network

This scenario represents the case in which the mobile node sends data packets to a correspondent node statically installed in MN's home network. Four topology configurations have been used due to the asymmetrical configuration. Only the network with the CN was configured to have mobiles. An overall number of 60 mobile nodes was used. The simulation time was set to 14 days.

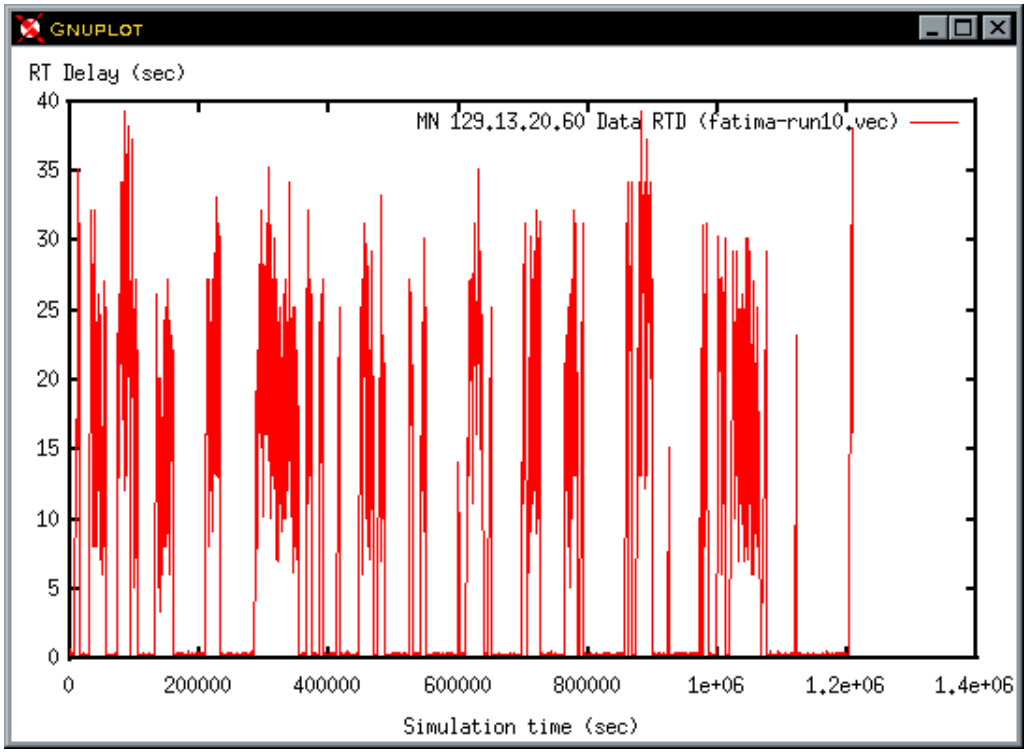

Figure 8.3-1 RTD: MN sending home, FATIMA-FATIMA, 60 MNs, MN@FAT

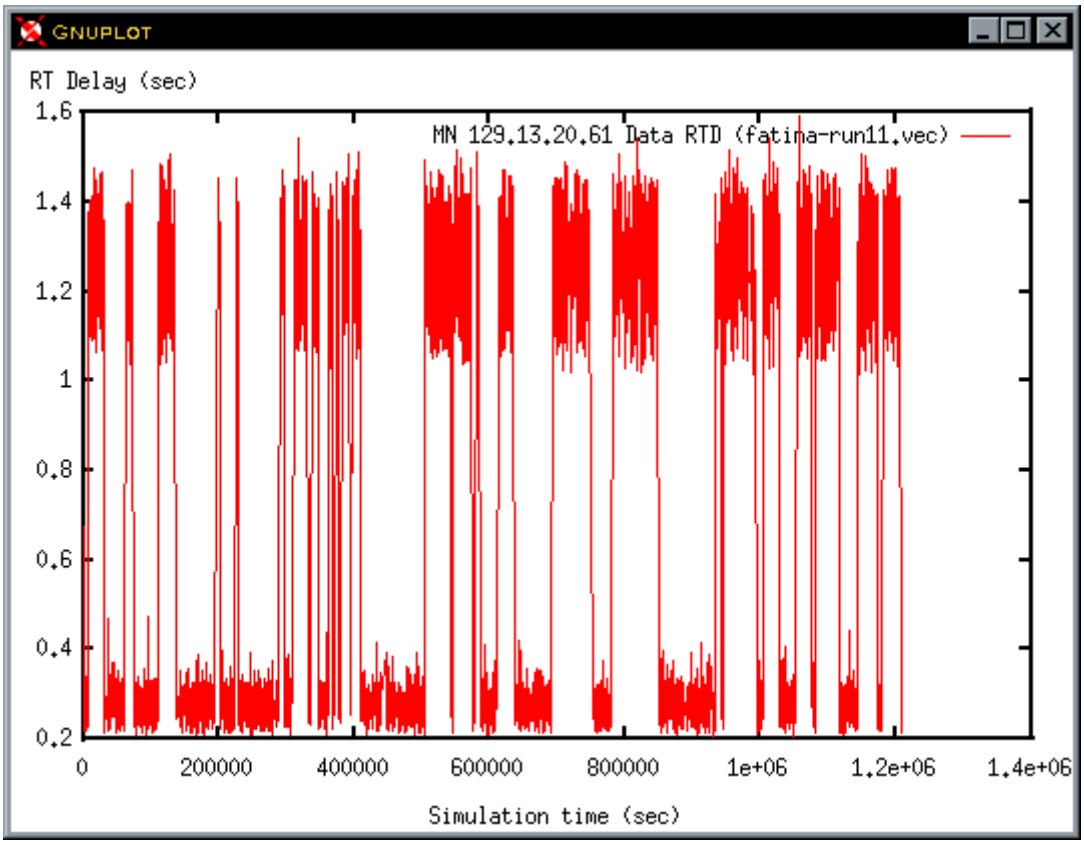

Figure 8.3-2 RTD: MN sending home, FATIMA-MobileIP, 60 MNs, MN@FAT

Taking a look at Figure 8.3-2, we see the same typical oscillating RTD-behavior in the second case, which we have already seen in Chapter 8.2. Being at home and sending data to a CN in its home network, MN basically uses local network connections. That explains the remarkably low round trip delay values at the bottom of Figure 8.3-2. After a global handover to a foreign net (here: MobileIP network), the round trip delay becomes higher, what is evident at the top of the diagram. That is due to the additional backbone use (while contacting home network from outside).

In the Figure 8.3-1 we can observe the same extremely low RTD values when being in the home network. Otherwise we observe that the RTD is increasing. That is the right simulation behavior since the packets take their way through both simulated FATIMA systems: through the responsible FAP over the ESP-channel to the foreign GW and further over the backbone to the local GW and from that over the ESP-channel to the responsible HAP which then can give the packet to the local CN. The reply packet is intercepted and sent by the HAP over exact the same way back to the absent MN doubling the RTD.

In the Figure 8.3-3 we can see almost the same situation. A MN being at home in some standard Mobile IP network contacts its CN directly over the local links. After a global handover it registers from a FATIMA network, from which it then sends its data packets over the FAP, ESP-channels and the central GW being loaded by the data traffic of the involved mobile nodes.

In Figure 8.3-4 we find a similar (and very typical) situation as in Figure 8.3-2, because the foreign network is equipped with standard Mobile IP.

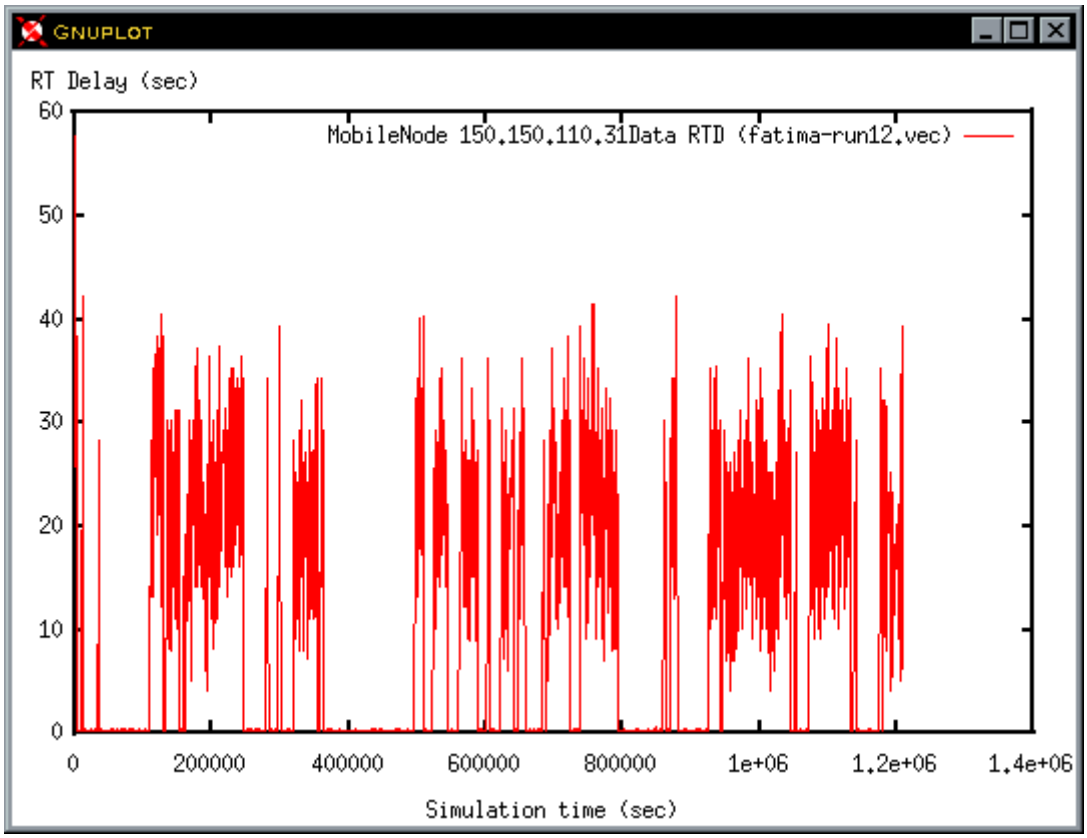

Figure 8.3-3 RTD: MN sending home, MobileIP-FATIMA, 60 MNs, MN@MIP

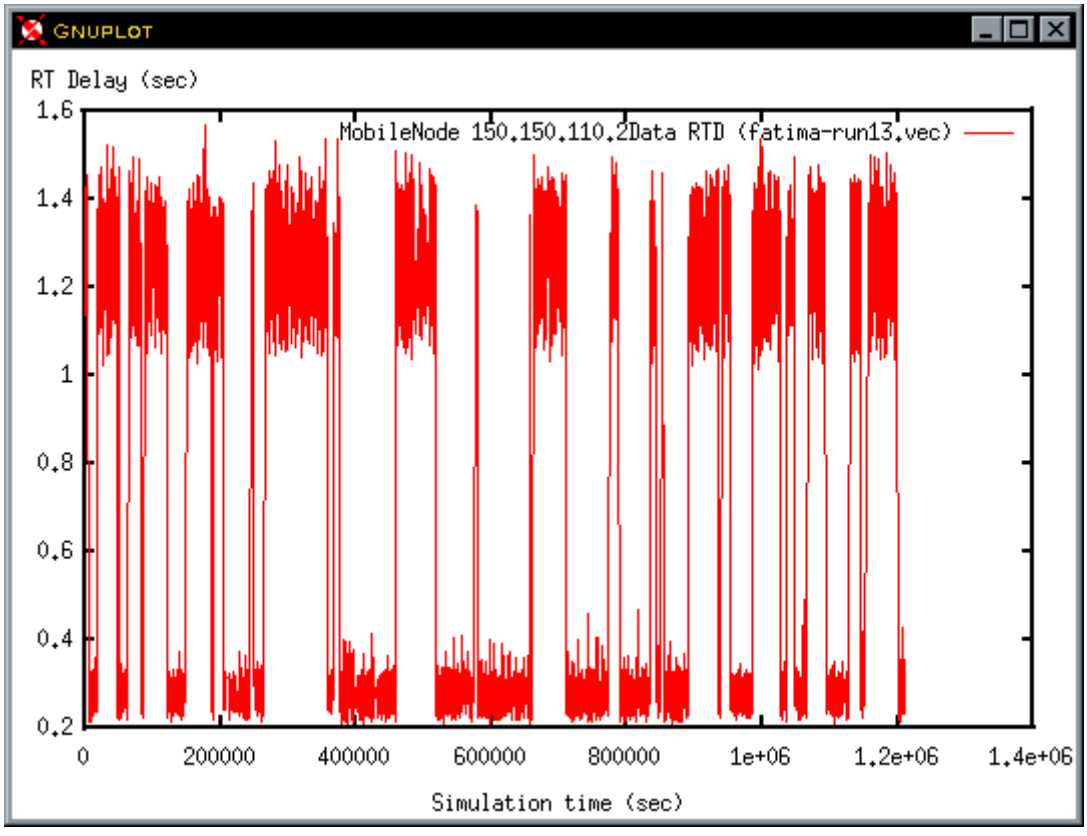

Figure 8.3-4 RTD: MN sending home, MobileIP-MobileIP, 60 MNs, MN@MIP

8.4 Scenario 3: MN sending to a CN in the visited network

This scenario represents the case in which the mobile node sends data packets to a correspondent node statically installed in the foreign network. This is the inverse case to the situation presented in Chapter 8.3.

That is why the same topology configurations have been used. However this time only the network without the CN was configured to have mobiles. The same overall number of 60 mobile nodes was used. The simulation time was set to 14 days.

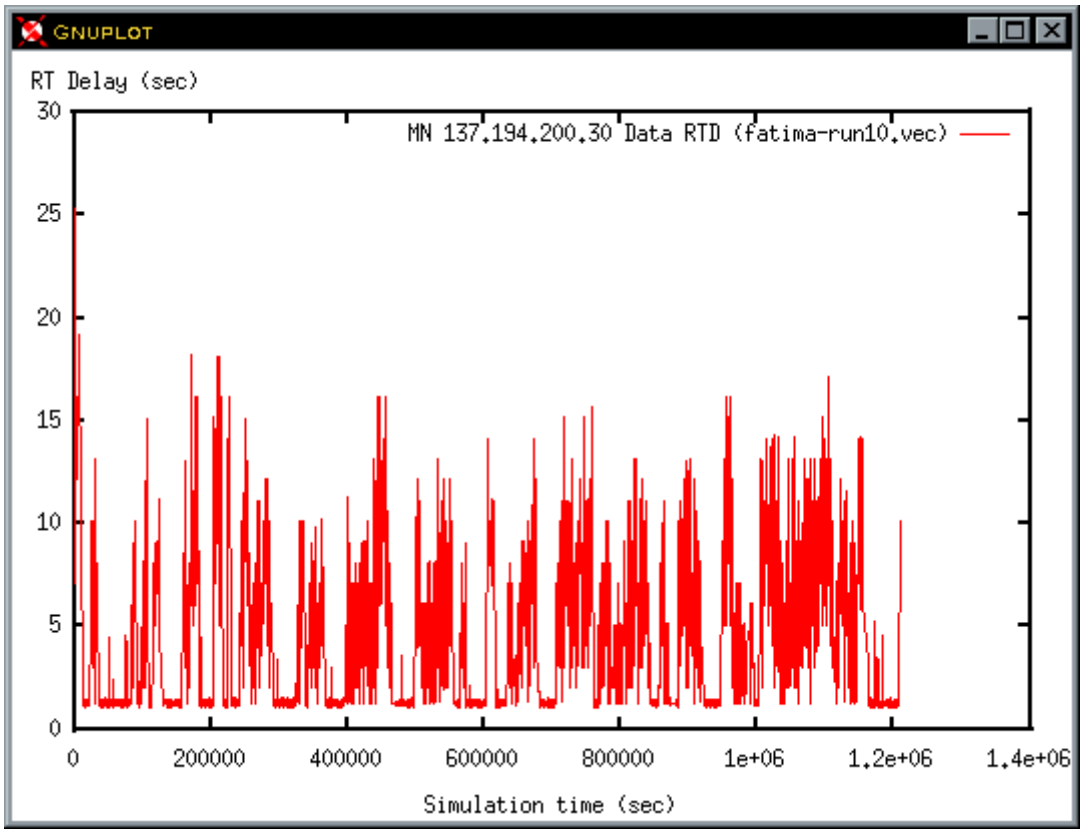

Figure 8.4-1 RTD: MN-foreign CN, FATIMA-FATIMA, 60 MNs, MN@FAT

Being in its home network, the MN does not use any mobility support. Hence its packets take the usual way over its local network, the gateway, the Internet backbone and the foreign gateway in the foreign network where they finally arrive at the CN. Its answer, also sent without any mobility support, takes the same way back to the MN's home network. This corresponds to the usual Internet routing and remains the same for all involved configurations. However, in the FATIMA-FATIMA topology case, the packets have to pass the two FATIMA gateways and experience a random delay caused by the defined wait times for the packet processing of the other present mobile nodes.

Being in the foreign network, MN's answer is delivered directly within this local network (by the FATIMA gateway, in the case where the foreign network is a FATIMA network or even by the FA in the standard Mobile IP with Reverse Tunneling case). Conversely, CN's answer is sent back to the home network of the CN. Even in FATIMA case, the central gateway does not intercept this packet because the CN is none of the registered mobile nodes; it is statically installed in the network. So, the answer packet goes back to the home network from where it is sent back by the central

gateway or intercepted and sent back over the backbone by the standard HA. For the topology using only standard Mobile IP networks it results in almost the same values for the measured RT delays, because the time for the packet sent within the (foreign) local network is very low compared with the time, which passes while sending over the backbone. This behavior is illustrated in Figure 8.4-4.

For the first three configurations it results in very similar RTD behaviors, since in every of these configurations there is a FATIMA central gateway involved in the packet delivery and suffering high load processing the packets of the other mobile nodes. This is illustrated in Figure 8.4-1, Figure 8.4-2 and Figure 8.4-3.

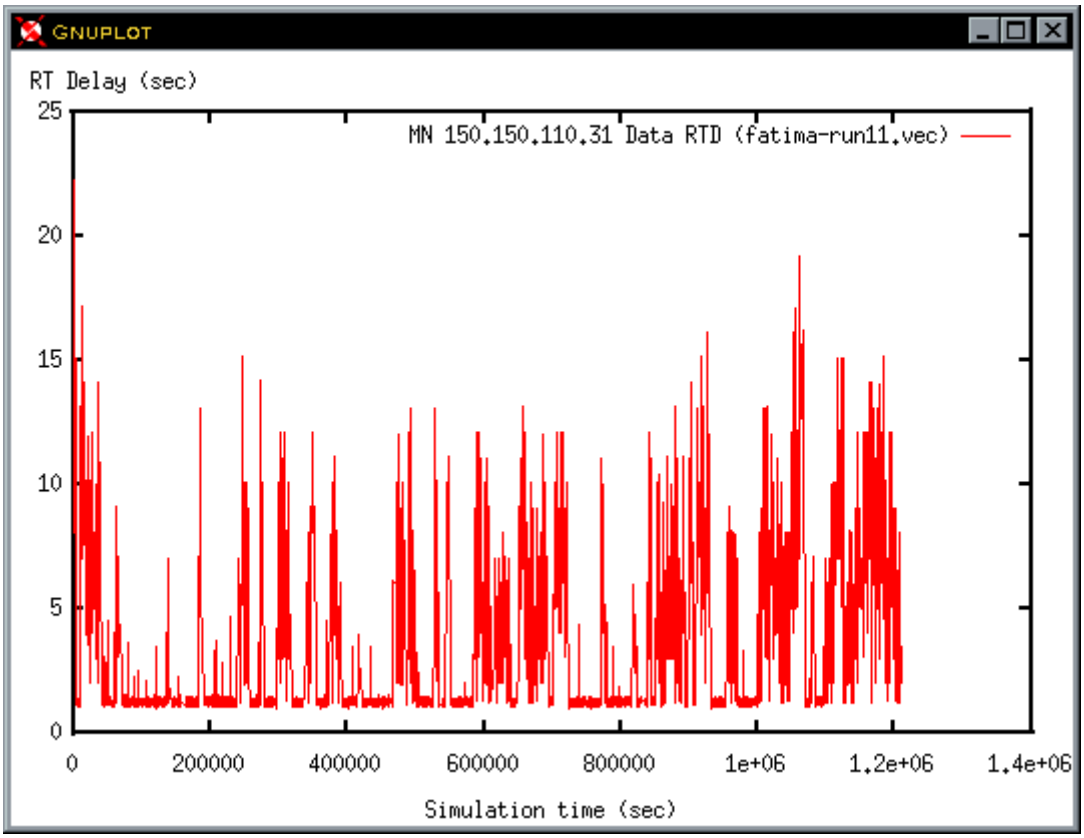

Figure 8.4-2 RTD: MN-foreign CN, FATIMA-MobileIP, 60 MNs, MN@FAT

Because of the explained independency of the mobility support for the case where the MN is in his home network, the overall RTD value mainly depends on the nature of the foreign network from the point of view of the mobile node. That is why the quantitative behavior of the FATIMA-Mobile IP configuration is slightly worse than that of the Mobile IP-FATIMA configuration.

In order to find a better illustration of this dependency, we have computed the mean in the interval `[0..t]` for each measured sample. The illustration of this mean for each involved configuration is shown in Figure 8.4-5. We used the same samples for the better comparison. After a typical stabilization phase (the curve approximates the real mean of the sample with every further sample value), we see that the both curves of the topologies where FATIMA network is used as a foreign network finally arrive at the same mean value (MNs 150.150.110.31 and 150.150.110.61). The pure Mobile IP case is characterized by the constant mean run (MN 160.160.110.32). The FATIMA-Mobile IP case (137.194.200.30) basically follows the first two samples involving FATIMA.

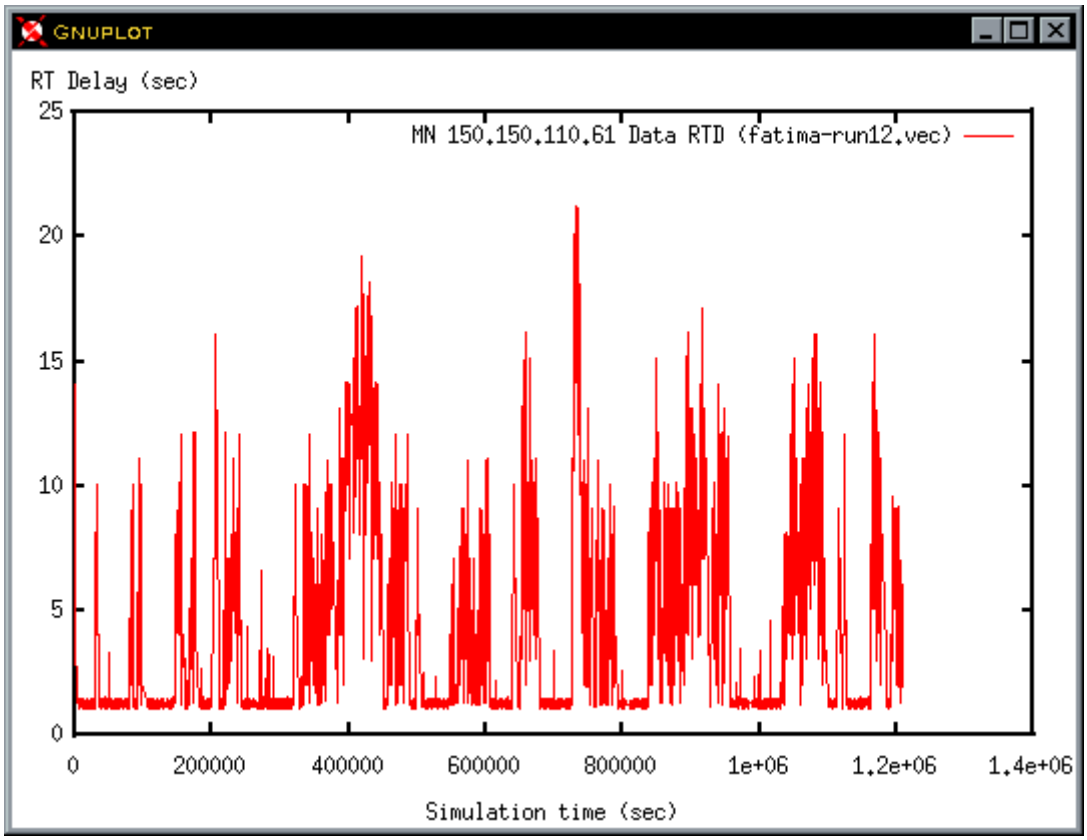

Figure 8.4-3 RTD: MN-foreign CN, MobileIP-FATIMA, 60 MNs, MN@MIP

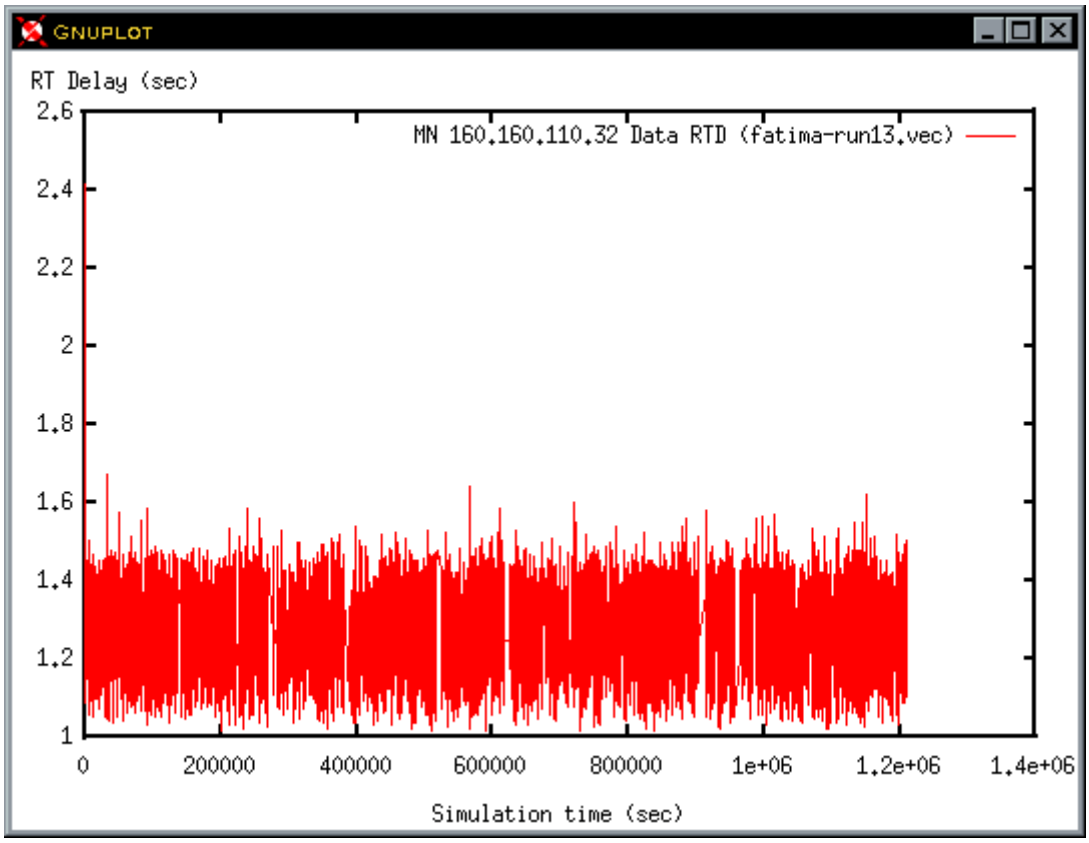

Figure 8.4-4 RTD: MN-foreign CN, MobileIP-MobileIP, 60 MNs, MN@MIP

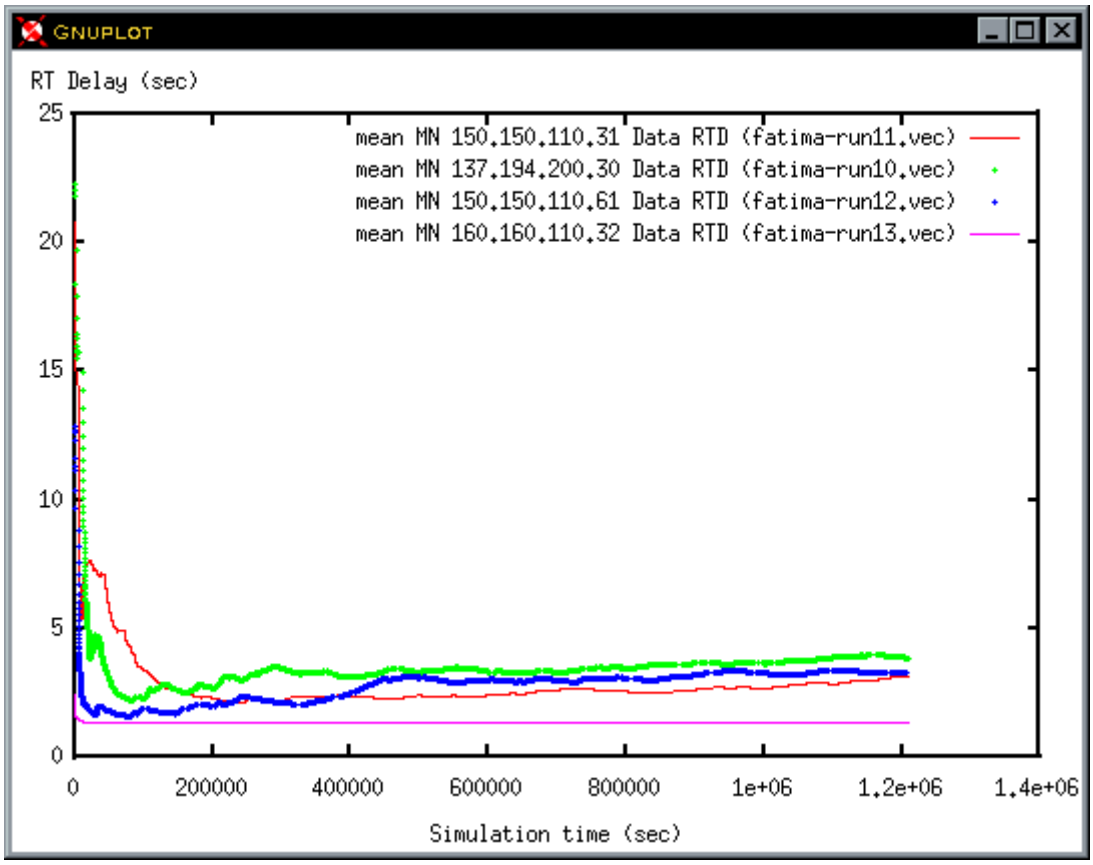

Figure 8.4-5 RTD: Means in [0..t] of measured samples for each topology

8.5 Scenario 4: Control traffic with local handoffs (FHE)

This scenario tests the performance of the Fast Handoff extension as defined in Chapter 3.6. In order to concentrate on the control traffic, the following changes have been made to the standard test base (→Table 8.1-1). The average visit time was set down, so the FHE could be used more often.

Handover values		
	Average visit time (exp mean)	400s
Data flow values		
	Send data	OFF
Features		
	FHE	ON

Table 8.5-1 Changes to the standard test base

The test run was 30 days of simulation time with 20 mobile nodes per network. One of the FATIMA networks was equipped with three FAPs, the remaining networks had two FAPs / FAs as usual.

We measured the number of packets and the overall traffic (sum in bytes) in the Internet node (represented by the Network module in the simulation). Because the data sending was turned OFF, all the received packets were registration packets sent between the both networks. The resulting Table 8.5-2 illustrates the gathered numbers:

Type	Number of packets	Sum (bytes)
Mobile IP only	42 226	4 053 700
Mixed mode	39 275	3 837 840
FATIMA only	36 377	3 628 720

Table 8.5-2 Results for FHE

As we can see, both the overall amount of passed control data and the number of packets decreases when using FATIMA networks supporting FHE. Basically, that is exactly what the FHE has been designed for. However, the average visit time has been set down to 400s while the demanded registration lifetime remained at the same value (5000s). Considering that a rough probability for a local handover is set to about 0.7, we should admit that under these circumstances the FHE could be used comparatively often.

In practice, not all mobile nodes will be capable of understanding FHE, which would result in not using it and, thus, worse performance results. Moreover, the quantitative improvement, which we are talking about here, is quiet low if we consider the passed simulation time (30 days). That is comprehensible since the registration process involves a total of four packets (when using a foreign agent instance) and only two of these packets are sent over the Internet backbone. Thus, even if this control traffic is minimized, the absolute economy effect is not impressive.

9 Conclusion and Outlook

Mobility support for IPv4 will remain an important topic in the development work of the next years. The approach shown and evaluated here is an interesting step on the way to free and relatively secure mobility. Generally, this approach is very likely to achieve the previewed results one day if the development efforts in this direction will be continued. Though the system definition is quite heavy compared to standard Mobile IP a lot of interesting possibilities and features can be proposed to users and administrators alike.

The implementation of this simulation without the complete and clear FATIMA-draft can be seen as the first breakthrough on the way to the complete standard. Although this implementation itself probably arises more new questions than it will ever be able to give answers to, it hopefully has helped a lot to show the theoretical feasibility of the project. It will probably also help to accelerate the standardization and the implementation processes by having built a small basis for both. The concept problems and deficiencies presented in Chapter 6 are believed to be solvable.

The first quantitative results gathered in this simulation and illustrated in Chapter 8 should be used with care due to the very theoretical nature of the developed simulation. Nevertheless, these results give a first comparative overview of the capabilities of the introduced and planned features. So, we could already claim that every try to optimize the Mobile IP control traffic probably is not worth dealing with. Even if the wanted results for the control traffic optimization can be achieved, the absolute amount of avoided traffic will be very low. Therefore, further development work should focus on security issues like firewall integration and secure data transfer and perhaps route optimization features. The other obvious observation is the high load at the FATIMA's central gateway, which is evidently causing the worse quantitative results compared to the standard Mobile IP. Thus, a research work should be invested in the modularization of the central gateway or perhaps in the development of a suitable cluster concept.

Once the problems shown in Chapter 6 are solved, a real and proper draft should be written. Then, the real implementation work can begin. This implementation can then be used to find more sophisticated conceptual and practical problems, which could additionally occur in reality.

These steps are believed to be a consequential and reasonable continuation of this work.

References

RFCs

[RFC0768] J. Postel. *User Datagram Protocol.* RFC 0793 (1980)

[RFC0791] J. Postel. *Internet Protocol*. RFC 0791 (1981)

[RFC0792] J. Postel. *Internet Control Message Protocol*. RFC 0792 (1981)

[RFC0793] J. Postel. *Transmission Control Protocol.* RFC 0793 (1981)

[RFC1256] S. Deering. *ICMP Router Discovery Messages*. RFC 1256 (1991)

[RFC1828] P. Metzger, W. Simpson. *IP Authentication using Keyed MD5*. RFC 1828 (1995)

[RFC1928] M. Leech, M. Ganis, Y. Lee, R. Kuris, D. Koblas, L. Jones. *SOCKS Protocol Version 5*. RFC 1928 (1996)

[RFC2002] C. Perkins. *IP Mobility Support*. RFC 2002 (1996)

[RFC2003] C. Perkins. *IP Encapsulation within IP*. RFC 2003 (1996)

[RFC2004] C. Perkins. *Minimal Encapsulation within IP*. RFC 2004 (1996)

[RFC2131] R. Droms. *Dynamic Host Configuration Protocol*. RFC 2131 (1997)

[RFC2138] C. Rigney, A. Rubens, W. Simpson, S. Willens. *Remote Authentication Dial In User Service (RADIUS)*. RFC 2138 (1997)

[RFC2267] P. Ferguson, D. Senie. *Network Ingress Filtering: Defeating Denial of Service Attacks which employ IP Source Address Spoofing*. RFC 2267 (1998)

[RFC2344] G. Montenegro. *Reverse Tunneling for Mobile IP*. RFC 2344 (1998)

[RFC2401] S. Kent, R. Atkinson. *Security Architecture for the Internet Protocol*. RFC 2401 (1998)

[RFC2406] S. Kent and R. Atkinson. IP Encapsulating Security Payload (ESP). RFC 2406 (1998)

[RFC2486] B. Adoba, M. Beadles. *The Network Access Identifier*. RFC 2486 (1999)

Firewalls

[Ches&Bell 94] W. R. Cheswick, S. M. Bellowin. *Firewalls and Internet Security*. Addison-Wesley Professional Computing Series (1994)

[Wack&Car 94] J. P. Wack, L. J. Carnahan. *Keeping Your Site Comfortably Secure: An Introduction to Internet Firewalls*. NIST Special Publication 800-10, U.S. Department of Commerce (1994)

[Schmeh 98] K. Schmeh. *Safer Net*. dpunkt Verlag (1998)

[BSI GSBH] Bundesamt für Sicherheit in der Informationstechnik. IT-*Grundschutzhandbuch*. http://www.bsi.bund.de/gshb (2000)

Cryptography

[Beth et al. 98] T. Beth, W. Geiselmann, P. Wichmann. *Signale Codes und Chiffren II*. Lecture script (1998)

[Geiselmann 97] W. Geiselmann. *Public Key Kryptographie*. Lecture script (1997)

OMNeT++

[OMNeT Web] A. Varga. *OMNeT++ Discrete Event Simulation System Web Resources* http://www.hit.bme.hu/phd/vargaa/omnetpp.htm

[OMNeT Man] A. Varga. *OMNeT++: Discrete Event Simulation System User Manual*. http://www.hit.bme.hu/phd/vargaa/opp-docs/usman.htm (1999)

IP and Mobility

[Schiller 99] J. Schiller. *Mobile Communications*. Addison Wesley Longman (1999)

[Valko et al. 99] András G. Valkó, Javier Gomez, Sanghyo Kim, Andrew T. Campbell. *On the Analysis of Cellular IP Access Networks*. Proceedings of the 6th IFIP International Workshop on Protocols for High Speed Networks PfHSN~99 (Aug 1999)

[Castel et al. 98] C. Castelluccia,. *A Hierarchical Mobile IPv6 Proposal*. Proceedings of AMOS ACTS Mobile Summit (Jun 1999). Also published as INRIA technical report TR-0226 (Nov 1998)

[Ramj et al. 99] Ramachandran Ramjee and La Porta, Thomas and Sandy Thuel and Kannan Varadhan and Shie-Yuan Wang. *HAWAII: A Domain-based Approach for Supporting Mobility in Wide-area Wireless networks*.

Prooceedings of the International Conference on Network Protocols (ICNP~99). IEEE Computer Society (Nov 1999). Extended version under http://www.bell-labs.com/user/ramjee/papers/hawaii.ps.gz (1999)

[Calhoun 98] P. R. Calhoun. *DIAMETER Mobile IP Extensions*. Work in progress (draft-calhoun-diameter-mobileip-01) (1998)

[Perk&Calh 99] C. Perkins, P. R. Calhoun. *AAA Registration Keys for MobileIP*. Work in progress (draft-ietf-mobileip-aaa-key-00) (1999)

[Perk&John 99] C. Perkins, D. B. Johnson. *Route Optimization in Mobile IP*. Work in progress (draft-ietf-mobileip-optim-08) (1999)

[Dyn HUT] Helsinki Institute of Technology. *Dynamics HUT Mobile IP Implementation*. http://www.cs.hut.fi/Research/Dynamics

FATIMA

[FATIMA] S. Mink. *Konzeption einer Firewall-Architektur für Mobile IP*. University of Karlsruhe, Germany, Institute for Telematics (itm), diploma thesis (1999)

[FATREP] S. Mink, F. Pählke, G. Schäfer, J. Schiller. *FATIMA: A Firewall-Aware Transparent Internet Mobility Architecture*. In Proc. of ISCC 2000, pages 172-179, Antibes, France. IEEE Computer Society (2000)

[MIPSEC] S. Mink, F. Pählke, G. Schäfer, J. Schiller. *Towards Secure Mobility Support for IP Networks*. In Proc. of ICCT 2000, pages 555-562, Beijing, China. IFIP (2000)

[FATFW] F. Pählke, G. Schäfer, J. Schiller. *Paketfilter- und Tunnelkonfiguration zur firewall-verträglichen Mobilitätsunterstützung in IP-Netzen*. Kommunikation in Verteilten Systemen (KiVS) (2001)

Simulation

[Ross 00] Sheldon M. Ross. *Introduction to Probability Models*. Academic Press (2000)

[Law et al. 99] Averill M. Law, W. David Kelton. *Simulation Modeling and Analysis*. McGraw-Hill (1999)

[Paxs et al. 97] Vern Paxson, Sally Floyd. *Why We Don't Know How to Simulate the Internet*. Proceedings of the 1997 Winter Simulation Conference (Dec 1997) ftp://ftp.ee.lbl.gov/papers/wsc97.ps

[Mukherjee 94] A. Mukherjee. *On the Dynamics and Significance of Low Frequency Components of Internet Load*. Internetworking Research and Experience, vol. 5, pp. 163-205 (Dec 1994)

[Forsb et al. 99] D. Forsberg, J. T. Malinen, J. K. Malinen, T. Weckström, M. Tiusanen. *Distributing Mobility Agents Hierarchically under Frequent Location Updates*. Proceedings of the Sixth IEEE International Workshop on Mobile Multimedia Communications (MOMUC 99). IEEE Communications Society (Nov 1999)

[Corson et al.00] M. Scott Corson, Alan O'Neill. *An Approach to Fixed/Mobile Converged Routing*. University of Maryland, Institute for Systems Research (2000)

[Press et al. 93] William H. Press, Saul A. Teukolsky, William T. Vetterling, Brian P. Flannery. *Numerical Recipes in C*. Cambridge University Press (1993). Also available online under: http://lib-www.lanl.gov/numerical/bookcpdf.html

Miscellaneous

[IETF] *Internet Engineering Task Force*. http://www.ietf.org

[Linux] *The Linux Home Page at Linux Online*. http://www.linux.org

[FreeBSD] *The FreeBSD Project*. http://www.freebsd.org

[MIP Web] *Mobile IP Web Resources*. http://www.computer.org/internet/v2n1/mobile.htm

[MIWG] *Mobile IP Working Group*. http://www.ietf.org/html.charters/wg-dir.html

[GNUFSF] *GNU Project and the Free Software Foundation*. http://www.gnu.org

[GNUGCC] Free Software Foundation. *GNU Compiler Collection*. http://www.gnu.org/software/gcc/gcc.html

Appendix A: Installing and running the simulation

The simulation code has completely been developed under RedHat [Linux] (Red Hat Linux release 6.0) with the GCC C++-compiler →[GNUGCC] version 2.91.66 (egcs-2.91.66) and OMNeT++ v2.0beta3. It was later updated to GCC v2.95.2 and OMNeT++ final release 2.0. The program was also successfully tested with OMNeT++ v2.0beta5. Unfortunately, newer versions cannot be guaranteed to be usable with the present code since OMNeT++ experiences slight changes from version to version.

Here you can find brief instructions on how to install and run the simulation. First of all, you will need OMNeT++. Please refer to [OMNeT Web] for the newest release. You have to properly install the downloaded OMNeT++ version at the host where you want to run the simulation. Please refer to [OMNeT Man] on the precise instructions on how to do so.

Once installed and running (you should be able to start the sample simulation included in each OMNeT++ release), you should install the provided files into some directory (called *DIR* later). Before you proceed, it is strongly recommended to read and understand the [OMNeT Man], at least the chapter explaining how to launch and configure the simulation programs.

You should change the respective switches in the `makefile` to point to the right directories. See the comments in the provided `makefile`. See the same file for the additionally defined targets.

`OMNETPP_ROOT = /usr/local/omnet`
This should be set to the directory where you installed your OMNeT++ package to.

`TK_LIB_PATH = -L/usr/lib`
You will need to modify this value if your TK libraries are not in that path.

`X_LIB_PATH = -L/usr/X11R6/lib`
Set the path to your X libraries here if it is not the standard path given above.

However, you can also let automatically create a new `makefile` using OMNeT++'s `makemake` command. You should make a backup copy of the provided one first.

After having adapted / created a suitable `makefile`, you can change into the source code directory and type:

```
user@host:DIR> make
```

This will take some time compiling and building the executable. You can start it afterwards by typing:

```
user@host:DIR> ./fatima
```

As each OMNeT++ executable, it supports switches, options and different other settings, which you can adjust in the provided `omnetpp.ini` file. There you can specify the needed simulation runs and parameters. Please refer to [OMNeT Man] for the full list of possible switches.

Appendix B: Fine-tuning the simulation

Here you can find the description of all used program parameters. Having changed their values and recompiled the source code you can change the simulation behavior.

NED file parameters

The parameters used in the NED files are described here with the currently used names. These names can be freely changed. However, changing these names in the NED files, you will have to change these names also in the `RootClass.h` file as explained in Chapter "Constants used in the `RouteClass.h` file".

Parameter	Layer	Format/Example	Explication
`address`	IP	`"129.13.50.1"`	The IP address of the node
`ipmask`	IP	`"255.255.*.*"`	The IP mask of the node
`route`*N*	IP	`Gate:IP-mask` `"1: 129.13.*.*"`	Matching IP packets will be sent over the gate with the given number. This is used to route packets over the right interface. Several route parameters can be specified and have to be named route1, route2, route3, etc. sequently
`coa_addr`	Mobile IP	`"129.13.50.2"`	The advertised CoA address
`ha_ip`	Mobile IP	`"129.13.50.1"`	The address of the HA
`vmntype`	Mobile IP, VMN	`10`	To set the type of the VMN agent. See `RootClass.h: enum VMNTypes`
`mnmask`	SIM	`"255.255.*.*"`	The IP-mask to be set in MNs
`mnbaseip`	SIM	`"129.13.100.1"`	The first IP from which on the built MNs will obtain their addresses being built by their MG. The invalid IPs will be avoided
`mnnumber`	SIM	`50`	The number of the MNs to be built at this MG
`mobg_id`	SIM	-	The responsible MG for this MN. Will be set automatically by the MG

Constants used in the `RouteClass.h` file

This is a complete list of constants, which can be set in the `RootClass.h` file. These constants correspond to different categories.

Constants holding parameter names in the corresponding NED file

See Chapter "NED file parameters" for the respective meaning of these parameters. These can be found in the table under the name set as value for the constant here. Changing these values you can also change the corresponding names in the NED files if you don't like them.

Parameter names concerning IP layer:

```
static const char _ROOT_ADDRPARAM[] = "address";
static const char _ROOT_NMASKPARAM[] = "ipmask";
static const char _ROOT_ROUTEPARAM[] = "route";
```

Parameter names concerning Mobile IP and simulation:

```
static const char _ROOT_COAPARAM[] = "coa_addr";
static const char _ROOT_HA_ADDRPARAM[] = "ha_ip";
```

Parameter names concerning the simulation itself:

```
static const char _ROOT_VMNTYPE[] = "vmntype";
static const char _ROOT_MNNETMASK[] = "mnmask";
static const char _ROOT_MNBASEADDR[] = "mnbaseip";
static const char _ROOT_MNNUMBER[] = "mnnumber";
static const char _ROOT_MOBGENID[] = "mobg_id";
static const char _ROOT_MOBIFACE[] = "mobiface";
```

Constants holding gate names in the corresponding NED file

These are the gate names as they are expected to be found in the module implementations. Setting these gate names to other values, you can freely change your gate names in the corresponding NED files.

Gate name used for sending handover messages between the MGs:

```
static const char _ROOT_MOBGEN_HOGATE[] = "next_ho";
```

Other gate names (the name will be appended to the prefix defining the direction):

```
static const char _ROOT_INGATEPREFIX[] = "from_";
static const char _ROOT_OUTGATEPREFIX[] = "to_";
static const char _ROOT_NETGATENAME[] = "net";
static const char _ROOT_MOBGATENAME[] = "radio";
```

Constants holding real life parameter values

These constants hold values, which will influence the behavior of the simulation and therefore the simulation results.

Different real life constants:

- `static const int _ROOT_BROADCAST = 0xffffffff;`
  The broadcast address value used in the simulation.

- `static const int _ROOT_LLC_MTU = 1500;`
  The MTU of the LLC layer.

- `static const int _ROOT_NET_MTU = 0xffff;`
  The MTU of the network layer.

MobileIP settings

- `static const USHORT _ROOT_MOBIP_UDPPORT = 434;`
  UDP port used in the simulation.

- `static const bool _ROOT_MOBIP_FHE = true;`
  Switches the usage of the FHE ON / OFF (→Chapter 3.6).

- `static const bool _ROOT_MOBIP_REVERSETUNNELING = true;`
  Switches the usage of the Reverse Tunneling option ON / OFF (→Chapter 2.6).

- `static const int _ROOT_MOBIP_REGCLEAN_PERIOD = 1000;`
  Period of time in seconds after which the used data-bases will be auto-cleaned by the agents deleting temporary expired entries.

- `static const int _ROOT_MOBIP_REREG_DELTA = 60;`
  If the remaining time of the registration is less than the number of seconds set here, the MN will try to re-register.

- `static const USHORT _ROOT_MOBIP_REGLIFE = 5000;`
  Mobile IP registration lifetime in seconds demanded per default (`0xffff` - infinity).

- `static const int _ROOT_ADVERT_INTERVAL = 300;`
  Router Advertisements interval in seconds →[RFC1256].
  The lifetime of the router advert will be `3*interval`. Advertisements are sent in random intervals in range `[0.75*interval, interval)`. Default value for the interval due to [RFC1256] is 600 seconds but it could be set lower since we use some mobile agents here.

Simulation message length settings (in bytes):

- `static const int _ROOT_MSGL_IP = 20;`
  IP header length.

- `static const int _ROOT_MSGL_UDP = 8;`
  UDP header length.

- `static const int _ROOT_MSGL_ICMP = 4;`
  ICMP header length.

- `static const int _ROOT_MSGL_MOBXT = 22;`
  Length of the AE added to each registration message.

- `static const int _ROOT_MSGL_MOBCTRL_REQ = 24;`
  Length of the Registration Request message.

- `static const int _ROOT_MSGL_MOBCTRL_RPLY = 20;`
  Length of the Registration Reply message.

- `static const int _ROOT_MSGL_RA_FA = 12;`
  Length of the router advertisements sent by the foreign agent instances.

- `static const int _ROOT_MSGL_RA_ELSE = 8;`
  Length of other router advertisement messages.

Simulation handover control settings

- `static const int _ROOT_HO_GLOBFAC = 6;`
  The factor applied to the mean used to compute delay till the next global handover.

- `static const int _ROOT_HO_EXPMEAN = 1000;`
  The mean of the exponential distribution used to compute time till the next local handover.

- `static const double _ROOT_HO_LOCPROB = 0.7;`
  Probability that a local handover will occur next time.

- `static const double _ROOT_HO_GLOBPROB = 0.3;`
  Probability that a global handover will occur next time.

Simulation dataflow control settings

- `static const bool _ROOT_DATA_SEND = true;`
  Turn sending data packets in the mobile nodes ON / OFF.

- `static const char _ROOT_DATA_DESTADDR[]`
  Destination address, to which all the data messages will be sent.

- `static const int _ROOT_DATA_MAXBURST = 100;`
  Maximum possible burst length, i.e. the number of packets sent in sequence. The real number is determined by a uniform distribution function.

- `static const double _ROOT_DATA_INTER_MEAN = 500;`
  The mean of the exponential distribution used to determine the delay time till the next burst.

Constants holding values for simulation program switches

Simulation display settings

- `static const bool _ROOT_CONSOLEOUTPUT = true;`
  Console output ON / OFF.

- `static const bool _ROOT_OMNETOUTPUT = true;`
  OMNeT++ window output ON / OFF.

- `static const int _ROOT_NETMSGKIND = 1;`
  The kind value for the usual network packet (influences the color of the message in the OMNeT++ presentation).

- `static const int _ROOT_INTERNMSGKIND = 5;`
  The kind of the messages sent internally and being not real network packets (influences the color in the OMNeT++ presentation).

Message names, as they will be set in the OMNeT++ messages:

- `static const char _ROOT_MSGNAME_IP[] = "IP";`

- `static const char _ROOT_MSGNAME_ICMP[] = "ICMP";`

- `static const char _ROOT_MSGNAME_UDP[] = "UDP";`

- `static const char _ROOT_MSGNAME_MOBCTRL[] = "MOBCTRL";`

- `static const char _ROOT_MSGNAME_MOBXT[] = "MOBXT";`

- `static const char _ROOT_MSGNAME_RA[] = "ADVERT";`

Implementation optimization settings:

- `static const int _ROOT_HTSIZE = 256;`
  Initial size of the hash table used for the databases.