Protocol Design and Analysis of a HIP-based Per-Application Mobility Management Platform

László Bokor László Tamás Zeke Szabolcs Nováczki Gábor Jeney
Budapest University of Technology and Economics, Department of Telecommunications
Magyar Tudósok krt.2., H-1117 Budapest, Hungary
Tel: (+ 36 1) 463-3261, Fax: (+ 36 1) 463-3263
{goodzi, koci, nszabi, jeneyg}@mcl.hu

ABSTRACT

Rapid evolution of wireless networking has provided wide-scale of different wireless access technologies like Bluetooth, ZigBee, 802.11a/b/g, DSRC, 3G UMTS, LTE, WiMAX, etc. The complementary characteristic of the above architectures motivates next generation network operators to integrate them in a supplementary and overlapping manner. Recent wireless devices are equipped with multiple interfaces, thus enabling concurrent communication sessions. With the advance of such heterogeneous structures – and considering that users are often running applications simultaneously – the traditional per-host mobility management approach cannot be the optimal solution for handling connection changes. Instead, the concept of per-application mobility management is to be introduced, where a dedicated interface (i.e. access network) is selected by each application according to its QoS prerequisites and the actual networking conditions. Aiming to benefit from this novel concept in practice, in this paper we designed and evaluated a HIP-based per-application mobility management platform founded on the promising Host Identity Protocol (HIP) and the cross-layer building blocks closely incorporating with it.

Categories and Subject Descriptors

C.2.1[**Network Architecture and Design**]: Wireless communication
C.4 [**Performance of Systems**]: Reliability, availability, and serviceability
I.6.4 [**Simulation and Modeling**]: Model Validation and Analysis
I.6.5 [**Simulation and Modeling**]: Model Development - *Modeling methodologies*
I.6.6 [**Simulation and Modeling**]: Simulation Output Analysis
I.6.8 [**Simulation and Modeling**]: Types of Simulation – *Discrete event*

General Terms: Algorithms, Management, Measurement, Performance, Design, Experimentation, Security.

Keywords: Per-application mobility management, Host Identity Protocol (HIP), Application-specific handover management, Cross-layer protocol design, Protocol simulation, INET/OMNeT++, HIPSim++ simulation framework, Performance evaluation and analysis.

1. INTRODUCTION

Current trends in the telecommunication world show rapid growth of Internet related services and ever growing demand for them. More and more users are willing to access the Internet from their portable devices. People want seamless, ubiquitous Internet access anytime and anywhere. To satisfy the demands, operators must use heterogeneous access technologies: WiFi, WiMAX, UTRAN, HSPA, LTE and GERAN coexist in most European countries. Moreover, they usually provide overlapping coverage: users can choose from a set of available access technologies based on their preferences, policy, or QoS requirements.

When multiple networks are available for accessing the Internet, the user can dynamically change between them. The first solution was to use only one interface, and switch, if the signal quality drops below a given threshold. A self-explanatory improvement can be reached if all available interfaces are used simultaneously. This is the so-called multihoming. However, using such solution raise the question of how packets should be distributed among available interfaces: what is the policy the user wants to follow. If the user runs several applications (e.g. file downloading, voice communication, video streaming, e-mail), each application has its own criteria which usually contradict each other. A further improvement can be achieved, if multihoming is not controlled by the whole terminal/host (this is called per-host, or per-terminal mobility), but the connection of each application is handled independently (this is called per-application mobility). For instance, e-mails do not require huge bandwidth, basic GSM can do the job. However, if the user downloads files, they are hardly going to use basic GSM: high-speed connection is required. In the case of per-application mobility management, several connections exist and they can use completely different interfaces. Each application can be described by their QoS requirements, and thus the best access technologies can be bundled to them. Section 2 describes the per-application mobility concept, and refers those publications where it has been introduced.

IP was designed in the 1970's, when all Internet hosts were connected using wires: they were fix hosts, not changing their locations. In contrast, nowadays users are rarely connected using wires: most users are mobile, thus changing continuously their point of network attachment. The shortcomings of the Internet Protocol (IP) come from the early days. Although IPv6 was designed to address all open problems of IP, due to the fact that designers wanted to keep the original concept, there are still many strange issues.

The most spectacular one is the double role of IP addresses. An IP address identifies the host on the Internet: all communication starting from or ending at a given entity is identified by its IP address. On the other hand, IP address has a topological locator role too: IP address consists of a subnet identifier (called prefix in IPv6), which tells the position of the entity on the Internet. These two roles (identifier and locator) make things complicated when the node starts to move. Changing the network yields different IP address. However, changing the IP address results in loss of already established connections.

MobiWac'09, October 26–27, 2009, Tenerife, Canary Islands, Spain.

Obviously there are many solutions for the above mentioned problem. One of them is a brand new protocol, which is called Host Identity Protocol (HIP). HIP is a new approach which decouples IP addresses from applications by proposing a new, cryptographic namespace for host identities such providing sophisticated and secure mobility/multihoming support, and making it a powerful toolset as the basis of a prosperous per-application mobility implementation. Using HIP, hosts can communicate seamlessly, even if the access network changes. Some details of the signaling and the fundamentals of HIP are described in Section 3. Section 4 presents our proposal on how HIP is used for per-application mobility.

In order to evaluate the proposed platform we created a simulation framework for HIP and our HIP-based per-application mobility scheme in INET/OMNeT++. Our analysis shows that the presented platform provides enhanced QoS provision as it optimally utilizes the available access networks based on application profiles and networking conditions. Section 5 shows the simulation environment which we used to evaluate our solution. The simulation scenario is described in Section 6. The results obtained from the simulations are included in Section 7. Finally, Section 8 concludes the document with some general thoughts on some possible extensions of the method.

2. RELATED WORK

Per-application mobility is quite a new mobility concept which refers to scenarios when a host is multihomed with more than one wireless interfaces and wants to control mobility decisions in an application-wise manner. Note that application mobility or process migration and per-application mobility are different concepts. The goal of application mobility solutions [1][2][3][4] is to move an already running application from a source host to a destination host during its execution. This improves load share management, fault tolerance and enables data access locality. The most important challenge here is to achieve mobility transparent to the applications moved.

On the other hand per-application mobility focuses on multihomed hosts with multiple wireless interfaces and on the ability to switch the traffic of each application running on the host independently between these interfaces. This research area is not much discovered yet thus only a few related work can be found in the literature. The most important effort made in the field of per-application mobility is the work of Moonjeong Chang et al. presented in [5] and [6]. Here the authors define a per-application mobility framework with a network-level handover management subsystem. The solution uses cross layer techniques to collect dynamic measurement results from the entire protocol stack. These results are used to trigger vertical handover decisions in a per-application manner. The proposed platform consists of the following main elements (Figure 1). There is a Monitoring agent (MA) for every protocol layer as an interface towards those layers. MAs collect dynamic status information in the different layers and forward them to the profile database (PDB). MAs are the way to send control information to the correspondent layer. The PDB stores and updates static and dynamic information, which is necessary for the decision engine (DE) to make vertical handover decisions on a per-application basis. The DE works according to policies to decide when to trigger a handover. Finally there is an IP Agent, which is responsible to map the actually used IP address to the address used by the ongoing session. This network-level component is also responsible of managing handover scenarios.

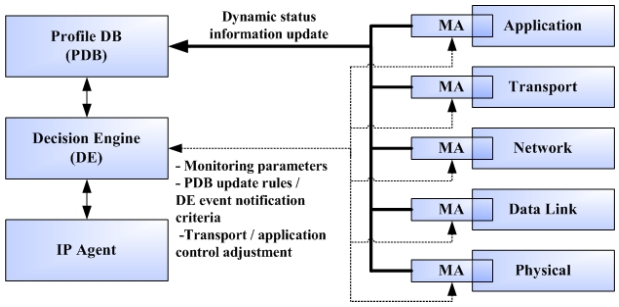

Figure 1. Per-application mobility platform proposed in [5]

Our per-application mobility management solution adapts the basic ideas captured in [5][6] and implements them in a HIP-based environment. We defined the details how to handle per-application mobility in a HIP environment, simulated and evaluated it. The basic protocol standards of HIP had to be modified to be able to handle per-application mobility. First concept of the node level HIP sessions had to be refined into application level sessions. Second the end-to-end readdressing mechanism was aligned to this concept change. Finally the ideas described in [5][6] were adapted into a HIP-aware environment. A Monitoring/Mediator Agent (MMA) is collecting information and handling control duties all through the stack, which is forwarded to the Decision Engine (DE). The DE is making application-wise handover decisions based on that information and according to the Application Profile DataBase (APDB). The APDB stores the QoS preference of applications running on the host. This can be considered as our contribution to per-application mobility management research activities. This work will be presented in the following sections.

3. FUNDAMENTALS OF HIP

This section is to give a short overview of Host Identity Protocol (HIP) [7], which lies as a basic framework for our per-application mobility solution. Our goal here is to show the key concepts of HIP first than highlight the most relevant protocol details to improve the intelligibility of the rest of the paper.

The main design goal of HIP is to implement the locator-identifier split concept that separates the dual roles of IP addresses. The core idea was to assign at least one globally unique identifier to every HIP-aware node. This is the namespace of Host Identifiers (HIs). With HIs HIP provides node identification while IP addresses remain pure locators. All the new functionalities to handle this idea form a protocol layer, which resides between the transport and IP layers and is called the Host Identity Layer. This is not only for locator-identifier split but also for a kind of demarcation of those layers. Using HIP transport layer sessions are bound to HIs not to IP addresses. Thus dynamic changes of IP addresses have no effect on higher layer connections any more. This enables HIP to be very powerful in handling mobility scenarios. To further empower the protocol the elements of the namespace were chosen to be cryptographic in nature. The goal of HIP design was to accouter the protocol with strong, built-in security mechanisms. This led HIs to be public-private key pairs. Thus HIP can benefit from all the advances of asymmetric cryptography and can use strong security mechanisms as a generic feature. HIs are usually variable length depending on the key generation method used to create them. Thus HIs are rarely

appear in protocol messages. Instead a 128 bit fixed length representation of HIs called the Host Identity Tag (HIT) appears to refer to the correspondent HIP entity. HITs are created by taking a one-way cryptographic hash function over a HI.

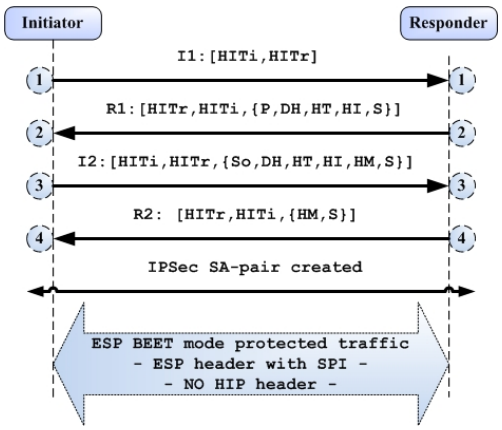

Figure 2. The HIP Base Exchange sequence

A four-way security handshake is used to build up a HIP session between two nodes, the Initiator and the Responder. This is the Base Exchange (BE), which results an IPSec Encapsulating Security Payload (ESP) Security Association (SA) pair [8]. Furthermore the BE implements a Diffie-Hellman key exchange to create shared symmetric key at the peers. This key is used to encrypt payload in ESP packets. The Initiator (I) starts the BE by sending the I1 message, which is not more than a HIP header with source and destination HITs. The aim is to trigger the exchange. The Responder (R) replies with an R1 message containing a HIP header and some additional HIP parameters. One of these parameters is the PUZZLE (P), which is a cryptographic challenge and must be solved by I. This is to force I to consume resources and time to solve it, which protects R against certain kinds of Denial-of-Service (DoS) attacks. Further parameters are the Diffie-Hellman (DH) parameters of R, the HIP_TRANSFORM (HT) parameter for the encryption and integrity algorithms supported by R to protect the exchange, the HOST_ID (HI) parameter for the HI of R and finally a signature (S). In the third packet, I2, I sends the SOLUTION (So) of the puzzle. The rest of the parameters (DH, HT, HI and S) were already defined except HMAC (HM) which is a message authentication code protecting I against reply attacks. Finally the R2 message concludes the exchange (see Figure 2 for the message sequence).

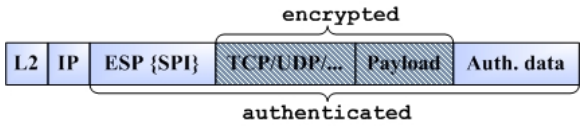

Figure 3. ESP BEET mode transport format

After the BE hosts can forward data to each other on the IPSec ESP SA pair. [9] Note that while HIP control packets (e.g. packets used during the BE) have a HIP header attached, the payload packets are not holding HIP header. Instead the Security Parameter Index (SPI) of the ESP header is used to map incoming messages to the correspondent security association. While HIP can be used by any ESP transport modes [9] defines a new ESP transport mode that is optimized for usage with HIP. This is called the Bound-End-to-End-Tunnel (BEET) mode. Figure 3 shows the form of packets sent in this mode.

In case of mobility scenarios the HIP association between hosts has to be updated. This is handled by the UPDATE mechanism of HIP (Figure 4) [10]. In the simplest case the mobile host upon IP address change sends an UPDATE packet to its peers. This packet holds a LOCATOR (L) parameter that is used to indicate the new address to be used and a sequence number (Seq). The SA to be updated is indicated in the ESP_INFO (EI) parameter. This parameter has special importance when the SA itself is being replaced during the update procedure as well as in case of multihoming scenarios. The peers must verify the new address to avoid being victims of security attacks. The peers send an UPDATE packet to the new address with an ECHO_REQUEST (ERq) parameter. This includes a nonce that the mobile must echo back to the peer in a third UPDATE packet. The packet also contains a sequence and an acknowledgement number. Finally there is an ESP_INFO parameter, which indicates the SA to be updated. When the SAs are to be replaced and in case of multihoming this parameter contains the new and old SPI values. The packet may be signed (S). The mobile host replies with a third UPDATE packet including an ECHO_RESPONSE (ERs) parameter. This is to echo back the nonce received in the previous packet and to make the new address being verified at the peer host. The packet also holds an acknowledgement number. Note that unverified addresses can be used to send only a limited amount of data. A Credit-Based Authentication method is used to control the communication on unverified addresses [10].

There are some scenarios when the end-to-end readdressing functionality is not sufficient. Initial reachability and simultaneous movement of mobile hosts are handled by the Rendezvous mechanism of the protocol [11]. In these situations a special HIP-aware network entity, the Rendezvous Server (RVS) assists the communication by forwarding the I1 packet of the BE to R, whose actual location is not known by I. Mobile hosts should register to one of these RVSs and update their location information upon mobility. Note that considering per-application mobility management HIPs main advantage is its effective end-to-end update mechanism. This enables to update the sessions application-wise between the communication endpoints. On the other hand RVS mechanism still needed to establish HIP contexts when it comes to mobile endpoints.

As it was expressed already, HIP was designed to be able to work on multihomed hosts [10]. In this case the host has more than one physical interfaces or global addresses. It is recommended to use different SAs for different interfaces or addresses. To do this, the multihomed host creates a new inbound SA and a corresponding SPI. The host can notify its peers about additional interfaces or addresses by using a bit modified UPDATE mechanism. The first UPDATE packet should hold an ESP_INFO parameter having the NEW SPI field set to the newly created SPI value and setting the OLD SPI field set to zero. The packet also contains a LOCATOR parameter that indicates the new address-SPI mapping and the old one as well. One step closer to per-application mobility is another way of grouping SAs. One can assign different SAs to every transport layer socket. However this might lead to scalability problems when having lots of applications all of which using possibly more than one transport layer socket. To reduce this overhead another way is to assign one SA application-wise. In this case all the sockets used by one application are assigned to one SA and all applications will use different SAs for

communication. This latter case is very important when considering per-application mobility and will be detailed later in the next sections.

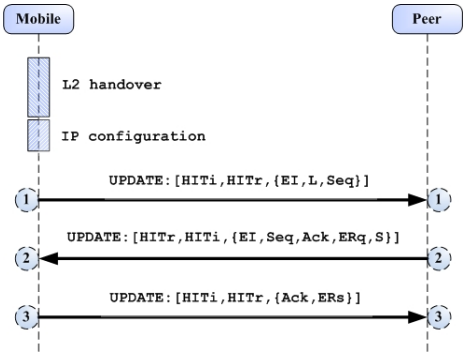

Figure 4. HIP update procedure without rekeying

In this section we have given a short introduction to HIP. We have shown the core concepts and highlighted some details significant for per-application mobility management. HIP turned out to be effective in end-to-end readdressing and provides strong generic security services both of which are very important in our case. These properties address the main shortcomings of the current TCP/IP architecture thus HIP can be considered as a base technology candidate in the next generation Internet era.

4. HIP-BASED PER-APPLICATION MOBILITY MANAGEMENT PLATFORM

The main goal of our HIP-based per-application mobility management platform is to adapt the paradigm of per-application mobility to HIP at protocol-level by introducing as few changes in the existing HIP standards and recommendations as possible.

4.1 Main architecture

The architecture of our HIP-based per-application mobility management platform relies on the module structure and design proposed by Moonjeong Chang et al. in [5][6] but extends their IP-level per-application solution with the advanced mechanisms of the promising Host Identity Protocol aiming to introduce a more effective and secure solution for per-application mobility management. Figure 5 shows the details of our platform and the following subsections contain the details of the particular modules.

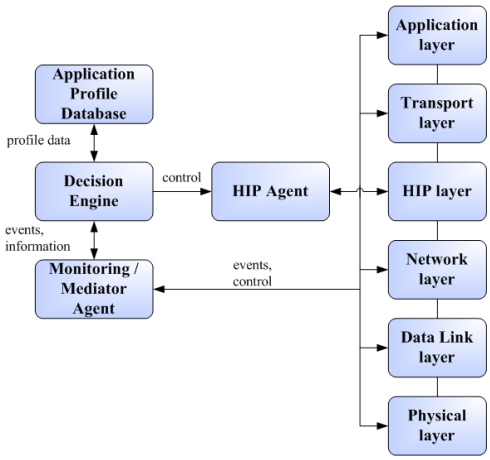

Figure 5. The proposed management platform

4.1.1 Monitoring/Mediator Agent

The Monitoring/Mediator Agent (MMA) collects protocol specific information from different layers of the TCP/IP hierarchy in order to provide information for cross-layer optimization decisions of per-application mobility. Passive and active measurements are both allowed for monitoring purposes. The crucial data pieces to be collected by the MMA for decision support are the following:

- Availability and basic information on the interfaces;
- Actual IPv6 addresses of available interfaces;
- Information of running applications/sockets.

Control of different layers is also an important task of this module: intervention into different protocol layers can be performed after decisions in order to extend the possibilities of cross-layer optimization.

4.1.2 Application Profile DataBase

The Application Profile DataBase (APDB) stores and maintains profile attributes of ongoing application sessions in order to relay user preferences and QoS requirements of applications to the Decision Engine. Profile attributes can be originated from static database (e.g. filled with pre-defined minimum requirements or user preferences) or can be gathered in a dynamic way directly from the applications. The latter case requires API changes in existing codes but provides more precise profile information and better results.

4.1.3 Decision Engine

The Decision Engine (DE) processes cross-layer information provided by the MMA and application profile data originated from the APDB. The main task of this module is to assign active applications to interfaces and to make handover decisions in a per-application manner. In order to achieve this, DE stores all the actual details of running applications (e.g. assigned interfaces, SAs, etc.) and in case of incoming updates from the MMA it recalculates the optimal interface for every affected application. If handover is needed, the DE instructs HIP Agent to initiate the appropriate HIP mechanisms (Figure 6).

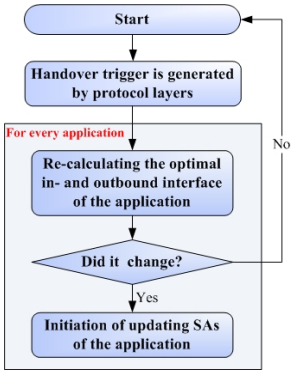

Figure 6. Mechanism of handover decision

4.1.4 HIP Agent

In our HIP-based per-application mobility framework the HIP Agent is responsible to initialize mobility procedures in the HIP layer based on the control information sent by the DE, and also to maintain per-application bindings between the host machine and its partners. The HIP Agent is not completely separated from the Host Identity Layer: it can be considered as a collection of

administrative and control functions required for HIP-based per-application mobility management. These functions are detailed in Section 4.2.

4.2 Operation of HIP extensions

From the HIP point of view, multihomed Security Associations are the protocol entities which can make a HIP system to be able to handle mobility in an application-wise manner. In consequence, a certain SA grouping scheme and a modified UPDATE mechanism are the keys to the HIP-based per-application mobility management framework. Because our approach is end-to-end based and uses spanned SAs between communicating peers as basis for operation, therefore standards of HIP DNS and RVS extensions can remain intact and only SA handling, packet processing and UPDATE procedure needs to be slightly modified.

4.2.1 SA grouping and LOCATOR extension

In the proposed SA grouping scheme we create and manage exclusive SAs for every running application between two HIP hosts and distinguish them with a quintet of Source HIT, Destination HIT, Source port, Destination port and Transport type. Such a quintet can be assigned even to smaller entities than applications (e.g. to sockets) but in that case scalability easily becomes a serious issue: applications may own a considerably big number of sockets resulting in unwanted signaling overhead of SA management and related HIP messages. We assume that every host in the communication supports the proposed SA grouping method otherwise separation of packets belonging to different applications can not be possible.

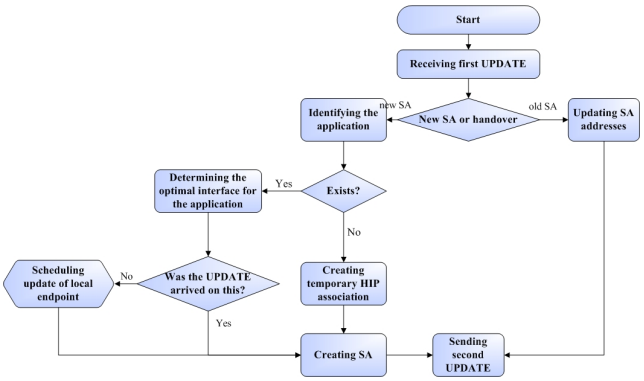

Figure 7. Processing incoming first UPDATE messages

Since we can not presume on that applications transact bi-directional sessions, there is a need to unambiguously represent which SA belongs to which application. An evident way to achieve that is using the above defined quintet in LOCATOR parameters for marking applications and SAs. HIP signaling messages already contain Source and Destination HITs, thus LOCATOR parameters must be extended only with fields of Source and Destination port and Transport type. These extensions can be easily derived from the existing *type 1 LOCATOR*s: a new 16 bit Source port, a 16 bit Destination port and a 4 bit Transport type descriptor field are to be introduced.

4.2.2 Modified UPDATE mechanism

In our per-application mobility management framework HIP UPDATE procedure is initiated in two distinct cases:

- If a new application is to be introduced in an existing HIP association (requires setup of a new SA);
- If an already running application is to be handed over (i.e. a new interface was chosen for the application and handover is to be performed).

In order to support the above cases no significant modifications are needed in the standard HIP UPDATE: the sequence itself and its main parameters are not changed except by the appearance of the new LOCATOR type and the extension of processing mechanism of incoming UPDATE messages.

The new type of LOCATOR parameter was introduced in Section 4.2.1, while the extended processing mechanism of incoming HIP UPDATE packets is shown in Figure 7.

At the reception of an incoming packet first it has to be decided whether a new SA is to be created or readdressing and rekeying of an existing one is to be performed. This decision can be made by parsing the ESP_INFO parameter of the incoming message. If the latter is the case, the SA will be updated and the second UPDATE message of the standard sequence will be sent. If a new SA has to be set up, then one of the LOCATOR parameters of the incoming UPDATE message pertains to the association to be created, and contains information (e.g. SPI value) helping to compile the quintet which identifies the application and the relating socket. If the application can not be found (i.e. the socket is not active) then we build up the HIP connection and let the upper layers to handle the exception. If the application is running, then we first determine the optimal interface for that application (see Section 4.1.3 for further details on interface selection and decision mechanisms) and check whether the UPDATE message arrived from the optimal interface or not (i.e. whether the SA is to be built through the optimal interface or not). If the incoming interface is the optimal one, then we continue the UPDATE procedure by setting up the SA and sending out the second UPDATE packet according to the standard message sequence. If the inbound interface of the first UPDATE packet is not optimal for the application, then we create the SA over the sup-optimal route but simultaneously schedule a secondary update for the SA's local endpoint in order to redirect the communication path of that particular application to the optimal interface.

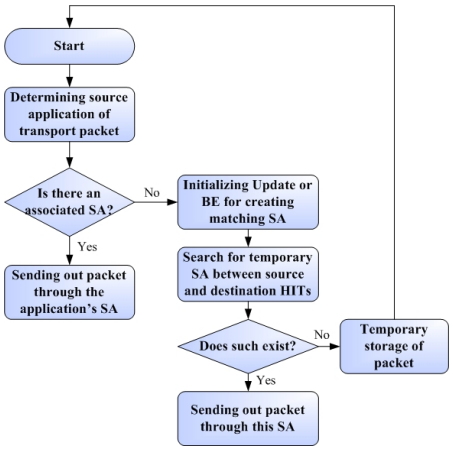

Figure 8. Processing packets arriving from the transport layer

4.2.3 Processing transport layer packets

In our framework the processing scheme of packets arriving from the network does not bring considerable changes to the existing HIP RFCs: signaling messages are processed in the normal way with the extensions introduced above, while incoming data packets are treated completely according to the standards (SPI

values are used to identify the appropriate SA). However, in case of packets arriving from the transport layer (originated by the applications) the standard way of operation changes because the fitting SA has to be selected for the outbound communication (Figure 8). In order to do this, every transport packet has to be inspected and the originating application and its corresponding SA has to be determined. If no associated SA exists, and then HIP UPDATE or BE procedures are initiated and the packet will be temporarily stored or sent out through an existing – but not yet optimized – interface, as long as the BE/UPDATE finishes and the optimized path (i.e. SA) gets ready.

5. Simulation Framework

In order to provide an extensible and precise simulation model for our HIP-based per-application mobility management platform, first we developed an IPv6-based Host Identity Protocol model called HIPSim++[1]. The model is built on the top of the 20081128 version of INETwithMIPv6 [12] which is an extension and TCP/IP model collection of the component based, modular OMNeT++ discrete event simulation environment [13][14]. Despite the fact that HIP relies on the functions of IPSec, a full implementation of IPSec and relating algorithms is not part of our simulation model: HIPSim++ does not possess properly realized Diffie–Hellman mechanisms, RSA engine, cryptographic hash functions and puzzles because precise mapping of all the security algorithms is out of scope of our current efforts. The main design goal of HIPSim++ was to provide a basis for our HIP-based application-specific mobility management proposal by accurately simulating core HIP instruments focusing on the advanced mobility and multihoming capabilities and wireless behavior of the protocol and providing only skeleton implementation of the above mentioned mathematical apparatus.

Our implemented HIP layer registers HIT-IP bonds for every communication session, and when packets from the transport layer arrive, destination and source HITs are replaced by destination and source IP addresses. Higher layers know only about HITs and Port numbers: applications are not aware of that they are using HITs instead of IP addresses. By realizing this scenario, all the advantages and benefits of applying HIP can be exploited and also HIPSim++ can be easily used in the existing INET-based simulation models.

5.1 Main Modules of HIPSim++

The core of our HIPSim++ implementation is the HIP layer module named as *HIP module* which creates a daemon instance called *HIPSM* for every new HIP session. This daemon is responsible for all mechanisms of the HIP State Machine (HIP SM) described in [8], e.g. for handling HIP Base Exchange and HIP mobility functions. One such daemon instance cares of one SA, which will be identified by the local SPI. HIP SM daemons are registered by destination and source HITs (and SPIs) in the *HIP module*. HITs have to be provided by the applications (or rather the transport layer), therefore HIP-capable DNS extensions [15] are also integrated into HIPSim++. The *HIP module* is also responsible for managing changes of states and/or addresses of host interfaces. Main methods of the module are the following:

- *handleMessage*: Leads the incoming packets towards the appropriate methods (packets can be arrived from the transport and the network layers).
- *handleMsgFromTransport*: Checks whether there is an existing HIP SM for the packet's destination HIT and forwards the packet towards the appropriate HIP SM.
- *handleMsgFromNetwork*: If the arrived packet is a HIP I1 (see Figure 2), then creates a new SM. If another HIP signaling packet has arrived, then searches for the appropriate SM and forwards the packet to it. If a HIP data message comes in, then the method gets the packet out from the ESP and forwards it to the appropriate SM based on the SPI value.
- *handleAddressChange*: This method is applied by the HIP module in order to gather information about lower layer events (like IP address changes) using the capabilities of the INET's *NotificationBoard* object. After processing such lower layer information, HIP UPDATE mechanisms can be initiated at the relevant SMs.

5.1.1 HIPSM module

The *HIPSM module* implements the main functions of the HIP State Machine. In our model transitions of HIP State Machine assumes that packets are successfully authenticated and processed. This behavior is in consistence with the standards, therefore our skeleton implementation of security algorithms do not hamper our model to accurately simulate HIP mechanisms. One instance of *HIPSM* represents and manages one HIP connection with one Security Association. *HIPSM* handles transitions occurring during HIP Base Exchange, RVS registration, UPDATE mechanism, etc. and generates HIP messages according to the state transitions. *HIPSM* module also handles changes in partner IP addresses (sets the locators by receiving and processing UDPATEs), but the actual storage happens in the main *HIP module*. Main methods of the module:

- *handleAddressChange*: If an ADDRESS_CHANGED message is received from the *HIP module*, *HIPSM* starts the UPDATE procedure in which an UPDATE message containing the current local locators will be sent towards the partners.
- *handleMessageLocalIn*: Handles packets received from the upper layers. If no HIP connection has been set up for a destination HIT of an incoming packet, then the method starts the HIP BE and stores this first message (*triggermsg*). If a packet arrives for a "BE in progress" HIP connection, then this packet will be discarded. After a successful BE every corresponding packet will be extended with an ESP header containing the appropriate SPI value.
- *handleMessageRemoteIn*: Deals with packets coming from the network. If a corresponding BE or UPDATE procedure is in progress, then this method will generate the appropriate answer messages. If no BE or UPDATE procedures are running for that particular packet, then the ESP packet will be decapsulated and the result will be passed to the *HIP module* for further processing.
- *handleCreditAging*: Implements the basic procedures of the HIP's Credit-Based Authorization (CBA) approach designed to prevent redirection-based flooding attacks. The method is called after receiving *creditMsg* periodical self messages and uses CBA *CreditAgingFactor* and *CreditAgingInterval* parameter values proposed in [10].

[1] HIPSim++: A Host Identity Protocol (HIP) Simulation Framework for INET/OMNeT++, Official homepage: http://www.ict-optimix.eu/index.php/HIPSim

- *createEspMessage*: Searches for appropriate SA for transport layer packets and encapsulates them into ESP messages marked with SPI. If no SA can be found for the application belonging to the arriving transport layer packet, then an UPDATE procedure will be started using the *handleAddressChange* method.

5.1.2 RvsHIP module

The *RvsHIP module* is derived from the *HIP module* in order to extend the basic HIP capabilities with the RVS functions by handling the incoming registration messages according to [16] and by forwarding I1 messages [11] to the appropriate HIP responder chosen from the registered ones. Main methods of the module:

- *handleMsgFromNetwork*: If the destination HIT of an arrived HIP I1 packet is the RVS's own HIT, then registration mechanisms are to be initiated and a new HIP SM is to be created. HIP SM daemons in the RVS are responsible for handling HIP UPDATE messages and corresponding procedures for registered HIP nodes. If the destination HIT of an incoming I1 differs from the RVS's own HIT, then it must be modified and forwarded towards the appropriate HIP node in the registration list according to [11].
- *alterHipPacketAndSend*: This method modifies the assigned I1 packet: a FROM parameter containing the original source IP address of the HIP packet will be added and the source IP address in the original IP header will be overwritten with the IP of the registered HIP node owning the destination HIT.

5.1.3 DnsBase module

The *DnsBase module* is a simple UDP application which realizes basic DNS server functionality for name resolution of HIP hosts and implements the new Resource Record (DNS HIP RR) defined in [15]. The module resolves domain names to HITs and IP addresses and in case of mobile HIP hosts also provides RVS information. Note that reverse DNS lookups are not supported in the current version of HIPSim++. Main methods of the module:

- *LoadDataFromXML*: Reads initial DNS database containing Resource Records of every host in the simulation from an `.xml` configuration file. Resource Records of a particular HIP host are within the `<DNSEntry>` tag where `<Address>`, `<HIT>`, `<NAME>`, `<RVS>`, etc. tags contain the different fields of a DNS HIP RR.
- *handleMessage*: Processes incoming DNS query messages and answers them by sending DNS responses with the appropriate Resource Records of the queried hosts.

5.1.4 PerappDecisionEngine module

The *PerappDecisionEngine* module implements the main functions of our per-application mobility scheme designed to operate in HIP environments. This module marks application sockets with quintets of Source HIT, Destination HIT, Source port, Destination port and Transport type, such providing the basic toolset for handling mobility in an application-wise manner. Main methods of the module:

- *calculateBestInterface*: Calculates the optimal interface for a given application profile based on the actual conditions. This interface will be used as the outbound interface for every packet of the application. Two application profiles (FTP, VoIP) are implemented (see Section 6 for further details).
- *recieveChangeNotification*: This method implements the Monitoring/Mediator Agent by handling INET's *NotificationBoard* messages coming from different protocol layers. In the current state of our simulation model socket information from the transport layer, IPv6 prefix information from the networking layer and link information from the link layer is handled. However control of different layers is also an important task of this method, current implementation does not take into consideration the possibility of such a behavior: no intervention into different protocol layers is performed after decisions.
- *PerappProfileDatabase*: This method implements APDB functions by reading and processing the initial profile data from `.xml` file.

5.2 Special Nodes in HIPSim++

HIP RFCs and Internet Drafts define three main types of nodes, namely the Initiator, the Responder and the Rendezvous Server. For introducing name resolution functions, also DNS server entity is to be used in a HIP architecture. All the above HIP nodes have been realized in HIPSim++ based on the existing INET modules and the newly introduced *HIP*, *HIPSM*, *RvsHIP*, and *DNSBase* modules. Functions of our HIP-based per-application mobility management scheme are implemented in the nodes by introducing the *PerappDecisionEngine* module.

5.2.1 Wired HIP Initiator/Responder (HipHosts6)

Wired hosts implementing HIP Initiator and/or Responder functions (i.e. HIP hosts) are derived from the INET's existing *StandardHost6* compound module by inserting the *HIP module* between the transport and the network layers. This node represents a basic HIP host with HIP mechanisms, HIP-based UDP/TCP applications but without support of mobility. The physical network interface is one Ethernet card, but in general any kind of network interface model can be used. HIP hosts contain a single instance of the HIP layer compound module which executes HIP procedures and creates HIP SM daemon instances for every HIP connection.

5.2.2 Wireless HIP Initiator/Responder with multiple interfaces

In order to exploit the multihoming capabilities of Host Identity Protocol, the number of physical interfaces of a HIP host can be increased freely in HIPSim++. *WirelessMultihomeHipHost6* is implementing HIP hosts with multihoming capabilities, where communication partners are continuously updated about the locator (i.e. IPv6 address) changes of all the active interfaces, and the most appropriate interface is used for data transmission. With a *PerappDecisionEngine* module introduced, per-application mobility management can be used in the node.

5.2.3 DNS Server (StandardHost6 with DNS)

A DNS Server node in HIPSim++ is responsible to provide name resolution for HIP hosts by implementing the basic functions described in [15]. DNS Server node is basically a *StandardHost6* compound INET module comprising also our DNS implementation called *DnsBase,* which runs appropriate DNS mechanisms. The DNS database used by our *DnsBase* module is an `.xml` file containing resource records of every node in the simulation topology. DNS queries are handled by the Host Identity Layer: the first transport packet initiates the query process based on the destination HIT (and the pre-set DNS IP

address), and the Basic Exchange starts right after the response provides with the locator belonging to that destination HIT.

5.2.4 HIP Rendezvous Server (RvsHost6)

RvsHost6 nodes implementing HIP rendezvous functions in our simulation framework are also derived from the *StandardHost6* compound module by interposing the modified HIP module prepared to handle RVS tasks (i.e. the *RvsHIP module*). *RvsHost6* node forwards I1 messages originated by (wired or wireless) HIP Initiators to the appropriate (wired or wireless) HIP Responder signed in the RVS. Therefore potential Responders must register themselves in the RVS and in place of their own IP address, Responders must use their RVS's IP address in the Domain Name System. Wireless HIP nodes must continuously inform their RVSs about events of locator changes.

5.3 HIPSim++ Messages

In this section we introduce the most important message constructions of our HIPSim++ HIP simulation framework.

5.3.1 HIP signaling messages

In accordance to [8], different HIP messages start with a fixed header. The HIP header is logically an IPv6 extension header such in HIPSim++ all HIP messages are implemented as additions to the INET's *Ipv6ExtensionHeader*. Almost all the already standardized HIP message types and parameters are defined in our framework, including also the Locator parameter which is realized as an array of *HIPLocator* structures. An important exception is the ESP_INFO parameter which is missing due to the simplified management of IPSec SPIs in our simulation model.

5.3.2 HIP data messages

In HIPSim++ we currently use the Encapsulated Security Payload (ESP) based mechanism for transmission of user data packets [3]. As proper implementation of all the cryptographic mechanisms in HIP is outside of the scope of our researches, we use only simplified Encapsulating Security Payload Header [18] mechanisms for distinguish HIP data packets based on SPIs. Every HIP data message travels in ESP: packets coming from the transport layer will be encapsulated in an *ESPHeaderMessage* labeled with the appropriate SPI value. Every *ESPHeaderMessage* has a special object per header to carry the SPI value as parameter. This object is derived from the *IPv6ExtensionHeader* class of INET in order to overcome some inflexibility issues of the existing IPv6 implementation and making the ESP packets to pass through the networking layer towards the HIP module.

5.3.3 DNS messages

The basic HIP namespace resolution functions are implemented using a simple query/response message pair called *DnsQuery* and *DnsResponse*.

5.3.4 Per-application messages

Five different messages are defined for realizing inboard signaling functions of our HIP-based per-application mobility management platform. PERAPP_NEWAPP indicates the appearance of a new application. Changes in socket descriptors of running applications are signaled with PERAPP_CHANGEDAPP. Exiting of a running application is indicated by PERAPP_DELETEAPP. If source address or interface of a running application changes, PERAPP_UPDATE is sent. PERAPP_LAST_IF_DOWN message stands for the event when the last usable interface was disappeared. All the above messages contain the ID of the default interface, and the affected applications and their assigned interfaces.

6. The Simulation Scenario

We used the above introduced INET/OMNeT++ model to construct a simulation scenario aiming to evaluate the proposed HIP-based per-application mobility management platform against the standard per-host multihoming mechanisms where only one SA is built-up between endpoints and only one interface is chosen for every application based on static priority. During our evaluation we only considered pre-defined QoS parameters of applications / access networks (i.e. static profiles) and dynamic connection discovery as inputs of handover decision. It means that no active measurements of available networks and actual application performances were used as decision supporting data during the simulated communication process. Two different applications with different QoS characteristics (i.e. QoS profiles) were specified with two different decision policies in our defined simulation scenario:

- FTP (based on INET's *TCPSessionApp*): The profile of FTP aims to receive as much data from the source as possible; therefore the policy of this application was defined to maximize the TCP throughput. In order to achieve this, the policy sets the FTP application to always use the network with the highest bandwidth among the available connections. The FTP application is a TCP Reno session of a downlink data transmission.
- VoIP (based on INET's *UDPEchoStream*): The main goal of the profile defined for VoIP is to minimize the handover latency, packet loss and end-to-end packet delay variation. Hence the policy of VoIP was specified to use the network with the lowest number of possible handovers and simultaneously communicating applications. The application itself is a 15 Kbyte/sec CBR stream with packet size of 80 byte.

Besides the above described two applications a third one was also introduced in order to generate background traffic for comparison purposes. The decision profile of this application is the same as the FTP's but here an UDP CBR stream is transmitted (also based on *UDPEchoStream*) with adjustable bandwidth and 1024 Kbyte packet size.

Figure 9 introduces the network topology used during our simulations. Communication was performed between the mobile HIP Initiator owning two interfaces for WLAN and UTMS and the wired HIP Responder on an Ethernet network. The simulated wireless environment consists of a 3G UMTS coverage providing 2.6 Mbps (downlink) bandwidth with 80 ms average RTT and two 802.11b WLAN access areas offering 11 Mbps maximum bandwidth and 30 ms average RTT. DNS and RVS nodes are operating in full compliance with the corresponding HIP RFCs. The HIP mobile travels on a fixed route signed by the white arrow and it takes 140 sec to get to the end of the path (i.e. the length of a complete mobile communication session is 140 sec). The first 20 sec is for the initiation (WLAN MAC association, IPv6 configuration, HIP RVS registration and BE, starting the applications, etc.) thus 120 sec can be considered as the measurement interval. For every measured parameter 25 simulations were run, each with different random seeds meaning that every plotted dot in our graphs was created as an average of 25 simulations. In our graphs "No perapp" signs the results gained by applying the standard HIP per-host multihoming procedures in

the introduced scenario (WLAN was prioritized before UMTS) while "Perapp" stands for the results of running our HIP-based per-application mobility management platform (with the introduced application and network profiles and policies).

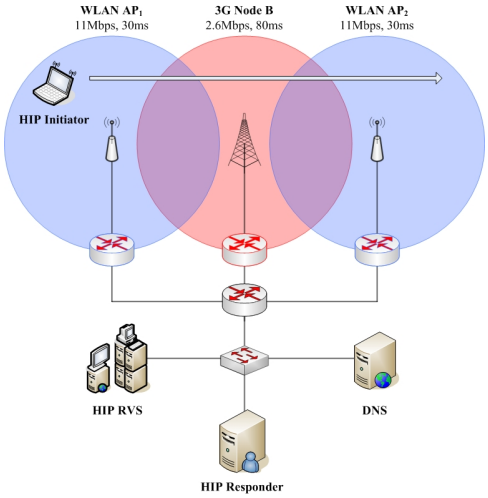

Figure 9. The used simulation topology

7. Results

Several performance measures were compared from every defined application profile's point of view. This section contains the results of the simulations. Standard HIP per-host multihoming is compared against our HIP-based per-application mobility management proposal.

Figure 10 and 11 shows the Jitter and RTT values of the VoIP application during a whole simulation run. The colored segments of the graphs are marking the actually used network(s).

Figures show that in cases of single network coverage (sole WLAN or UMTS) no big difference occurs between the two methods, but our HIP-based per-application mobility management platform overcomes the standard scheme when overlapping access networks are available. According to the pre-defined application policies and the dynamic network discovery our HIP extension assigns applications to interfaces in an optimal way resulting in low jitter and low congestion.

In Figure 12 results of total FTP throughput are depicted in the function of the background traffic. Every column stands for the average FTP throughput value of 25 complete simulation runs. It is shown, that the throughput decreases as the background traffic increases because of the congestion and the lost available bandwidth. However, the measured throughput is always higher when per-application mobility management is used, and the gain is getting more significant as the background traffic increases.

We have also analyzed the total amount of lost UDP packets of the VoIP application during a complete mobile communication session. Figure 13 presents that in cases of low background traffic no significant advantage of per-application mobility paradigm can be pointed out, but when high background traffic is injected in the network, the method shows its power. Thanks to the defined application policies VoIP always uses the network with less congestion and HO (here the UMTS) thus its cumulated UDP packet loss decreases compared to the standard HIP per-host multihoming (where WLAN is used when available).

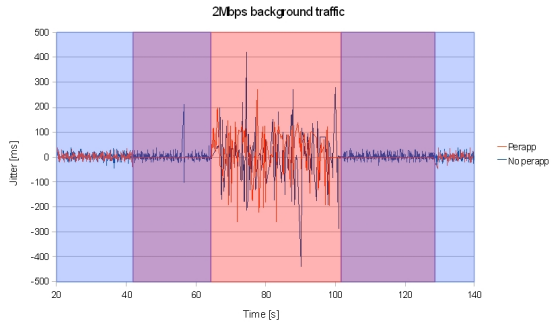

Figure 10. Graph of the measured jitter on VoIP

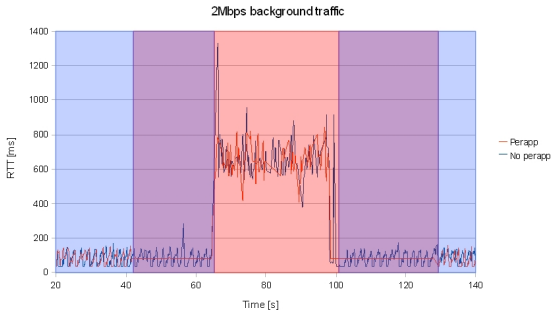

Figure 11. Graph of the measured RTT on VoIP

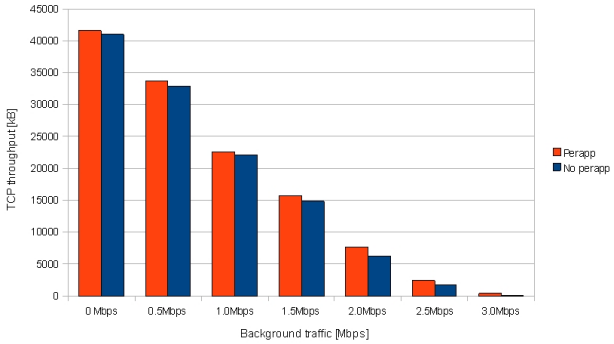

Figure 12. Cumulated TCP throughput of FTP

This behavior can be recognized also in Figure 14. where the average Jitter of the VoIP application is plotted. If per-application mobility management is employed, the variation of RTT values is smaller thanks to the optimally chosen and dedicated UMTS access network. This advantage expands with increasing background traffic.

However the dedicated UMTS connection of VoIP provides more stable communication in means of packet loss and Jitter, it also has some drawbacks. Due to the characteristics of UMTS the RTT of the VoIP application is higher in the per-application case than it is observable when standard HIP multihoming is applied and low background traffic is presented (Figure 15). Nevertheless, the Figure also shows that our HIP-based per-application mobility management platform still overcomes standard HIP multihoming performance if the network is loaded with higher volume of background traffic and the network (here the WLAN segment) gets more congested.

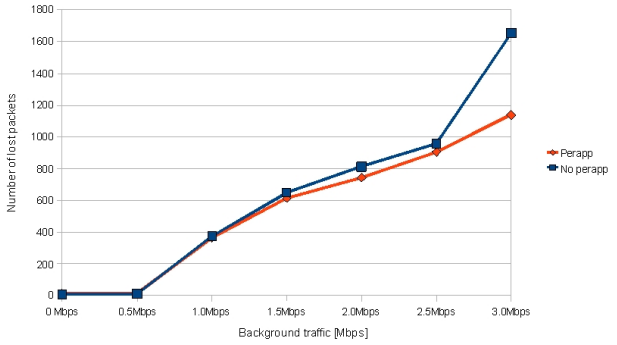

Figure 13. Cumulated UDP packet loss of VoIP

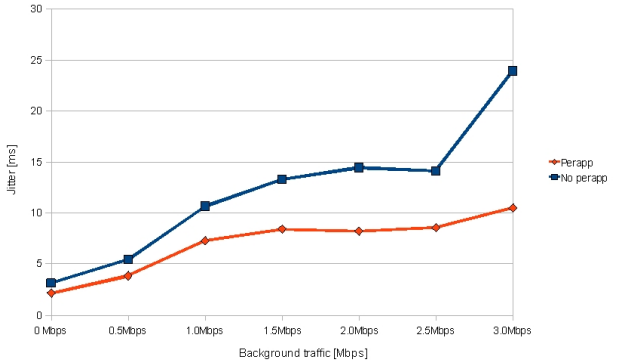

Figure 14. Average jitter of VoIP

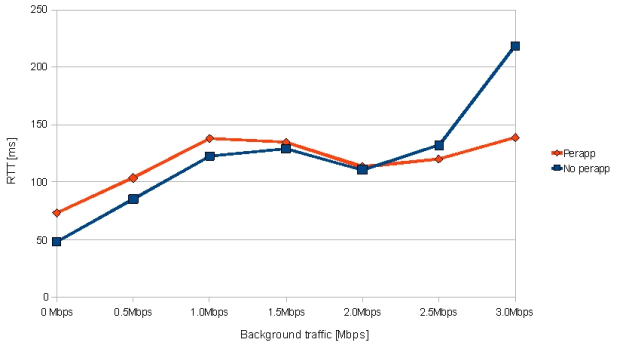

Figure 15. Average RTT of VoIP

8. CONCLUSIONS AND FUTURE WORK

In this paper we presented the design and evaluation of a HIP-based per-application mobility management platform which aims to be a comprehensive handover decision architecture founded on the promising Host Identity Protocol (HIP) and the cross-layer building blocks closely incorporating with it. In order to assess our platform against standard per-host HIP multihoming performance, we modeled HIP and the proposed per-application handover mechanisms in the INET/OMNeT++ simulation environment. Simulation results show that our HIP-based application-wise mobility system overcomes standard HIP mobility solutions in the most cases, even if the current protocol model does not comprise active network measurements and complex profiles and policies.

As a part of our future activities we will further extend our HIP-based scheme with more advanced cross-layer extensions, like dynamic application profiling; active, on-the-fly intervention into protocol layers' operation; and active measurements based interface selection.

9. ACKNOWLEDGMENTS

This work is supported by the ICT-OPTIMIX project which is partly funded by the 7th Framework Programme (FP7) of the European Committee. The authors would like to thank all participants and contributors who take part in the studies.

10. REFERENCES

[1] T. Koponen, A. Gurtov, P. Nikander, Application Mobility with HIP, in Proc. of ICT'05, May 2005

[2] T. Koponen, A. Gurtov, P. Nikander, Application mobility with Host Identity Protocol, Extended Abstract in Proc. of NDSS'05 Workshop, February 2005

[3] Vasantha Kumar, B.P.; Manjunath, D., Tunnel-accessed NATs for always-best-connected and application mobility, 6th International Conference on Information, Communications & Signal Processing, 10-13 Dec. 2007 pp: 1–5

[4] Ping Yu, Jiannong Cao, Weidong Wen and Jian Lu, Mobile Agent Enabled Application Mobility for Pervasive Computing, Ubiquitous Intelligence and Computing, Volume 4159/2006, pp: 648–657

[5] Moonjeong Chang, Hyunjeong Lee, Meejeong Lee: "A per-application mobility management platform for application-specific handover decision in overlay networks," Comput. Netw. (2009), doi:10.1016/j.comnet.2009.02.018

[6] Moonjeong Chang, Meejeong Lee, Hyunjeong Lee: "Per-Application Mobility Management with Cross-Layer Based Performance Enhancement," IEEE WCNC 2008., March 31 2008-April 3 2008 pp: 2822–2827

[7] R. Moskowitz, P. Nikander: "Identity Protocol (HIP) Architecture", RFC 4423, May 2006

[8] R. Moskowitz, P. Nikander, P. Jokela, T. Henderson: "Host Identity Protocol", RFC 5201, April 2008

[9] P. Jokela, R. Moskowitz, P. Nikander: "Using the Encapsulating Security Payload (ESP) Transport Format with the Host Identity Protocol (HIP)", RFC 5202, April 2008

[10] P. Nikander, T. Henderson, C. Vogt, J. Arkko: "End-Host Mobility and Multihoming with the Host Identity Protocol", RFC 5206, 2008

[11] J. Laganier, L. Eggert: "Host Identity Protocol (HIP) Rendezvous Extension", RFC 5204, April 2008

[12] F. Z. Yousaf; C. Bauer; C. Wietfeld: "An Accurate and Extensible Mobile IPv6 (xMIPV6) Simulation Model for OMNeT++", in the Proc. 1st Int. Conf. on Simulation tools and techniques for communications, networks and systems (SIMUTools2008), ISBN: 978-963-9799-20-2, Marseille, France, 2008

[13] Andras Varga, Rudolf Hornig: "An Overview of the OMNeT++ Simulation Environment", in Proc. 1st Int. Conf. on Simulation tools and techniques for communications, networks and systems (SIMUTools'08), ISBN:978-963-9799-20-2, Marseille, France, 2008

[14] OMNeT++: A public-source, component-based, modular and open-architecture discrete event simulation environment. Official homepage: http://www.omnetpp.org/

[15] P. Nikander, J. Laganier: "Host Identity Protocol (HIP) Domain Name System (DNS) Extension", RFC 5205, April 2008

[16] J. Laganier, T. Koponen, L. Eggert: "Host Identity Protocol (HIP) Registration Extension", RFC 5203, April 2008