A Policy System for Simultaneous Multiaccess with Host Identity Protocol

Sébastien Pierrel
NomadicLab
Ericsson Research
Jorvas, Finland
Email: first.last@ericsson.com

Petri Jokela
NomadicLab
Ericsson Research
Jorvas, Finland
Email: first.last@ericsson.com

Jan Melén
NomadicLab
Ericsson Research
Jorvas, Finland
Email: first.last@ericsson.com

Kristian Slavov
NomadicLab
Ericsson Research
Jorvas, Finland
Email: first.last@ericsson.com

Abstract— **In this paper we describe a Host Identity Protocol (HIP) extension that allows multihomed HIP hosts to use multiple access networks simultaneously. This extension defines how to identify data flow and how to route them based on higher level policies and specifically address the issue of the return path by transfering the policies to the peer.**

I. Introduction

In the early days of the Internet, hosts were big and clumsy and remained in fixed locations. The advances in technology has brought us light and small hosts that are portable. In order to maintain active connections to other hosts, the mobile host must be able to handle movements and inform other communicating parties that it has changed the topological location in the network.

One problem with the current Internet architecture is that the IP address is used both for describing the topological location of the host and, at the same time, to identify the host. The Host Identity Protocol (HIP) [1] is one proposal to solve this semantic overloading of IP addresses. HIP introduces a new name space, the Host Identity name space, where the host identities are cryptographic. The location information, i.e. the IP address, is used only for routing purposes, not to identify the host. The resulting architecture provides a simple, yet secure, way to provide mobility and multi-homing for end-hosts.

The basic protocols in HIP define the connection setup and IP Encapsulating Security Payload (ESP) usage with HIP [3]. The connection setup is a four-way handshake, a so called Base Exchange, during which the hosts authenticate each other and generate a shared keying material using Diffie-Hellman procedure. This association is called as the HIP Association. In addition to these, the usage of ESP with the HIP is defined. The required protocols are negotiated during the Base Exchange and the generated keying material is used in ESP Security Associations (SAs).

The Mobility and Multihoming specification [2] defines how the host mobility is handled and the current location information is provided to the peer host. The multi-homing is defined with the possibility to use simultaneously multiple access networks towards different HIP hosts.

The basic HIP Multihoming is limited to the Host based multi-homing: all connections between two HIP hosts use always the same path and when the multi-homed host changes the used access networks, all flows are transferred to the new interface.

In this paper we describe an extension to the Mobility and Multi-homing management that allows separation of different flows between two HIP hosts. Each flow can use different paths independent of each other. This provides new possibilities to use different kinds of connections over different types of access networks (e.g. streaming video uses the interface that has higher bandwith and control uses the most reliable connection with lower latency).

The separation of the location and identity information enables also moving between different IP versions as the transport protocols are bound to identities instead of locators. The changing of binding between identity and locator is transparent towards the transport layer.

The rest of the paper is organized as follows. In section 2 we describe shortly HIP as well as HIP mobility and multihoming support. In section 3 the simultaneous multiaccess is defined and the required extensions to HIP are introduced. Section 4 describes the implementation and section 5 gives some ideas how to continue the work. Finally, section 6 concludes the paper.

II. HOST IDENTITY PROTOCOL

A. HIP - Separation of Namespaces

If you are asked a question ”Who are you?” and you respond with your home street address, you are not actually answering the question. Nonetheless, the question is answered in an analogous way in the current Internet. When a host is identified, the IP address, providing the topological location of a node in the Internet, is given as the answer.

In real life, if you have to prove your identity and the asking person is unsure, you show your ID-card. Respectively, if you are asked to give your address, you will give the street address providing your (home) location. Using this analogy in the Internet, the host identity and location information must be separated from each other. HIP provides one possible solution for decoupling the location from the identity.

Each HIP [1] enabled host has identities, one or more, long-term or short-term, that can be used to identify it in the network. In HIP, the identifier is the public key of a public-private key pair. When the host possesses the private key, it

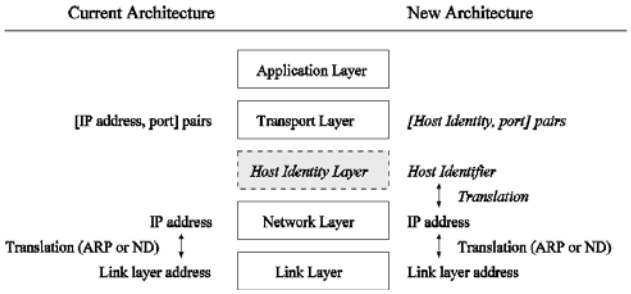

Fig. 1. New host identity layer

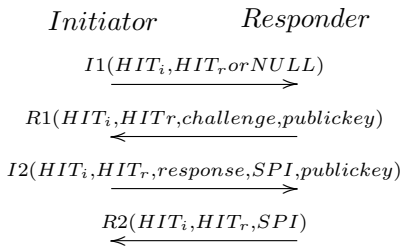

Fig. 2. HIP Four-way Handshake

can prove that it actually "owns" this identity that the public key represents. This can be compared to showing an ID-card.

Each host can generate short-term keys to be used only for a short time. These are handy when it is not necessary for the node to be identified with the same identity later. For example, buying books from a bookstore may be a long-term relationship, while once contacting a server that may collect user profiles may be considered to be a short-term action where the long-term identity is not wanted to be revealed.

The HIP Host Identity (HI), being a public key, is not practical in all actions; it is somewhat long. In HIP, the HI is represented with a 128-bit long Host Identity Tag (HIT) that is generated from the HI by hashing it [7]. Thus, the HIT identifies a HI. Since the HIT is 128 bits long, it can be used for IPv6 applications directly as it is exactly the same length as IPv6 addresses.

When HIP is used, the upper layers, including the applications, do not see the IP address any longer. Instead, they see the HIT as the "address" of the destination host. The location information is hidden at a new layer, introduced between the Transport and Network Layers 1. The IP addresses no longer identify the nodes; they are only used for routing the packets in the network while the HI is used as the identity. Mapping between identities and locators is done at the new layer.

B. Establishing Connection Between HIP Hosts

HIP defines a base message exchange (Base Exchange) containing four messages, a four-way handshake. During the message exchange, the Diffie-Hellman procedure is used to create a session key and to establish a pair of IPsec ESP Security Association (SA) between the nodes [3].

Figure 2 shows the four-way handshake. The negotiating parties are named as the Initiator and the Responder. The Initiator, before initiating a connection, resolves the identity and locator for the peer host. This information is used to generate the negotiation initializing message, I1. The I1 packet contains both hosts' HITs and IP addresses. It is possible that the destination HIT is zeroed when the Initiator doesn't know the HI of the peer. This connection initialization is called Opportunistic Mode.

When the Responder gets the I1 packet, it does not perform any deeper check of the message: it just responds to the message with an R1 message. At this stage the Responder does not create any state for the Initiator, i.e. it just forgets that it has received an I1 message from some host. The most important content in the R1 message is the puzzle that the Inititator has to resolve and provide answer to the Responder before the connection can be established. The puzzle method ensures that the Responder can control the required work that the Initiator has to perform before connection establishment. This provides some protection against Denial-of-Service attacks. In addition, R1 initiates the Diffie-Hellman procedure for establishing shared keying material between the hosts.

Once the Initiator has resolved the puzzle and authenticated the Responder using the public key information and signature included in the R1 packet, it creates an I2 packet containing the result for the puzzle, as well as its Diffie-Hellman parameters and other required information (e.g. Security Parameter Index, SPI) to establish the ESP Security Association. After verification of the puzzle solution and performing Initiator authentication based on the public key and signature included in the packet, the Responder creates the required keying material using Diffie-Hellman shared key and establishes ESP SAs. It also creates the final HIP message, R2, that contains the missing information to establish ESP SAs. After receiving R2 message, the Initiator can finalize the ESP SAs.

C. HIP Mobility and Multihoming

With HIP, the separation between the location and the identity information makes it clear that the packet identification and routing can be cleanly separated from each other. The host receiving a packet identifies the sender by first getting the correct key and then decrypting the packet. Thus, the IP addresses that are in the packet are irrelevant.

1) Rendezvous Server: A HIP mobile node, moving in the network, may constantly change the point of topological attachment to the Internet. When the point of topological attachment is changed, the IP address changes too. Like in MIPv6, this changed location information must be sent to the peer nodes. The DNS is too slow to be used for constantly changing location information. Therefore, there must be a more stable address that can be used to contact the MN. In HIP, a Rendezvous server (RVS) is specified to handle initial connection attempts towards a mobile node. The RVS maintains the mapping between its client HIT and client's IP address.

In DNS, the mobile node maintains the mapping between the host name, HI and its RVS's IP address. When some host initiates a connection towards the mobile host, it resolves the HI and location information from the DNS and gets the IP

address of the RVS. The connection initialization packet, I1, is delivered to the RVS. The RVS, maintaining the mapping between its client's HI and current IP address, forwards the I1 packet to the mobile host [4].

2) Location Update Mechanism: When the mobile node moves, it needs to sends the new location information, i.e. the IP address, to all peer nodes that it is communicating with and, in addition, to its RVS so that new connection initializations can be done via the RVS. To accomplish this, the HIP Mobility and Multi-homing protocol [2] defines a three message exchange that updates the addresses of the mobile node to peer nodes.

After receiving a new address, the mobile node creates an UPDATE packet with a LOCATOR parameter containing the new IP address. It sends this packet to the peer node, which verifies the signature and validity of the message. From the peer point of view this is still not enough for updating the mobile node's location information; the IP address information can be false, either due to a mistake or on purpose. The peer has to initiate an address verification process.

The peer creates an UPDATE packet containing an ECHO_REQUEST parameter in addition to the ACK parameter that acknowledges the receiving of the original UPDATE message and sends it to the mobile node as a response. The ECHO_REQUEST contains some random data that the peer host expects to receive back from the mobile node in order to complete the address verification.

After receiving the ECHO_REQUEST, the mobile node generates and sends the final UPDATE message containing acknovledgement to the received UPDATE message and an ECHO_REPLY parameter where it inserts the same random data that was in the received ECHO_REQUEST parameter. Once the peer receives this packet, it can start to use the new address.

The specification defines also an optimization that allows the peer node to start using the mobile node's new IP address already once it has received the first UPDATE packet. It makes the address verification but, at the same time, it is allowed to use the new address for a limited amount of data.

Because the MN can move between networks using different IP address versions, the address received by the CN may also be from a different address family than the previous address. This is not a problem because using different address families is supported in HIP. However, if there is a case when both hosts are in access networks that support only different address families, a proxy node is required between the hosts. The proxy provides a "virtual interface" for one of the nodes and makes address translation between for the traffic.

3) Multihomed Host: Multihoming handling does not differ much from the mobility management. The multihomed host has to inform the current set of IP addresses to the peer nodes so that they can use them. In general, the location information update procedure is similar to the one used in mobility case. Based on the location information exchanged, hosts create security associations between them.

The HIP multihoming [2] defines how the security associations are created per interface. The mechanism is the same as for mobility, thus the SAs are created using the same three packet exchange as in the mobility case with addition that two first packets contain the new security parameter index (SPI) values to be associated to this set of addresses.

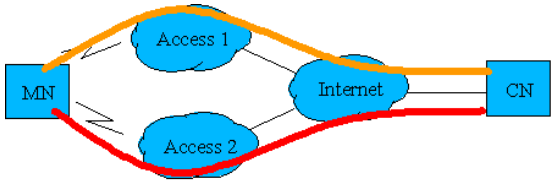

Fig. 3. Mobile node simultaneously connecting to a correspondant node on the Internet from multiple access networks

The HIP multihoming can be seen as a special case of mobility in this respect. The current specification does not define how these security associations should be used just how those are created. In this paper we are describing how the flows could be separated to smartly use all available interfaces for different types of flows.

III. SIMULTANEOUS MULTIACCESS

A. Theory

As illustrated by figure 3, a multi-homed host can be connected to multiple different access networks simultaneously, thus having more than one network interface for sending and receiving data traffic. The multi-homed host may want to define the usage of the available networks based on reliability, speed, cost, or other information.

Two different cases for multiaccess can be identified. Firstly, the multi-homed host can use the same access network for all connections between itself and one peer. For connections between itself and another peer host, a different interface can be used. Secondly, the multi-homed host can separate connections between itself and a peer host so that each of the connections can use different access networks and they can be transferred to other access networks independently from each other.

In the first case, the peer host does not have to know about the multi-homed hosts capability to use multiple networks. It can see only that all connections are using the same IP address. When the multi-homed host changes connections that are going towards this peer to use another interface, the transfer can be done using simply mobility management messages. When the peer receives a location update message from the multi-homed host, it starts using this new address.

In the second case, however, the peer node has to be able to make decisions when it selects the destination IP address for packets going to the multi-homed host. Otherwise it cannot route the traffic using the connection that the multi-homed host wants to use for certain traffic.

The following subsections describe how hosts can identify different flows from each other, what information is required at end-hosts and what kind of policies we are using to make the actual routing decision.

1) Identifying flows: A socket can identify a data flow on the application layer but on lower layers this entity becomes a set of parameters that depends on the protocols being in use on the network and transport layers. A legacy TCP/IP socket is bound to IP addresses on the network layer whereas in a host having Host Identity Protocol support sockets are bound to HITs. The protocol numbers and related parameters remain still the same. Transmitted packets are encapsulated with ESP when sent on the network making it impossible to identify the flow where the packet belongs to while the packet is traversing the network. Packets can only be identified to belong to an ESP Security Association from the SPI value in the ESP header.

This paper focuses on the most common protocols used by applications like TCP, UDP, SCTP and DCCP, which all identify the connection by source and destination port numbers. These numbers are unique between two hosts and are suitable to be used for identifying flows.

2) Policies: The host having multiple interfaces has to have a set of rules to describe the usage of the available networks. These rule sets are called policies. The multi-homed host uses the set of rules to select the correct SA for the outgoing packet and the packet handling is done as described in the previous subsection.

Because the policy set is defined at the multi-homed host and the peer host has to be aware about both the set of policies and the network interface configuration at the multi-homed host side, the required policy information needs to be trasmitted from the multi-homed host to its peer.

At the peer host, the policy received from the multi-homed host determines which SA each of the individual data connections should be using. When the peer host is sending data to the multi-homed host, it verifies to which SA the flow is mapped. Based on that information, the host delivers the packet to the network using the IP address associated with the particular SA.

All the flows leaving a multi-homed host are protected by ESP. At the multi-homed host, a separate ESP Security Association (SA) is set up between the multi-homed host and the peer host for each of the outgoing interfaces at the multi-homed host. Thus, selecting the outgoing interface is done by selecting the correct ESP SA. HIP needs to send its policies to the ESP layer so that packets are encapsulated accordingly.

When a packet goes down from the transport layer, the ESP layer will select the outgoing SA according to the policies preset by HIP and the port and HIT information in the packet. Once this is done and the matching outgoing SA is found, the packet is encapsulated and sent to the peer using the interface that the selected ESP SA defines.

As a side effect of the multi-access, the multi-homed host's routing table contains more than one default route, which would mislead a standard IP stack. Therefore it should be noticed that outgoing IP packets are sent out to the appropriate router. This is achieved by selecting the route from the source address field of the IP packet, which the ESP processing filled with the address of the interface according to the SIMA policies.

At the peer side a similar selection is made for outgoing traffic. In that case, however, the selection defines the destination IP address which in turn defines the route that the packet will take while traversing from the peer host to the multi-homed host.

B. HIP Extension

In the basic HIP or in HIP mobility support specification there is no defined mechanism to transmit the required policy and interface information between the hosts. However, this can be achieved by defining a new extension in HIP that can be used to transmit all required information, including SPI and the flow identifier triplet (source port; destination port; protocol number).

HIP defines an UPDATE message exchange which can be used to update e.g. security association information or location information. The base HIP specification [1] defines only the UPDATE skeleton and other specifications are free to use that for their own purpose. We have defined a new exchange, called POLICY UPDATE which is used to deliver the specified information from the multi-homed host to its peer hosts.

In figure 4 the new POLICY parameter, containing the required information, is presented. One UPDATE message can contain as many POLICY parameters as there are SAs in the context. Every POLICY parameter bears the flow identifier(s) bound to a given SA, identified by its SPI. The option *Seq_num* is used by the peer to acknowledge the policy update. The bits of the field *Reserved* are unused.

C. Policy Exchange and Usage

1) Creating policies : Policies are rules that define the usage of the available interfaces. We are not interested in how the sets of policies are transmitted to the host; they can e.g. be given by the operator or they can be configured by the user. The location of policies in a node is implemenation dependent. In our implementation policies are stored in a database called the SIMA database and is part of the process managing HIP sessions.

In our proposal, an Application Programming Interface (API) is defined for communicating the policy information to the underlying system. The API is used by applications and they give the required information to the kernel for making the correct routing decisions. Typically, an application will give the list of preferred interfaces to the system with the call:

```
sima_policy_add(flow f, array interfaces);
```

$flow f$ is an object determining the flow for which the policy will be added to the system. This can either be a socket file descriptor or a collection of parameters. The API makes it also possible to define policies for flows that do not presently exist, i.e. when a socket is not yet created. For that purpose, some parameters of the flow identifier (as described in §III-A.1) can be null. This kind of policy can match multiple flows and will be applied to those that do not match a more specific policy.

Alternatively, the API provides a system call to remove a policy from the system:

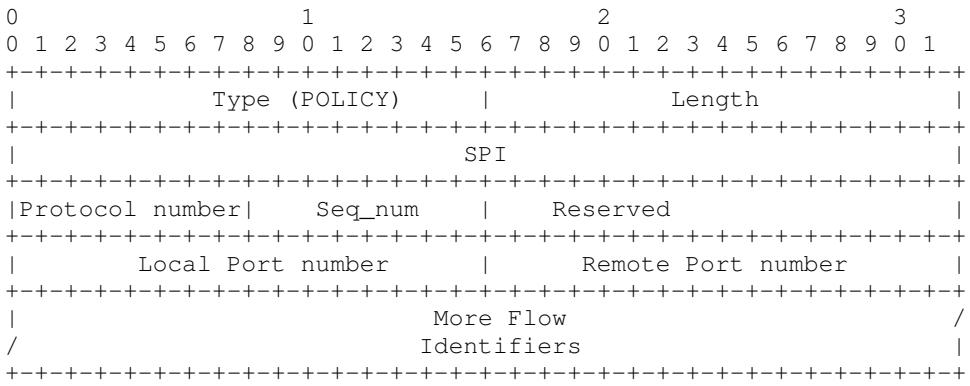

Fig. 4. POLICY parameter of an UPDATE packet

```
sima_policy_del(flow f);
```

and yet another call to get the list of policies currently in use by the system:

```
sima_policy_list();
```

which returns an array of policies.

Every HIP context creates an instance of policy set from the SIMA database in order to apply the rules to the underlying IPSec system. A policy received from a peer is added to the context database associated to the peer. This ensures that a remote policy cannot affect nor modify policies regarding other peer hosts.

2) Sending policies: Initially, the multi-homed host negotiates a HIP association with the peer host. Once the HIP association has been created, the multi-homed host creates policy information from the context SIMA database. This information is included in the new POLICY parameter and the multi-homed host initiates a POLICY UPDATE message exchange.

The peer host receives the policy information and acknowledges it to the multi-homed host. If the peer does not support policies, it just ignores the packet. The multi-homed host may resend the packet, but finally it has to decide against using the multi-access features with this peer node. In case it supports SIMA, the peer should replace the SIMA context database with those embedded in the received POLICY UPDATE message.

As an enhancement, the policy information could be embedded already in the HIP base exchange since the I2 and R2 messages contain sufficient information to anticipate the SPI values.

3) Using policies at the multi-homed host: When there are changes in the sets of available access networks, the information is sent to the peer host in location update messages. The multi-homed host transfers the policies to the peer host, so that the peer can send packets using the correct destination address in all cases. Both end-hosts must have a common view how the different flows are mapped to security associations that are created by the multihoming of HIP.

Policies are updated after the mobile node has finished the location update exchange and the address has been verified to be reachable. The updating is done with a single round-trip re-using the UPDATE mechanism defined by the HIP base protocol [1] containing the POLICY UPDATE parameter described in §III-B

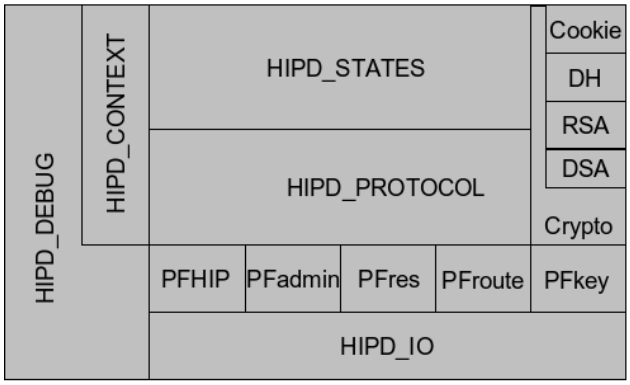

Fig. 5. hip4inter.net HIP daemon layers

IV. Implementation

Our HIP policy implementation is implemented on top of the hip4inter.net FreeBSD HIP implementation [8]. The hip4inter.net implementation is one of the most mature versions of HIP containing the basic HIP four-way handshake, mobility and multihoming. The hip4inter.net implementation is easy to extend as the HIP protocol is run as a user space daemon and it uses the PF_KEY [5] socket to control the IPsec policies and security associations. The hip4inter.net HIP daemon code has been written using the layered principle as the base of the implementation the figure 5 presents the different layers of the code. The basic HIP implementation consists of roughly 18000 lines of code.

The identity and locator separation in hip4inter.net implementation has been done using a new IPsec mode called "Bound End-to-End tunnel (BEET) mode ESP" [6]. The BEET mode looks similar as a transport mode ESP but semantics is more like in tunnel-mode ESP. The ESP packet contains only the IP addresses of the end-hosts, just like in transport mode, but the HITs of the end-hosts are actually used as the inner addresses. They are just not transmitted over the network.

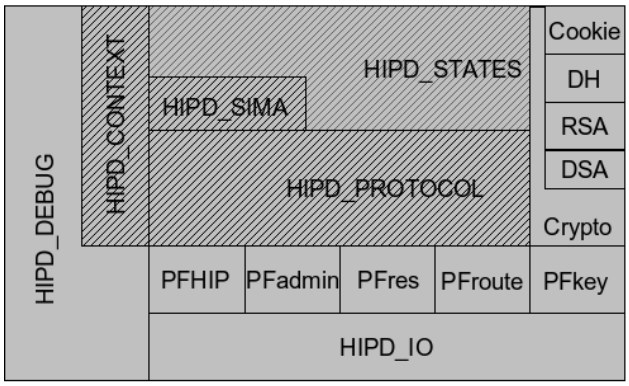

Fig. 6. the extended HIP daemon layers

At the receiving end-host, the SPI is used to find the correct ESP SA. Once found, the IP addresses are converted to the matching HITs and packet is further processed. In BEET mode the policies will contain the identifiers which are mapped to actual locators that are used to transport the packet to the receiver.

The HIP daemon listens the kernel's routing socket to detect when the interface has been brought up, a new address is assigned to an interface, or if an address has been removed from the interface. Once such an event is received through the routing socket, the mobility update mechanism is invoked. The policy updating is also triggered at the same time as described later in this document.

A. Implementation details

As stated before the implementation is based on the hip4inter.net code extending it to include multihoming policies which allow us to separate the traffic through multiple interfaces. The extension required roughly 2000 lines of code which included a new API for applications to set list of preferred interfaces, logic for installing multiple IPsec policies through the PF_KEY interface, and creation and parsing of new policy update HIP protocol message. As the basic HIP daemon followed the layered structure we also decided to follow it and the figure 6 presents the structure with the policy implementation. The hatched parts are the modified and new ones.

Simultaneous multiaccess (SIMA) API extends the normal socket API by adding calls which a SIMA-aware application can use to get the list of available interfaces and set the list of preferred interfaces. The application may also be interested in knowing when network interfaces are added or removed. The application can subscribe to a notify message which will be delivered to the application whenever an interface either becomes available or unavailable. Each application is able to set and update a list of preferred interfaces, which it would like to be used when communicating with the peer identity. In our prototype the application sets the list of interfaces in the order of the preference according to some characterics (for example an application with adaptive bandwith usage will prefer the higher bandwith interface over the reliable, but slower connectivity). Providing characteristics of the network interface is beyond the scope of this document. When the application sets a list of preferred interfaces, the protocol, source port and destination port used by the application are recorded to distinguish the application preferences from other applications preferences. Each instance of application is identified by combination of transport protocol, source identifier, source port, destination identifier and destination port. The implementation also supports wildcarding of some of the parameters in which case the matching policies are ordered so that the policy with most wildcarded parameters has the lowest priority.

The HIP daemon can register these preferences at any time but will act accordingly when a HIP context is established with a peer. In order to establish this context, the hip daemon performs the standard Base Exchange with one network interface. At this point both the initiator and responder will install one default policy, which can be used to start the commnication between them. Next the initiator checks whether it has more interfaces that could be used to communicate with the peer node. If there are more interfaces available it will try to create security associations between them using the mobility and multi-homing update mechanism of HIP protocol.

After all security associations have been created the updating of policies is started. The policy update will require one extra round trip from initiator to the responder. In the first message the initiator sends the policy update message containing all the source, destination and SPI mappings. The responder will acknowledge the policies telling which of the policies were successfully accepted and installed through the PF_KEY interface. If the policy is not accepted the default policy will be used that source, destination port pair. After receiving the acknowledgement message from the responder the initator will install the policies that were accepted by the responder and start using them for communicating with the peer node.

Every time the node runs the mobility update exchange it has to update the policies regardless of whether the update exchange was successful or not, requiring one extra round-trip after the mobility update.

B. Practical tests

The testing of this prototype was done in live GPRS and wireless LAN networks. The scenario was such that there was adaptive video conferencing software and a high bandwith data download running at the same time. The download always preferred the wireless LAN network with a flat rate subscription and the video conferencing preferred the GPRS connectivity as the connection was more reliable on the urban area and not just limited to hotspots with variable bit rate.

The testing verified that the concepts described in this paper improve the mobile node's capacity to control the network access usage, enabling us to use bandwith available on different networks, allowing us to differentiate the different types of traffic to different interfaces and letting the application always select the interface that would best suite its needs.

Finally, as the prototype always had at least one interface up and running, the user was able to communicate with the peer at all times making the user experience of the network reliability better than what it is in the plain TCP/IP network where all the connections are interrupted everytime the node moves or experiences any kind of network outage (even short interrupts).

V. Future work

The prototype has been tested only in a small network and we are planning to run tests in higher traffic networks with more nodes involved to evalutate the performance of the system.

In the next version of prototype the policy update exchange should be combined with the normal base exchange and location update messages. This change would reduce the number of round trips needed to be taken before all the security associations have been set-up.

Moreover, the policy transfer method could be optimized so that the whole set of policies would not be transferred everytime, but only the difference.

VI. Conclusion

The prototype described in this paper is a proof of concept for simultaneous access. Flow identification is an important part of simultaneous multi access and HIP makes the handovers easy due to the fact that data flows are bound to the stable host identifier. The policy transfer to the peer presented here as the solution to the return path problem, is so closely related to the mobility management protocol that it would be difficult to define a generic SIMA protocol for different mobility protocols.

This sytem requires a higher level system to automate the creation of policies. In order to get efficiency out of the multiaccess, a system evaluating the characteristics of the paths from the hosts to the core network would also be needed.

Acknowledgment

Sébastien Pierrel and Petri Jokela were sponsored by the TEKES-funded Multiaccess Experimentations in Real Converging Networks (MERCoNe) project. The authors would like to thank all the partners who were involved in the MERCoNe project.

Jan Melén and Kristian Slavov are partly funded by Ambient Networks, a research project supported by the European Commission under its Sixth Framework Program. The views and conclusions contained herein are those of the authors and should not be interpreted as necessarily representing the official policies or endorsements, either expressed or implied, of the Ambient Networks project or the European Commission.

References

[1] R. Moskowitz, P. Nikander, P. Jokela, T. Henderson, *Host Identity Protocol*, draft-ietf-hip-base-06.txt, Internet-Draft, work in progress, IETF, June 2006

[2] T. Henderson, *End-Host Mobility and Multihoming with the Host Identity Protocol*, draft-ietf-hip-mm-04.txt, Internet-Draft, work in progress, IETF, June 2006

[3] P. Jokela, R. Moskowitz, P. Nikander, *Using ESP transport format with HIP*, draft-ietf-hip-esp-04.txt, Internet-Draft, work in progress, IETF, Oct 2006

[4] J. Laganier, L. Eggert, *Host Identity Protocol (HIP) Rendezvous Extension*, draft-ietf-hip-rvs-05.txt, Internet-Draft, work in progress, IETF, June 2006

[5] D. McDonald, C. Metz, B. Phan, *PF_KEY Key Management API, Version 2*, RFC 2367, July 1998

[6] P. Nikander, J. Melén, *A Bound End-to-End Tunnel (BEET) mode for ESP*, draft-nikander-esp-beet-mode-06.txt, Internet-Draft, work in progress, IETF, August 2006

[7] P. Nikander, J. Laganier, F. Dupont *An IPv6 Prefix for Overlay Routable Cryptographic Hash Identifiers (ORCHID)*, draft-laganier-ipv6-khi-05.txt, Internet-Draft, work in progress, IETF, September 2006

[8] hip4bsd, HIP implementation, http://www.hip4inter.net