UPMT: Universal Per-application Mobility management using Tunnels

Marco Bonola, Stefano Salsano, Andrea Polidoro
Dip. Ingegneria Elettronica, University of Rome "Tor Vergata"

***Abstract*—In this paper we describe an application level Mobility Management mechanism for IP networks, called UPMT (Universal Per-Application Mobility management using Tunnels). The mechanism is able to provide Vertical Handovers over heterogeneous IP based access networks on a per-application basis, i.e. it is possible to independently route different applications over different access networks and take separate handover decisions for each application. UPMT is able to support legacy applications, does not require any support from the access networks nor any change to the TCP/IP stacks in the Mobile Hosts (MH), is able to run on NATed access networks that provide private IP addresses to MH and is fully transparent to Correspondent Hosts. UPMT relies on tunneling the IP packets from the MH to an Anchor Node on IP/UDP tunnels. UPMT provide the MH and the applications with a "virtual" NAT service across many different physical access network. The paper provide the specification of the tunneling architecture and of the mobility management signaling, based on SIP protocol. The Open Source implementation of UPMT for Linux OS is ongoing and its status is presented.**

***Index Terms*— Application Level Mobility, Mobility Management, SIP protocol, Vertical Handover**

I. INTRODUCTION

The support of mobility in IP networks is at the same time a topic of great practical interest and a widely explored area of research. Cellular networks are now offering high rate IP based data connections, Wi-fi coverage is almost ubiquitous in office environments and is becoming more and more wide spread at home and in "hot spots". Other wireless technology like WiMax in the outdoor environments or Bluetooth in PANs (Personal Area Networks) complete the scenario. Multimode devices are now able to connect to several wireless (and wired) access technologies. The capability to move from one access technology to another one, switching also the active connections, is typically referred to as "Vertical Handover".

Several mobility solutions support vertical handovers at different levels of the protocol stack (e.g. from layer 2 up to the application level), see [1] for a survey.

As we will see in the next section, there are several requirements that can be considered when defining a solution for mobility management (or when evaluating a defined solution). In this paper we add the requirement that it should be possible to transport different application on different access technologies at the same time, and take separate handover decisions for each application if needed. As an example use case, consider a mobile device connected to an office Wi-fi network. The device is performing a background file transfer (e.g. to backup the device data on a backup server) and at the same time the user is engaged in a video conference. If the user physically moves out of your office, loosing the Wi-fi connectivity, she would like her video conference to be handed off in a seamless way on a 3G high speed data access, but likely it is not worth handing over the background data transfer as well (for both cost and performance reasons). We believe that this requirements has been overlooked or cannot be fulfilled in most of the available solutions, as was also pointed out by Chang et al. [2]. The per-application mobility management solution proposed in [2] require the cooperation of both communicating hosts that must be equipped with the new solution.

In this paper we describe an application level mobility management solution called UPMT (Universal Per-application Mobility management using Tunnels). We worked out this solution starting from our past work called MMUSE (Mobility Management Using Sip Extensions) [3], [4]. While MMUSE was only targeted to provide mobility support to real time SIP based applications (e.g. VoIP, video conferences), UPMT targets all types of application (using both TCP and UDP). UPMT inherits from MMUSE the use of SIP for signaling, while introduces a different transport mechanism based on a set of tunnels between the Mobile Host and an "Anchor Node". The Anchor Node provides a "virtual" NAT services to the MH across the set of access networks and physical NAT boxes. This way, it is possible to interact with all legacy hosts and applications.

In section II we describe the scenario and requirements, section III and IV respectively describe the terminal and application architecture and the networking architecture. Section V, 0 and VII describe the signaling and data transfer procedures, first at abstract level and then going into protocol details. Section VIII discusses some performance issues related with the proposed solution. Finally, in section IX we report the status of the open source implementation or UPMT and in section X we discuss the extensions to UPMT that we are considering to overcome its current limitations.

This work was supported in part by the EU under the project FP7 – 224024 "PERIMETER"

II. Scenario And Requirements

The reference scenario is depicted in Fig. 1. A Mobile Host can be connected through different access networks, either wireless or wired (e.g. Wi-Fi, Bluetooth, GPRS/EDGE, 3G/HDSPA, WiMax, fixed Ethernet). Most of these access networks provide private IP addresses to the Mobile Host and connect to the Internet through NAT boxes (Local NATs).

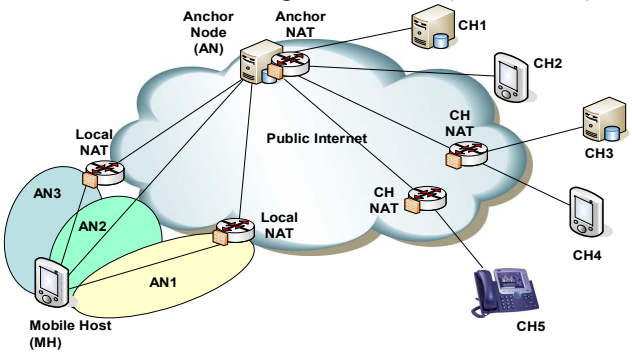

Fig. 1. UPMT Reference scenario

The Mobile Host will connect to "Correspondent Hosts" (CH) that can be either on public IP addresses (e.g. CH1 and CH2) or behind NAT boxes (CH-NAT, e.g. CH 3-4-5). The Correspondent Host can be either a "server" e.g. a web server or a "client" e.g. a VoIP client. The mobility services are provided by an "Anchor Node" (AN). The AN provides the Mobile Host with a second NAT service (Anchor-NAT), which is a key element in our mobility architecture. In fact, the idea is that the Mobile Host will always access the Internet through the NAT in the Anchor Node. The UPMT mobility management procedures will allow the MH to select the communication channel towards the Anchor Node, if needed in a separate way for each application/flow to be supported.

In our previous work MMUSE [3], we have analyzed a set of requirements for a mobility management solution that was targeted to real-time applications using SIP protocol (e.g. VoIP and video conferencing clients). Interesting enough, that list of requirements is basically still valid, with the main addition of requirements 1-3 below, while requirements 4-10 were already introduced in [3] We anticipate that the current definition and implementation of UPMT is able to fulfill all the requirements, but the sentences reported in italics in Table I. For these issues, we have clear ideas on how to extend the UPMT solution to support them, as reported in section X.

We note that the UPMT mechanism can be seen by the Mobile Host (and by the applications therein) exactly as a NAT service [5]. In principle, all the applications that can be run behind NAT boxes can also run using UPMT (and we can control the type and the behavior of the NAT in the Anchor Node). In particular UPMT can provide all the "connectivity services" that are offered across NAT boxes. In this paper will focus on dynamic PAT (Port Address Translation) or NAPT (Network Address Port Translation) , but it is also possible to realize the so called "static NAT" or "port forwarding" services.

TABLE I
REQUIREMENTS FOR THE MOBILITY MANAGEMENT SOLUTION

1) The Mobility Management solution should support all types of applications (not only the SIP based ones as in MMUSE). Both TCP and UDP based application needs to be supported.
2) Legacy application must be supported on the mobile host side, i.e. we should not require applications to be rewritten/recompiled to run on the mobile hosts and exploit mobility features. *On the other hand it should be easy to deploy new applications or enhance existing applications to fully exploit the capability offered by the Mobility Management solution.*
3) The Mobility Management solution should be able to separately deal with different applications and decide different handovers for each one based on a set of criteria like for example: cost; QoS/QoE parameters like throughput, loss, delay/jitter; security… (N.B. A discussion of the criteria is out of scope of this paper).
4) The Mobility Management solution should not require additional support in the access networks. The access networks are only required to provide IP connectivity. The capability to offer Mobility Management services should not be an exclusive prerogative of the network operators. *On the other hand, it should be possible for network operators to provide added value and enhanced services, by exploiting their network and servers assets.*
5) The Mobility Management solution should not require any modifications to the Correspondent Hosts, nor to the applications running in the CHs. All existing terminals should be able to interoperate with roaming MHs. *On the other hand, if the Correspondent Host / Correspondent application is aware of the Mobility Management solution, this should be exploited to provide a more efficient solution.*
6) The Mobility Management solution should be compatible with NAT traversal. Users should be able to roam from an access network to another, even when one or both access networks use private IP addressing and are behind a NAT. *Terminals that are not connected behind a NAT or that are in the same access network should be able to communicate directly, wherever possible.*
7) *The Mobility Management solution should be able to scale by distributing any server side capability into a set of nodes, taking into account load sharing issues as well as optimization of forwarding paths.*
8) It should be possible to hide the actual location and the movements of the user should to the CH, in order to preserve the privacy of the users.
9) When switching from an access network to another one, the Mobility Management signaling should be sent over the new target network, since the old one could suddenly become unavailable; in such a case it is necessary to perform the whole handover procedure on the new network (this is known as "Forward" handover or "break before make"). On the contrary, if the old network is still available, the availability of both networks can be exploited in order to assist and speed up the whole procedure ("make before break").
10) The vertical handover must be as fast as possible. This means that the user should not perceive any service interruption. If it is not possible to completely hide the effect of the handover, then the service disruption should be minimized.

The problems may come from the "local NATs" depicted in Fig. 1, i.e. the NAT boxes in the path between the local access network and the Anchor Node, which are not under the control of the UPMT solution. These local NATs (that can also be associated to firewall functionality) may limit the "roaming" connectivity that can be offered on a given local access network. This is exactly what happens today when getting Internet access from a public ISP or in a corporate LAN: users are generally not able to run all their applications, and can even be prevented at all from accessing the Internet. The UPMT mechanism will need to "test" the different access networks and the related "local NATs"/firewalls and will

exploit connectivity only from the suitable access networks.

A. *Application level vs. network level mobility*

The proposed mobility management solution (UPMT) addresses the outlined set of requirements working at the application level. It is not possible here to discuss the relative merits of application level solutions versus the other types of solutions (layer 2, network layer, transport layer, see [1]). We will only provide some comments in comparison with network level solutions that have received a lot of attention and work within IETF in the last years, i.e. Mobile IPv4 and Mobile IPv6. In the authors' opinion, the ease of deployment and the capability to provide a nearly ubiquitous coverage without changing the networking infrastructure make application level solutions preferable to Mobile IPv4 as of today. It is debatable what can happen in the medium/long term when IPv6 will come to play bringing its built-in mobility mechanisms. We can only leave the question open, while personally believing in the value of the application level mobility solutions. We also note that most of the UPMT mechanisms described hereafter have their correspondence in Mobile IPv6 (either in the "baseline" mobility support or in extensions that are under standardization). This will be mentioned where possible, but without providing a one-to-one comparison with Mobile IPv6.

III. TERMINAL AND APPLICATION ARCHITECTURE

Let us start by considering the "local" architecture within the Mobile Host, which defines how the applications are going to use the mobility services. As shown in Fig. 2, the terminal includes a Mobility Management Client (MMC) which can be further decomposed into a set of entities.

The MM Overall Control Entity (MMOCE) sets the policies and drive the decisions. The MMOCE also offers a Graphical User Interface to the user. Most decisions will be handled automatically, some times the user will be contacted to react to particular events or to gain permissions to perform some operations.

The MM Measurement Entity (MM-ME) gathers measurements over the different interfaces at various level and detects triggers like "interface up" and "interface down" events.

The MM Execution Entity (MM-EE) provides transport services to applications and "control" and "management" services to MMOCE. The MM Execution Entity is also in charge to handle the signaling towards other remote entities (not shown in the figure) as needed to obtain connectivity and perform the handovers.

The mobility services offered by the MMC can be exploited by the applications through two different interfaces: the UPMT socket interface and the MMC demon adapter.

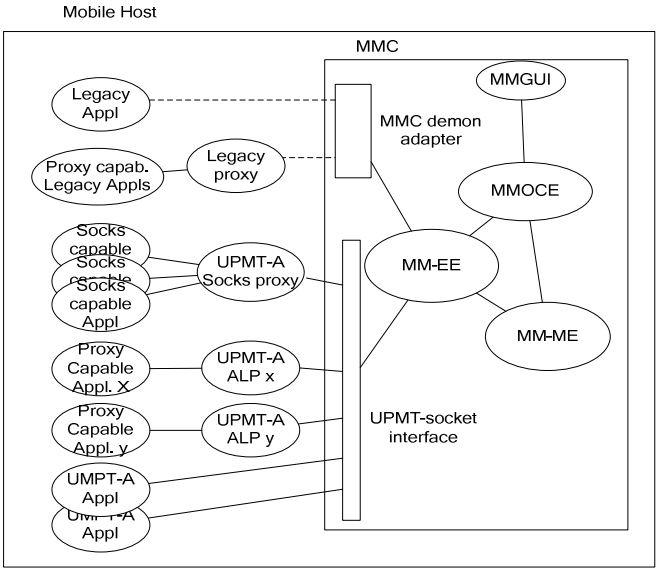

Fig. 2. Terminal architecture: Applications and Mobility Management Client

The UPMT socket interface extends the traditional socket interface by adding the means to explicitly control the mobility. It fully supports the traditional socket interface, without changing the signature of the methods or of the functions implementing the interface. Existing application could be modified at the source code level to use the new interface, becoming UPMT aware application. As we know that this is a difficult approach we considered a set of mechanism to reuse the legacy applications without any change at the source code level. A first possibility, for all the applications that support an application level proxy (ALP) is to develop an UPMT aware ALP (typically by modifying an existing ALP) that can run on the terminal. The UPMT aware ALP will connect to the MM-EE in the MMC and will be able to use the mobility services. A second, similar, chance is to develop a local UPMT aware Socks proxy which can provide services to all the Socks capable applications (several applications still support the old "Socks" interface).

The MMC demon adapter is not really an interface, as it is meant to support legacy applications that cannot change the way they interact with the outside world. The idea here is to provide a virtual network interface to the applications, have the applications open their socket on this virtual network interface and then handle the IP packets through the MMC. This is exactly the approach taken by the TUN/TAP drivers [7] which are often used to implement Virtual Private Networks. For the legacy application, it is a task of the MMOCE to select which applications should be handled by the MMC and which handover policy will be applied, and this is configurable by the user through the MMGUI. A straightforward approach is obviously to set the virtual network interface as the default gateway of the terminal, so that all application data will be sent via the MMC through the tunnels. Using this approach we can also use a legacy proxy for applications and have its flows intercepted by the MMC demon adapter. This approach obviously simplifies the work

of the MMOCE, at the price that all applications supported by the proxy would be handled in the same way at the mobility management level, not fulfilling requirement 3).

IV. NETWORKING ARCHITECTURE

The proposed architecture is depicted in Fig. 3. As introduced in section II, the MH is equipped with two (or more) interfaces connected through different access networks. Each interface has an IP address which may be either public or private. The MH will establish one tunnel toward the AN for each interface that it may want to use for sending and receiving data. The data to any Correspondent Host (CH) is encapsulated and sent to the AN on one of the tunnels, likewise the data coming from any CH is sent by the AN on one of the tunnels. In the solution discussed in this paper, the encapsulation performed at the MH and the AN is UDP/IP encapsulation, but different tunneling mechanisms can be used. The idea of having different tunnels potentially active at the same time between the MH and the Anchor Node is similar to the "Multiple Care Of Address" approach for IPv6 [6].

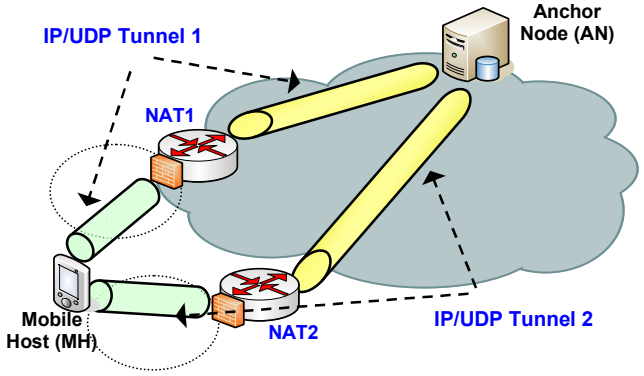

Fig. 3. High level network architecture

The internal architecture of the MH is depicted in Fig. 4 The Tunnel Forwarding Agent (TFA) will handle all the packets sent by the applications running on top of MMC. The interface between applications and the TFA can be either a virtual IP interface in case of legacy applications or the UPMT socket API in case of UPMT aware applications or UPMT aware local proxies.

The TFA forwards the data packets on the proper tunnel according to the Per-Application Forwarding Table (PAFT). This forwarding is based on the 5-uple: protocol, source and destination IP addresses and TCP/UDP ports. In practice the PAFT maps a socket into a tunnel *(socket -> TID)*.

The building of the PAFT is a critical aspect: it is easy for UPMT aware applications that will explicitly open the sockets through the UPMT socket API, while we need to introduce a socket identification and classification mechanism for legacy applications, that may associates packets to the process IDs of the applications.

The Tunnel Manager performs all the operations to establish, update, refresh and destroy the UDP tunnels between each active interface of the MH and the AN.

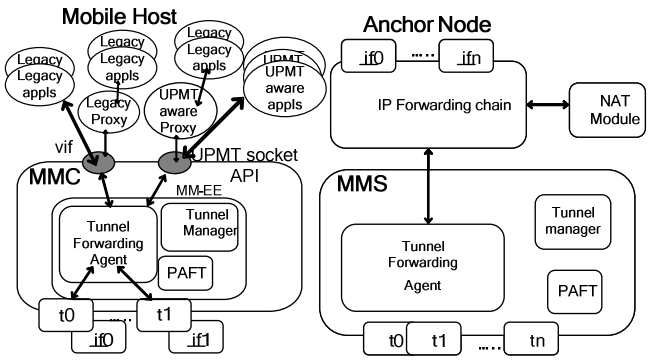

Fig. 4 Networking architecture of Mobile Host and Anchor Node

In the proposed architecture the AN (see Fig. 4), has a threefold role:

1. the AN acts as a **tunnel broker** for the MH. During the association procedure, a "Virtual IP Address" (VIPA) is assigned to the MH. This VIPA identifies the MH at the AN regardless of the particular tunnel through which the traffic is received. Each tunnel is univocally identified in the AN by the tunnel ID couple defined as *TID = (IP source address, UDP source port)*. We call it source address and source port because they are the address and port of the incoming UDP packets (i.e. received by the AN) that "opens" the tunnel. The AN will send back packets on this tunnel with IP destination address = received IP source address and UDP destination port = received source UDP port, in order to exploit the "symmetric" NAT traversal approach. Note that when the tunnel crosses a NAT, the IP source address and UDP source port of the packet received by the AN will be the NATed ones. In order to keep the pin-hole open in the NAT, a keep-alive mechanism handled by the client will be used, similar to the one defined in RFC 3948 [8].
2. the AN acts as the **anchor point** for the MH. All the tunneled packets from MH to CH are decapsulated and forwarded by the AN. The associations between application data flows and the tunnel from which they are received are kept at the AN in the per-Application Forwarding Table (PAFT). The PAFT is used by the AN to route packets coming from CH and destined to MH and its structure in the MMS is identical to the MMC, as it maps sockets into TIDs.
3. the AN performs **standard NAT** operations on all packets tunneled by the MH. In particular, after the packet is decapsulated, the virtual address of the MH, i.e. the IP source address of the packets sent by the MH, is replaced with a public addresses of the Anchor NAT.

V. PROCEDURES AND INFORMATION FLOWS

In this section, we first identify and describe at a logical level the four procedures and the associated information flows between the Mobile Host and the Anchor Node. An example sequence diagram is reported in Fig. 5, while Table II shows the set of messages and their parameters. In section VII we

present the mapping into protocols realized in our test-bed.

Association procedure - In order to associate with the MMS, the MMC sends an *Association request* message. The purpose of this message is initially to receive a virtual IP address (VIPA) in the *Association response* message from the MMS. The association is a soft state and it has to be refreshed with periodic Association request messages. The refresh Association request messages include the already assigned VIPA. The Association request messages can be subject to authentication by the MMS, in this case the procedure could change from a two-way handshaking to a four-way handshaking as the MMS may need to send some challenge to the MMC.

Tunnel establishment - As for the tunnel establishment, we considered two solutions. It is possible to have an "opportunistic" approach where tunnels are simply created by sending the first packet on the tunnel. The second option is to use some form of explicit signaling at the establishment of the tunnel, with a *Tunnel request* message. Explicit signaling can be used to support some form of authentication (e.g. to accept tunnels only from authorized MMC). Using explicit signaling it is possible to agree on an identifier for the tunnel, shared between the MMC and the MMS. This type of signaling could be realized in several different ways. One option is to piggyback information on the data packets, for example by reserving a signaling field at the beginning of the UDP payload, just before the tunneled IP packet. Another option is to have a similar signaling field to distinguish between tunnel packets carrying a tunneled IP packet and signaling packets. A third option is just to use a tunneled IP packed with destination IP corresponding to the MMS and destination UDP port directed to a process in the MMS that handles this signaling.

Handover Request - An *Handover request* message is used to switch a set of flows from one interface to another. The handover request message includes a list of flows and for each flow the target tunnel ID on which the flows needs to be handed off. If the request message is transmitted over a tunnel the target tunnel ID can be omitted for a flow and it means that the flow will be switched on the tunnel on which the Handover request message is received. This is the only option if there is no signaling to propagate the tunnel ID back to the MMC. An handover request message can be sent while the previous tunnel of the handed over flows is still active, thus supporting the so called "make before break" handover and minimizing the impairments. The Handover request can also be sent as a result of a sudden break of the connection supporting the previous tunnel. In this case, if the new tunnel is not yet active, the setup of the new tunnel and the handover request should ideally be performed at the same time in order to minimize the impairments.

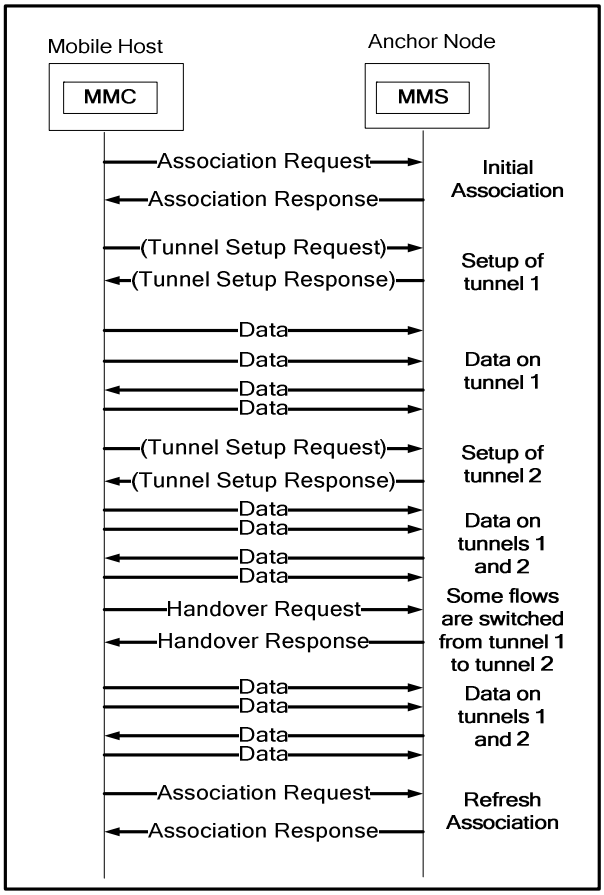

Fig. 5. A sample temporal diagram showing a set of information flows.

TABLE II
MESSAGES AND THEIR PARAMETERS

Association request (VIPA) Note that VIPA is missing in the initial association
Association response (VIPA) This message is sent using the same UDP socket of the tunnel
Tunnel setup request ()
Tunnel setup response (tunnel ID) This message is sent using the same UDP socket of the tunnel
Handover request ({[flow ID, tunnel ID]}) This message is sent on the same UDP socket of the tunnel. Therefore the tunnel ID can be omitted and (if omitted) implicitly derived from the received UDP port number/IP address. Note that a set of couples [flow ID, tunnel ID] can be transported in one single handover request message

Data transfer - The *DATA* messages correspond to the tunnel packets that carry the tunneled IP packet (which can be TCP or UDP or even a different protocol if the MMC NAT is able to handle them). The tunneled IP packet extracted from the DATA message automatically triggers the association of the corresponding flow in the PAFT (flow/tunnel association table). This process resembles the "learning" process of a NAT box. If needed, DATA messages could be authenticated and/or encrypted using various techniques.

Keep alive - Periodic *Keep Alive* messages are needed for each setup tunnel, in order to keep the NAT pin-holes open. As a rough estimate, the Keep Alive period can be in the order of tens of second. As we have a number of tunnels equivalent to the number of interfaces, this will not be a concern. The solutions to support the keep alive procedure can be borrowed from the ones discussed above for the Tunnel Establishment.

Actually using a combined solution for keep alive and tunnels establishment could result in higher efficiency (or at list in a simpler architecture).

VI. Example of Tunneling/NAT Operations

In this section we provide a detailed example of tunneling and NAT operations taking into account the scenario depicted in Fig. 6. The MH is equipped with two interfaces IF0 and IF1 connected to two different access networks.

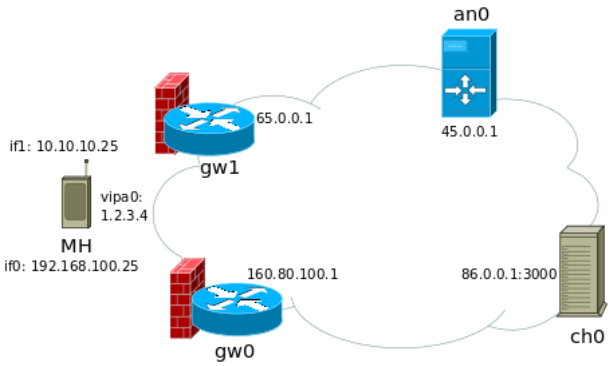

Fig. 6. Example scenario with IP addressing

Both interfaces have private IP addresses, respectively if0 = 192.168.100.25 and if1 = 10.10.10.25. Both interfaces are connected through a default gateway/NAT with public IP addresses, respectively gw0 = 160.80.100.1 gw1 = 65.0.0.1. The AN has public IP address 45.0.0.1 and the CH is a legacy UDP server with IP address ch0 86.0.0.1. and listening port 3000. The Virtual IP Address assigned by the AN to the MH in the association procedure is vipa0 = 1.2.3.4. The MH has setup two tunnels (referred to as t0 and t1) respectively on the interfaces if0 and if1. The tunnels IDs are tid0 and tid1. The source ports respectively for tid0 and tid1: for the MN 1500 and 1600; for the AN 33000 and 34000. The destination port for both tunnels is 1800.

Fig. 7 shows a generic exchange of two UDP packets between MH and CH. Assume that a legacy application running on the MH sends a UDP packet PCK1 with source port 2000 and destination port 3000, IP source address *vipa0* and IP destination address *ch0* 86.0.0.1, through the virtual interface *vif0*. This packet is handled by the local Tunnel Forwarding Agent and, according to the local PAFT is forwarded to the AN on tunnel *t0*. The outer IP/UDP header is formed according to the parameters of tunnel t0.

The AN, upon receipt of PCK1, updates the PAFT by inserting the following forwarding rule:

1.2.3.4 , 2000; 3000 (UDP), tid0

PCK1 is then processed by the NAT module as follows: (i) the virtual IP source address 1.2.3.4 is replaced with the public address *an0* 45.0.0.1; (ii) the UDP source port 2000 is replaced with port 40000 chosen by the NAT; (iii) the mapping is stored in the NAT table. Note that all operations performed by the NAT are standard symmetric address and port translations, and no NAT extensions are required. PCK1 is then forwarded to CH

In response to PCK1, CH sends a UDP packet PCK2 with destination IP address *an0* and destination UDP port 40000. Upon receipt of PCK2, the AN restores the NAT mapping by replacing: (i) IP destination address *an0* with the virtual address 1.2.3.4; (ii) UDP destination port 40000 with 2000.

PCK2 is finally intercepted by the AN Tunnel Forwarding Agent, encapsulated and forwarded back to MH through the proper tunnel t0 according to the specific PAFT entry.

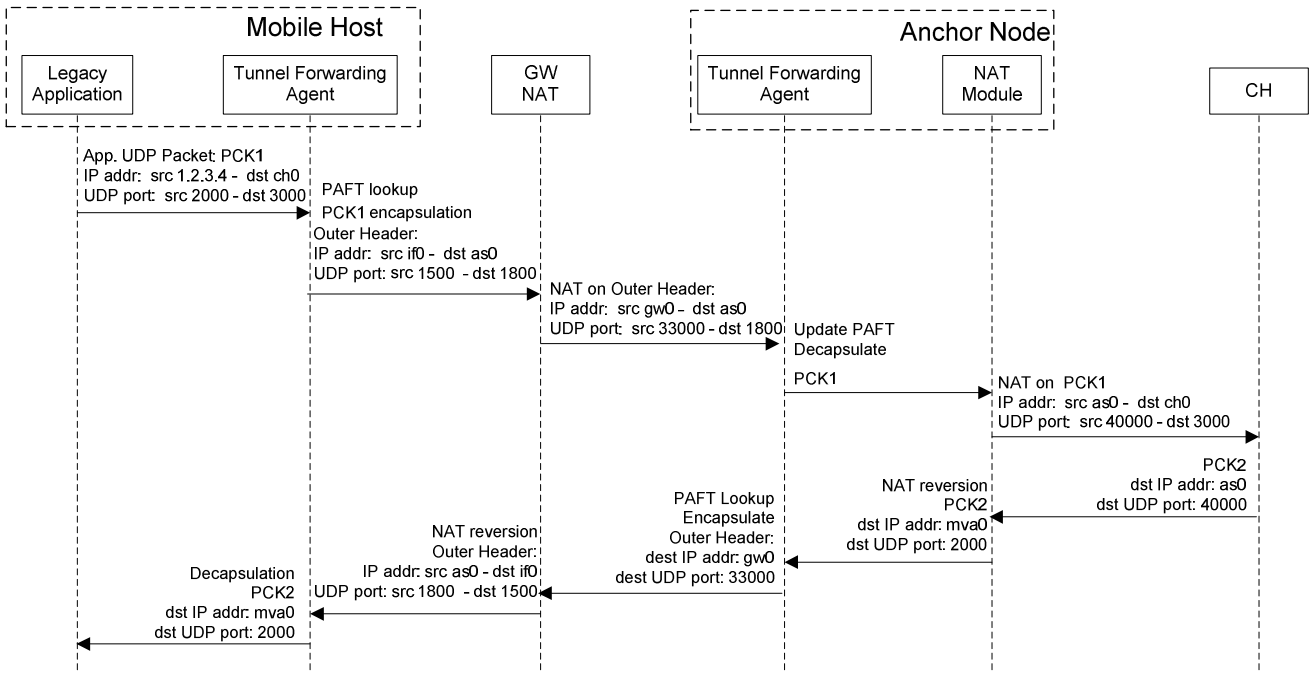

Fig. 7. Basic operations for data exchange

VII. Mapping of the Information Flows into Protocols

In this section we provide the proposed mapping of the information flows discussed in section V into SIP protocol messages. In principle, different protocols can be used to support the UPMT procedures, we used the SIP protocol because its features match very well the UPMT needs and because we could base our implementation on the evolution of the MMUSE testbed.

The SIP registration procedure offers authentication, registration refresh and retransmissions due to packet losses, therefore we can build upon the SIP REGISTER method and its associated mechanisms to implement the UPMT Association procedure. We have used the same approach in our MMUSE solution, where we have extended the SIP REGISTER method to support the needs of terminal mobility. We fully reuse the MMUSE mechanism, which allows to keep a bidirectional SIP signaling channel between the MMC and the MMS, following the roaming of the terminal across different interfaces. In order to support UPMT we added a new header to carry the Virtual IP Address, called "VIpAddr". The addition of this new header is not a problem, as the SIP messages with the new header needs will only be exchanged between our MMC and our MMS. Fig. 8 reports the same scenario considered in Fig. 5 with the protocol messages names. Few SIP messages are (partially) shown in Fig. 9 (see [3] and [9] for further details). Note that the messages related to this procedure are sent *outside* of the tunnels.

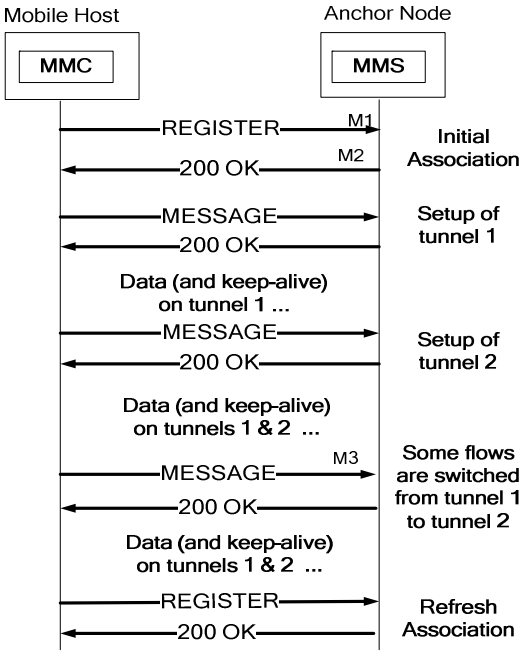

Fig. 8. Protocol messages

For the Tunnel Establishment and Handover Request procedures, we chose to use the SIP MESSAGE method. This method is used to exchange arbitrary information between two SIP entities, without setting up a session and is able to deal with retransmissions due to packet losses. In this case, we preferred not to define new SIP headers. Rather, the needed information is transported within the body of the MESSAGE, as plain text using JSON (a lightweight data-interchange format, see [10]). Both the Tunnel Establishment and the Handover Request MESSAGEs are sent inside the tunnels (as the corresponding 200 OK messages). The two procedures can be combined into a single one when it is needed to open a new tunnel and handover a set of the flows on it (e.g. after the sudden failure of an active interface). An example of the Handover Request message is reported in Fig. 9. The specification of the JSON syntax of the messages body cannot be reported here for space reasons and can be found at [11], together with a larger set of messages examples.

M1: *Association Request* Register – MMC to MMS

```
REGISTER sip:45.0.0.1 SIP/2.0
Via: SIP/2.0/UDP 192.168.100.25; branch=z9h; TID=Terminal_ID; rport;
To: <Terminal_ID>
From: <Terminal_ID>;tag=dba
Contact: <sip:Terminal_ID>
```

M2: Association Response 200 OK – MMS to MMC

```
SIP/2.0 200 OK
Via: SIP/2.0/UDP  192.168.100.25; TID=Terminal_ID;received=160.80.100.1
To:< Terminal_ID >
From:< Terminal_ID>; tag=dba
VIpAddr:1.2.3.4
```

M3: Handover Request – MMC to MMS

```
MESSAGE sip:45.0.0.1 SIP/2.0
Via: SIP/2.0/UDP 192.168.100.25; branch=z9h; TID=Terminal_ID; rport;
To: <sip:45.0.0.1>
From: <Terminal_ID>;tag=dba
Contact: <sip:Terminal_ID>
Content-Length: xx
Content-Type: application/text

{"mType":"HandoverRequest",
"match":{"syntax":"IPtables", "match":"-p tcp –destination 160.80.81.1 –
destinaton-port 80"},
"tunnelID":["13673865","any"]}
```

Fig. 9. Example of few selected messages

VIII. Performance Considerations

The performance analysis of the solutions for terminal mobility is a challenging task, as it includes: 1) performance of signaling procedures and of data transfer (both affected by network impairments like packet delay and loss); 2) evaluation of the load imposed to network entities (e.g. to the different types of "home" and "visited" servers foreseen); 3) issues in the operating systems of the Mobile Hosts. Application level solutions like UPMT are often challenged with respect to performances as compared to network level solutions. It is not possible here to provide a substantiated answer to these challenges. We can only provide and some hints for further readings and some opinions. As for the evaluation of the handover delays due to the signaling procedures, a comparison between application level mobility and network level mobility can be found in [13], showing the good performance of application level mobility. Looking at the results in [12], we observe that from the point of view of user perception the impairment due to the handover is dominated by the difference in the RTT on the two different networks, so that the type of mobility solution is not critical.

The most critical aspect of the UPMT solution as it is

proposed in this paper (and of all application level mobility solutions that rely on overlay servers / anchor nodes) is for sure the need of distributing the load on the Anchor Nodes. The opinion of the authors is that application level solutions can be engineered to perform (at least) as good as network level solutions, mainly depending on the capability to geographically distribute the Anchor Nodes or to have good network connections to the Anchor Nodes from the Access Networks.

The second important performance aspect to be consider for the UPMT solution, based on tunnels, is the tunnelling overhead. The UPMT solution can be adapted to different types of tunnels, the default choice that we have considered is to use UDP tunnels. The related overhead is 28 bytes per packet. In order to be fair, we observe that the most recent networking level solutions (e.g. Mobile IPv6) do not imply such tunnelling overhead.

IX. Status of Implementation

We are progressing in an open source implementation of UPMT for Linux OS. We are currently focused on laptop and netbooks, we plan to extend the work for the Android platform sooner than later. Updates of the status of the work and more information on the implementation are available at [11].

SIP Signaling - The signaling part has been largely derived from the MMUSE solution and it is completed at the time of writing, for both the MMC and the MMS. It is written in Java based on the open source SIP stack mjsip.

Virtual interface and Tunnel Forwarding Agent - We considered two possible solutions: (i) a kernel space approach similar to the IP/IP tunneling kernel module; (ii) a user space approach based on existing open source tools as the "Dummy interface" module and the Netfilter module. The first approach requires a new kernel module providing IP/UDP tunneling, exportation of virtual tunnel interface to user space and "per-application" forwarding based on PAFT. The second approach does not require modification to the kernel, therefore it can be installed as an application on top on any Linux box. At the time of writing we have completed the implementation of the user space approach (details and source code in [11]) and we will evaluate the need of the kernel space implementation based on performance considerations. An important implementation aspect (which applies to both solutions (i) and (ii)) is related to the need for multiple routing tables in the Mobile Host. A virtual routing table with one default entry with the virtual interface as output interface is used in order to force legacy applications to use the virtual IP address as IP source address. On the other hand, all tunneled packets sent by the forwarding agent needs the real routing table for the physical interfaces. The chosen solution uses Netfilter to mark all packets sent by a given Process ID or command and then uses policy routing with rules based on the mark assigned by Netfilter.

Classification of legacy applications – This currently our main implementation challenge. Once again, we are considering both kernel-based solutions that require the modification or adding of new kernel modules and user space solutions. A possible user space implementation is to actively monitor the list of processes and keep track of all the active sockets associated to them, using tools like *ps* and *netstat*. In this way, application flows classification can be done by the 5-tuples identifying its sockets.

X. UPMT extensions

It is worth listing the main open issues that we are considering in order to extend UPMT, facing its current weaknesses. (i) migration from a centralized architecture based on a single AN to an architecture based on distributed Anchor Nodes. Work in this direction is included in [9] for the MMUSE approach, but can be reconsidered in our context. (ii) more efficient support of Mobile Hosts with public IP address, avoiding the transit through the AN when it is not needed. Mechanisms borrowed by ICE could help. (iii) investigation and possible exploitation of alternative tunnel mechanisms

Acknowledgements

The authors would like to thank Daniele Dedda and Andrea Capitani for their work on the implementation of UPMT.

References

[1] Deguang Le, Xiaoming Fu, Dieter Hogrere, "A Review of Mobility Support Paradigms for the Internet", IEEE Communications surveys, 1st quarter 2006, Volume 8, No. 1

[2] Moonjeong Chang, Meejeong Lee, Hyunjeong Lee "Per-Application Mobility Management with Cross-Layer Based Performance Enhancement", IEEE WCNC 2008.

[3] S. Salsano, C. Mingardi, S. Niccolini, A. Polidoro, L. Veltri "SIP-based Mobility Management in Next Generation Networks", IEEE Wireless Communication, Vol. 15, Issue 2, April 2008

[4] MMUSE web site, http://netgroup.uniroma2.it/MMUSE

[5] Wikipedia, "Network address translation", http://en.wikipedia.org/wiki/Network_address_translation

[6] R. Wakikawa, V. Devarapalli, G. Tsirtsis, T. Ernst, K. Nagami, "Multiple Care-of Addresses Registration", draft-ietf-monami6-multiplecoa-14.txt, May 27, 2009

[7] TUN/TAP driver homepage, http://vtun.sourceforge.net/tun/index.html

[8] A. Huttunen, B. Swander et al., "UDP Encapsulation of IPsec ESP Packets", RFC3948, January 2005

[9] A. Polidoro "Mobility Management in Next Generation Networks", PhD Thesis, 2009, http://netgroup.uniroma2.it/MMUSE

[10] "Introducing JSON", http://www.json.org/

[11] UPMT web site, http://netgroup.uniroma2.it/UPMT

[12] S. Salsano , L. Veltri, A. Polidoro, A. Ordine, "Architecture and testbed implementation of vertical handovers based on SIP Session Border Controllers", Wireless Personal Communication, November 2007

[13] A. Polidoro, S. Niccolini, S. Salsano, "Performance Evaluation of vertical handover mechanisms in IP networks", IEEE WCNC 2008, Las Vegas, USA, 31 March – 3 April, 2008.