Achieving Scalability in the UPMT Mobility Management Solution

Marco BONOLA[1], Stefano SALSANO[1]
[1]*University of Rome "Tor Vergata", Via del Politecnico, 1, 00156, Roma, Italy*
Tel: +390672597770, Fax: + 390672597435,
Email: {stefano.salsano,marco.bonola}@uniroma2.it

Abstract: UPMT (Universal Per-application Mobility management using Tunnels) has been proposed as an application level solution for mobility management. Main features of UPMT are the "per-application" handover and the full compatibility with legacy applications, legacy hosts and existing networking infrastructure. A critical issue of the existing proposal is the scalability, as the specification foresee a single Anchor Node (AN) handling the mobility management signalling procedure and acting as "tunnel server" for relaying information towards the Mobile Hosts. Therefore in this paper we denote the existing solution as "UPMT-Centralized AN Mode" (CAM) ad propose to enhance it by defining the UPMT-Distributed AN Mode (DAM). The UPMT-DAM: 1) considers multiple Anchor Nodes; 2) it may allow a Mobile Host to directly communicate to a "Fixed" Correspondent Host (CH) or to another Mobile Host bypassing the Access Nodes and handling the mobility procedures with the CH or with the other MH.

Keywords: Application Level Mobility Management, SIP, Vertical Handover

1. Introduction

Current notebooks, netbooks, and handheld devices have more than one networking interface (cellular, wifi, Bluetooth, fixed ethernet). In principle, these devices could select among several different network access providers at a given time: wifi hot spot providers, cellular operators, corporate or campus networks (the actual chances to connect to these network is obviously limited by policy issues as you need to have access rights to one given access network). Handling seamless mobility across different access networks could be of great value for users. The vision is that the user could have its applications running independently of the current access network with no service disruption when changing from one access network to the other. A further step in this vision is that different application could be independently transported over one access network of the other depending on the combination of application requirements and factors like cost and quality of service of the different access network. The capability of roaming across among different network access technologies in an "intelligent" and automatic way has been called "Always Best Connected" (ABC) service in [1].

From the business model point of view, this "ABC service" could be provided by a single provider that controls different access networks with different technologies or by an "aggregator" entity which is able to combine the service from different providers. In the context of the EU research project "PERIMETER – User-Centric Paradigm for Seamless Mobility in Future Internet" the second approach has been followed. In particular a user-centric vision been considered, in which the user herself is controlling and coordinating the

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

roaming among different providers and access technologies. Anyway, we note that the mobility management solution proposed in this paper is independent from the underlying business model.

In order to fulfil the requirements of the "ABC services" there is a great interest in Application Level solutions for mobility management, as they do not require support from the network layer and from the underlying layers [2][3][4][5]. Therefore, they can be implemented using existing networking infrastructure and are especially suited for mobility management in the presence of heterogeneous and multi-operator access network technologies, in contrast with network-layer solutions like [6][7].

UPMT (Universal Per-Application Mobility management using Tunnels), which has been proposed in [8], is based on IPinUDP tunnels for transport and on SIP protocol for signalling. The UPMT solution was designed to fulfil a set of requirements which are listed and analysed in [8]. In short, UPMT offers per-application handover, supports legacy applications on the mobile devices and is fully interoperable with legacy hosts and with existing networking infrastructure, including private IP numbering and NATs (Network Address Translation). UPMT relies on the presence of an "Anchor Node" (AN) for packet relay between a UPMT aware Mobile Host (MH) and a legacy Correspondent Host (CH). The UPMT solution proposed in [8] considers the case in which only one "centralized" Anchor Node is present, we will refer to this solution as "*UPMT Centralized AN Mode"* (*UPMT-CAM*). We believe that this is not scalable and propose to extend UPMT considering a "*Distributed AN Mode*" (DAM). *UPMT-DAM* should: 1) provide the means for a MH to use different Anchor Nodes as needed to distribute load and to optimize performance (e.g. by choosing where possible an AN in the path between the Mobile Host and the Correspondent Host); 2) allow a "fixed" Correspondent Host to play the role of Anchor Node; 3) allow direct communication between Mobile Hosts bypassing Anchor Nodes whenever possible (the AN would be used as rendezvous point for signalling and to assist in the setup of end-to-end data tunnels, as in [10]). The challenge is that UPMT-DAM should still offer the same features of UPMT-CAM as regards per-application handovers, support of legacy applications, interoperability with legacy hosts and existing networking infrastructure.

In this paper, section 2 provides an introduction to UPMT, section 3 and 4 discuss some issues and propose the extensions for UPMT-DAM, section 5 focuses on the procedures for the case in which a Mobile Host is connected to a "fixed" Correspondent Host, section 6 describes the implementation and a simple testbed experiment, limited to UPMT-CAM.

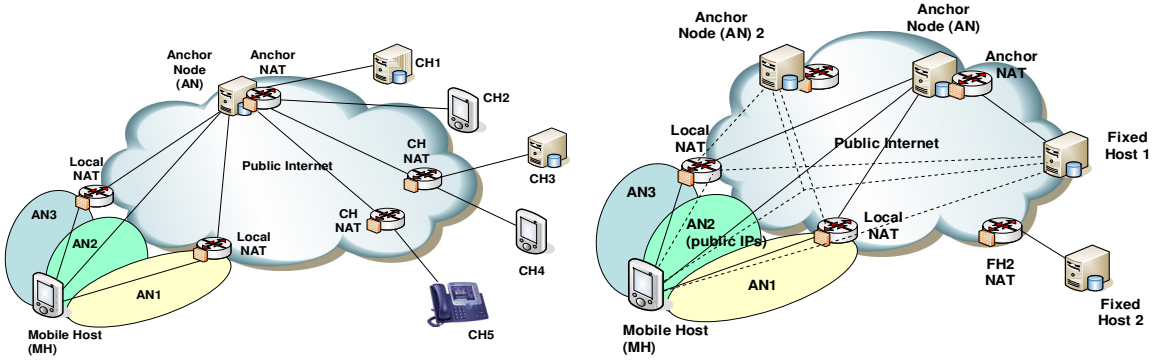
2. UPMT-CAM and UPMT-DAM basics

As shown in Figure 1 (left), a Mobile Host can be connected using several different access networks, most of which will provide private IP addresses and use NATs to interwork with the Internet. In the UPMT-CAM solution, the MH connects to an Anchor Node using IP in UDP tunnels, one tunnel per each access network. The MH uses a "Virtual IP Address" VIpA that is assigned to it by the Anchor Node. The Anchor Node provides a "second level" NAT to the Mobile Host, allowing the host to access to the Internet using a public IP address provided by the Anchor Node. The forwarding and tunnelling of packets in the Mobile Host and in the Anchor Node is regulated by a table called "*Per Application Forwarding Table*" (*PAFT*), which associates an application level "socket" to the tunnel used to send packets from the MH to the AN and from AN to MH. The procedures to establish the tunnels, to manage the PAFTs, and to drive the handover of application flows across the tunnels for UPMT-CAM are described in [8], [9]. We note that problems may arise as the different access networks are in general not coordinated and could use the same private IP address space when assigning the IP addresses to the network interfaces of the MH. This is not a specific problem of UPMT, as in any case the IP networking stack in the

device would not be able to correctly handle a situation in which the same IP subnet address is used on two different interfaces. We are investigating solutions to this issue based on the idea of offering a set of different IP addresses belonging to different IP subnets to the MH, so that the MH can properly choose the addresses to avoid overlapping.

The UPMT-DAM approach proposed in this paper considers three scenarios for the "distribution" the functionality that was centralized in UPMT-CAM: 1) Multi-ANs scenario (multiAN) in which MHs can use different ANs for communication with legacy CHs, the AN may be chosen with the goal to optimize the performances; 2) MH-to-FH end-to-end scenario (mh2fh) in which UPMT aware mobile hosts (MH) and fixed hosts (FH) communicate directly with each others without necessarily relying on ANs for packet forwarding; 3) MH-to-MH end-to-end scenario (mh2mh) in which MHs communicate directly with each others without necessarily relying on ANs for packet forwarding and use different ANs. The multiAN and the mh2fh scenarios are depicted in Figure 1 (right), which shows three tunnels setup between the MH and the AN with solid lines. The MH could switch the tunnels to a second Anchor Node or directly to a Fixed Correspondent Host (tunnels drawn with dashed lines). Mh2mh scenario is not drawn for space reasons.

Note that the discussion on VIpA assignment (section 3) and of the new "Per-Application Forwarding Table" (section 4) are common to the threes scenarios, while we provide the specification of procedures only for the mh2fh scenario (section 5). Work on the definition of the procedures for multiAN and for mh2mh scenarios is still ongoing. A very interesting issue in the case of the multiAN scenario is the problem of anchor point placement and selection. A MH could have a default Anchor Node and then it should be able to select a different AN with the goal of optimize the performance, depending on which are its active network interface and current traffic pattern. These aspects are out of the scope of this paper and are subject of our ongoing research.



UPMT-CAM UPMT-DAM: multiAN & mh2fh

Figure 1: Scenarios

3. The VIpA Assignment Issue

In the UPMT-CAM solution all traffic for a given Mobile Host is relayed by a single Anchor Node. The Anchor Node assigns the MH a VIpA during the Association Request phase. For the AN, every MH in the UPMT overlay network is univocally identified by a VIpA and thus every inner IP packet carries all the information required to be correctly forwarded without inspecting the external IP/UDP header (the tunnelling approach is shown in Figure 2).

In UPMT-DAM, we introduce a set of Anchor Nodes and allow direct communication between MH and UPTM aware Fixed Hosts and among MHs. If we want to keep the hypothesis of a bi-univocal association between a MH and a VIpA, we need to assign it in a coordinated way among the tunnel servers (multi AN, FHs or MHs). The VIpA assignment procedures need to be coordinated and a DHCP-like protocol is required. Moreover, since

VIpAs are merely IPv4 addresses, the UPMT architecture will eventually face the same addresses exhaustion limitation of IPv4.

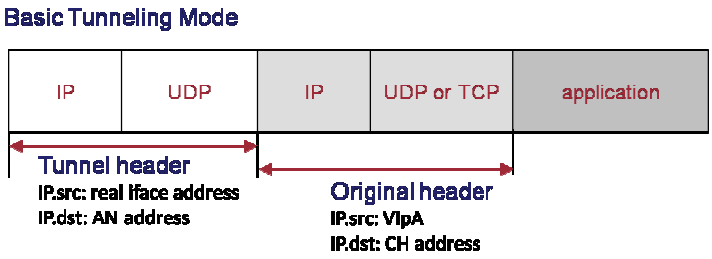


Figure 2: IP in UDP tunneling

Therefore we propose to use a new approach, in which the VIpA is uniquely associated to a MH in each tunnel server that assigns it ("Per-Association Unique VIpA"). Therefore different tunnel servers may assign the same VIpA to different MHs and the MH will in general receive different VIpA by the different tunnel servers (either an AN, a FH or another MH) to which it will connect. If MH is associated with *N* UPMT peers, the MH will be identified by *N* VIpAs, each of which will be unique only at the Tunnel Server that has provided it.

Since having *N* virtual interfaces on the MH would make the whole system not scalable and hard to transparently integrate legacy applications, the virtual interface will be configured with a fixed and locally unique IP address (*VIpA_fix*). Then we will have a NAT operation, *internal to the mobile host*, will be performed on IP.src for outgoing and incoming packets. For outgoing packets this NAT operation will change the IP.src with the VIpA, depending on the remote tunnel end-point to which the MH is communicating and on the given application flow. To realize this *internal* NAT of the *VipA_fix* address, a UPMT Tunnel Table (TT) is used to store all the bindings between each active tunnel and the VipA that has to be used as source IP address (for outgoing packets from that tunnel). The Tunnel Table can be either a different table or integrated in the PAFT.

4. Per-Application Forwarding Table extension

The PAFT structure described in [1] is strictly limited to a scenario in which only one "tunnel end point" (or "tunnel server") is available for the mobile host. We now want to support an end-to-end scenario with different tunnel end points, therefore the PAFT need to consider all possible remote peers for each application flow. The extended PAFT (see Table 1) consists in two fields: **app_flow** is the concatenation of the following fields: *protocol, ip src, ip dest, src port, dest port*; **local_tid** is a locally unique numeric identifiers for the tunnel bound to *app_flow*.

The PAFT is therefore linked through the key *local_tid* to a Tunnel Table (TT) which consists of the following 4 fields:

- local_tun : the couple *(out_iface, local_tun_idx)*, it identifies the interface, the IP source address and the index for tunnel bound to *app_flow*. The format of *local_tun_idx* depends on the specific tunnel mechanism. For example: (i) in case of IP/UDP tunnels *local_tun_idx* would be the UDP source port of the external UDP header; (ii) in case of IP/GRE tunnels *local_tun_idx* would be a GRE key field.

- remote_tun the couple (*tun_end_point, remote_tun_idx*). The first field is trivially the destination IP address of the tunnel. The second field is the remote tunnel index (again, UDP remote port or any other tunnel index depending on the tunnelling mechanism).

- vip_nat is the local VIpA assigned to the peer MN is communicating to through the tunnel identified by *local_tid.*

- un_tid is a numeric identifier that univocally identifies a tunnel between two endpoints. This field is used only with tunnels explicitly set up and allows signalling messages to be sent outside tunnels.

Let us consider an example of communication between a MH and an AN when the per-association unique VIpA assignment is performed. We assume a MH and two ANs as shown in Figure 1. The *VIpA_fix* 1.2.3.4 address is assigned to the virtual interface of the MH at bootstrap (or when UPMT is started). The MH performs the UPMT association and tunnel request procedures with AN1 (UDP tunnel remote port *p1*). Upon completion the MH receives a virtual address *VipA1* 5.0.0.1 from AN1 and sets up a tunnel identified by *tid1*. Later on MH associates to AN2 (UDP tunnel remote port *p2*), gets a virtual address *VipA2* 7.0.0.1 and sets up a second tunnel identified by *tid2*.

app_flow	**local_tid**
app_flow1	*Tid1*
app_flow2	*Tid2*

local_tid	**local_tun**	**remote_tun**	**vip_nat**	**un_tid**
tid1	eth0,2000	*AN1,p1*	5.0.0.1	1
tid2	eth0,3000	*AN2,p2*	7.0.0.1	2

Table 1: PAFT and TT

As for the first application flow, packets (with IP.src=1.2.3.4) are NATed so that IP.src is change to 5.0.0.1 and sent to AN1. Packets related to the second flow are instead NATed so that IP.src=7.0.0.1 and sent to AN2. After this "local" NAT, packets are forwarded through the proper tunnels, respectively *tid1* and *tid2*. When the encapsulated packets are received from the AN, upon de-tunnelling the inner packets received with the VIpA addresses as destination are "NATed" back so that IP.dst=1.2.3.4.

Since a MN is identified by a VIpA by each tunnel server it is associated with, we can continue to exploit two important features that have been proposed in UPMT-CAM. First we have the "implicit" handovers, which means that an application can drive the handover by simply sending packet on a new "target" tunnel. The PAFT in the anchor node will use auto-learning and there is no need of explicit tunnel setup). Likewise, in the Anchor Node the standard NAT procedure called "MASQUERADING" is feasible based on the VIpA.

5. Procedures for MH to FH communication in UPMT-DAM

Let us consider the case of a mobile client browsing a fixed web server. As shown in Figure 1 (right) there is the possibility to establish direct connections (dashed lines in the figure) between the MH and the fixed host, without using the Anchor Node as a relay (this is granted if the Fixed Host has a public IP address, like the Fixed Host 1 in Figure 1, if the fixed host is behind a NAT like the Fixed Host 2 it depends on the type of the NAT). The basic idea is that the AN is not used for permanent relay of the data, but it becomes a support node that helps in the establishment of the connections and in the NAT traversal and plays a relay role only when needed. This is exactly the same approach used by the ICE ([13]) NAT traversal solution in which the communicating entities try to establish a direct communication across NATs with the help of servers and falls back to a relayed communication when this is not possible. The Anchor Node will also be used as a rendezvous-point in some special cases, for example when both ends of the communication will move into a new access network at the same time (of course in the case of a MH to MH communication).

The details of the procedure that allows a MH to establish a connection with a UPMT server running a legacy application (e.g. a web server) are provided in [9]. The flow of messages is shown in Figure 3.

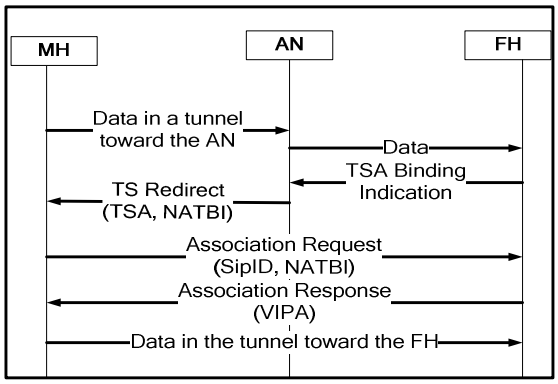


Figure 3: Message Flow

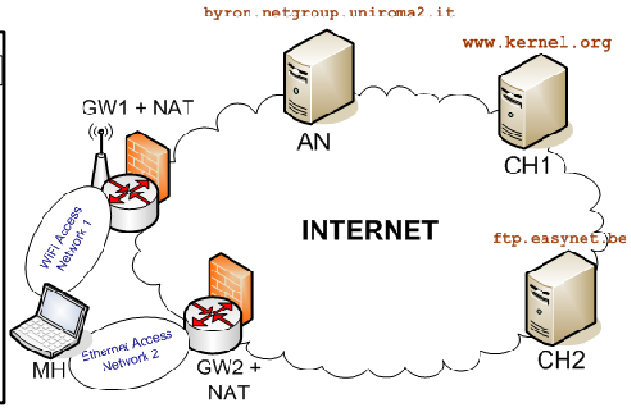


Figure 4: Testbed

6. UPMT implementation and testbed evalution

In this section, the open-source implementation of the UPMT solution for Linux OS is described along with a real demonstration. The current implementation only covers the UPTM-CAM. Further details about the implementation and the source code are available at [15]. The MH and AN implementations consist of the following two entities:

1. **Mobile Management Execution Entity - MMEE**: this entity is responsible for the encapsulation and forwarding of a given application flow within the proper tunnel.
2. **Mobile Management Control Entity - MMCE**: this entity is responsible for the UPMT signalling and is implemented as a user-space SIP User Agent extended to support all the procedures described in [8]. This application - largely derived from MMUSE [11] - is written in Java and based on the open source SIP stack mjsip.

MMEE development represented the most challenging part of our implementation work, it provides the following features:

1. MMEE is able to "put under UPMT control" a sub-set of application flows identified by the 5-tuple: (*protocol, IP.src, IP.dst, L4.srcport, L4.dstport).* Packets locally generated by applications under UPMT control are routed through a virtual interface that takes into account the decrease in the MTU due to the extra tunnel header. Packets from different application flows are handled **independently** from each others.
2. Packets from applications under UPMT control are intercepted within the IP stack and encapsulated within IP in UDP tunnels (Figure 2). Tunnelled packets are routed on the "real" interface bound to a specific tunnel and sent to the proper default gateway.
3. Packets from applications **not** under UPMT control are transparently routed according to the default routing table, which is only updated by the Operating System.
4. MMEE transparently handles **UPMT unaware** applications.
5. MMEE is controlled by another applications (i.e. MMCE).

MMEE has been implemented as a user-space daemon and relies on the following Linux Kernel 2.6.x features:

1. **Dummy module**: a dummy interface is used to implement the virtual interface. This kind of interfaces completely lacks the virtual methods to send and receive at Layer2. This is not a problem since the each packet routed through this interface is intercepted before reaching L2. The MTU of this interface is decreased of 28 bytes (IP header + UDP header) by using standard IP configuration tools.
2. **Raw Socket**: it is used to re-inject packets into the IP stack upon de-tunnelling.
3. **NETFILTER framework**: this kernel module and the relevant user-space tools are used to filter and intercept application data flows. Packets from application that are to be tunnelled are sent to the user-space UPMT daemon through NETLINK sockets

(NETFILTER NFQUEUE target), whereas packets from applications not under UPMT control are marked (MARK target) to be handled by the "default" routing mechanism. Moreover, MASQUERADE target is used at the AN and at the MH to translate/restore the ViPA.

4. **Policy Routing**: this kernel feature is used to create different routing tables and set the relevant forwarding look-up rules. In details, in addition to the default table, the following tables are used: (i) UPMT routing table with the virtual interface as default output interface (packets from application under UPMT control); (ii) *N* per-iface tables, where *N* is the number of real interfaces. This tables are used to route tunnelled packets through the real interface and to the proper default gateway. Packets from applications not under UPMT control are transparently supported by letting them match the forwarding rule pointing to the default routing table.
5. **Local Socket**: it is used as local control interface for other programs.

6.1 - Testbed evaluation

The testbed for our demonstration is depicted in Figure 4 and consists of the following elements: the Mobile Host **MH** is a laptop connected through a Ethernet interface (*iface1*) and a WiFi 802.11(*iface2*) interface. MH is running a legacy HTTP client (WGET) and a legacy FTP client (Firefox). The Anchor Node **AN** is a server at our university with public IP address 160.80.103.147. The two Correspondent Hosts (**CH1**, **CH2**) used in the demo are respectively (i) a legacy HTTP servers (www.kernel.org); (ii) a legacy FTP server (ftp.easynet.be). The evaluation is limited to the UPMT-CAM, according to the current status of the implementation, but no evaluation of UPMT had been provided in the original UPMT paper [8].

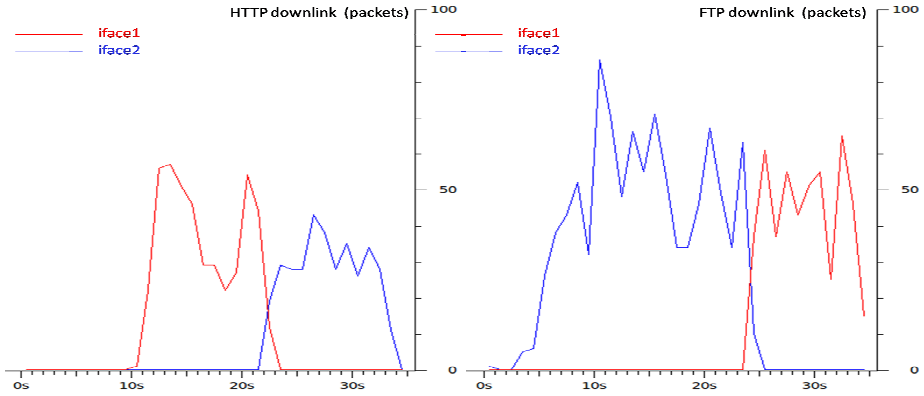


Figure 5: Measurements

The demo takes place as follows. At start-up, a set of IPTABLES rules are configured in order to route HTTP traffic through the tunnel bound to *iface1* and FTP traffic through the tunnel bound to *iface2*. Upon initial association with AN, MH starts a FTP (PASSIVE MODE) download from CH2 at time 2s and a HTTP download from CH1 at time 10s. After some time, the two application flows are swapped, using the **implicit handover** approach. In particular: (i) HTTP traffic is moved to *iface2* at time 21s; (ii) FTP traffic is moved to *iface1* at 23s. When the Anchor Node receives the packets on a different tunnel, it updates its PAFT and starts sending the downlink packets on the new tunnel. In addition, for the whole demo, a SSH session with AN is running totally unaware of the UPMT architecture according to the regular MH routing behaviour.

Figure 5 shows the downlink traffic (in packets) measured at the MH with a network analyzer. The red line represents the traffic captured for *iface1*, the blue line represents the traffic for *iface2*. The implicit handover of HTTP traffic (and similarly for FTP traffic)

takes place as soon as the AN receives the HTTP flow (*tcp, 1.2.3.4, 204.152.191.37, 4876, 80*), from a different tunnel. The PAFT is then updated and response packets are forwarded in the new tunnel. As shown in Figure 5 for both flow, the total of packets received on the "old" and "new" interface is not decrease during the handover procedure, this means that the handover procedure is performed in a seamless way, with no service interruption.

7. Conclusions and ongoing works

In this paper we have described the UPMT-DAM mobility management solution that addresses the scalability concerns of the original UPMT-CAM proposal. The challenge was to design a scalable application-level solution for mobility management that provides per-application handovers, support of legacy applications, interoperability with legacy hosts and full compatibility existing networking infrastructure. In addition we have provided some testbed results of the existing implementation of UPMT-CAM.

Our work on UPMT-DAM is ongoing in the following main directions. We are defining the detailed procedures and signalling messages for the multi-AN and for the mh2mh scenarios. In particular, the Mobile Host to Mobile host scenario is really challenging. As the support of NATs is one key feature of UPMT, we are considering a "NAT friendly" solution that allows two mobile hosts to establish the tunnels for mobility support even behind tunnels. Of course in some cases the use of an external relay will be unavoidable. A second direction is the security, we have defined a security approach based on IPsec in the internal packets carried in the tunnels and we are working on its implementation. The third main direction of work is the porting of current user-space PAFT implementation in the kernel space. This can boost performances and in particular it can improve the scalability of the Anchor Nodes.

References

[1] E. Gustafsson et al. "Always Best Connected", IEEE Wireless Communications, Feb 2003.
[2] J. Rosenberg et al., "SIP: Session Initiation Protocol, " RFC3261, Jun 2002.
[3] P. Vixie et al., "Dynamic Updates in the Domain Name System (DNS UPDATE)," RFC 2136, Apr. 1997.
[4] P. Eronen, "IKEv2 Mobility and Multihoming Protocol (MOBIKE)," Internet draft (work in progress), draft-ietf-mobikeprotocol-07, Dec 2005.
[5] Elin Wedlund, et al "Mobility Support using SIP", 1999
[6] C. Perkins, "Mobility Support in IPv6," IETF RFC3775, Jun 2004.
[7] H. Soliman et al., Hierarchical Mobile IPv6 Mobility Management (HMIPv6), IETF RFC4140, Aug 2005.
[8] M. Bonola et al. "UPMT: Universal Per-Application Mobility Management using Tunnels", IEEE GLOBECOM 2009
[9] S. Salsano et al. "The UPMT solution (Universal Per-application Mobility Management using Tunnels)", technical report, http://netgroup.uniroma2.it/TR/UPMT.pdf
[10] R. Moskowitz et al. "Host identity protocol (HIP) architecture", IETF RFC 4423, May 2006
[11] A. Polidoro "Mobility Management in Next Generation Networks", PhD Thesis, 2009, http://netgroup.uniroma2.it/MMUSE
[12] Netfilter.org projects homepage, http://www.netfilter.org/
[13] J. Rosenberg, "Interactive Connectivity Establishment", IETF Journal, Nov. 2006
[14] Linux Policy Routing documentations and software, http://www.policyrouting.org/
[15] UPMT home page, http://netgroup.uniroma2.it/UPMT