UPMT Per-Application Mobility Management Solution: a Demo for Linux and Android Terminals

Fabio Patriarca
Dip. Ingegneria Elettronica
Università di Roma "Tor Vergata"
Roma, Italy
fabio.patriarca.2@uniroma2.it

Stefano Salsano
Dip. Ingegneria Elettronica
Università di Roma "Tor Vergata"
Roma, Italy
stefano.salsano@uniroma2.it

Marco Bonola
CNIT - Research Unit
Università di Roma "Tor Vergata"
Roma, Italy
marco.bonola@uniroma2.it

Pasquale Cerqua
CNIT - Research Unit
Università di Roma "Tor Vergata"
Roma, Italy
pcerqua@libero.it

ABSTRACT
In this work we describe the demo of UPMT - Universal Per-application Mobility management using Tunnels. UPMT is a per-application mobility management solution highly suitable for "ABC" (Always Best Connected) mobility scenarios since it provides mechanisms to fulfill different and independent application requirements as a mobile user roams across different access network infrastructures and Service Providers. Thanks to the tunneling approach, the UPMT solution works as an overlay over current Internet and it does not require any enhancement of the networking infrastructure. With this demo we show how the Linux and Android implementations can choose the best network interface among all available ones independently for each active application, set handover policies and react to IP reconfigurations. The UPMT solution foresees a mobility management node, called Anchor Node. The Anchor Node includes a tool that perform server side monitoring of connected Mobile Hosts and tunnels and it will be shown in the demo.

Categories and Subject Descriptors
C.2.1 [**Computer-Communication Networks**]: Network Architecture and Design

Keywords
Application Level Mobility Management, Vertical Handover, Tunneling

1. INTRODUCTION
The vision of an intelligent and automatic "roaming" among different access networks has been called "Always Best Connected" (ABC) service in [1] almost 10 years ago. The ABC service needs to rely on a solution for seamless mobility management. Many different solutions have been proposed, operating at different levels of the protocol stacks [2]. Despite this considerable amount of past work, the mobility management is still an active area for research and standardization, see for example the activity of IETF DMM (Distributed Mobility Management) WG [3].

UPMT (Universal Per-application Mobility management using Tunnels) [4][5] is an "application level" solution, which provides:

1) support of legacy applications (no need to re-design and/or re-compile the existing applications);
2) independent per-application handover and independent handover of single flows (i.e. IP sockets) within an application;
3) full interoperability with legacy host and with existing networking infrastructure, including private IP numbering and NATs (Network Address Translation).

UPMT is an application level solution and as such, it does not require support from the network layer and from the underlying layers. Application level solutions (like [6][7]) 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 [8][9].

As shown in Figure 1, in its basic configuration UPMT relies on the presence of a mobility management node called "Anchor Node" (AN). The AN performs packet relay between a UPMT aware Mobile Host (MH) and any legacy Correspondent Host (CH). This means that the Correspondent Hosts do not need to be aware of UPMT.

A set of IP in UDP tunnels (one per each physical interface of the Mobile Host) is used to exchange the IP packets between the AN and the MH. The MH and the AN can select on a flow-by-flow base which tunnel to use to exchange packet and consequently choose which physical interface on the MH will be used for each application flow. The approach of using UDP tunnels allows to deploy an "overlay solution" running on top of existing networking equipment and provides the "NAT traversal" feature, i.e. it can work over most of the access networks even if they are using private IP addresses. The SIP protocol is used for mobility management signalling among the involved entities.

The Mobile Host can define its own Anchor Node in its local configuration of the UPMT service. Therefore there can be multiple ANs in the network, they can be hosted by service providers offering mobility services to their customers or by corporate networks providing mobility services to their employees. In turn, each organization can host multiple ANs to balance the load and for geographical distribution. No special networking equipment is needed, having a public IP address on a server is enough to host an Anchor Node. From the point of view of the Mobile Host, being connected to the Internet through a UPMT Anchor Node is like

being connected through any access network with private IP addresses, which grants Internet access using a NAT gateway.
Note that the requirement to have an "Anchor Node" performing the relay of the packets can be relaxed with further UPMT scenarios:

1) MH-to-FH : UPMT aware mobile hosts (MH) and fixed hosts (FH) communicate directly with each other without necessarily relying on ANs for packet forwarding;
2) MH-to-MH : MHs communicate directly with each other without necessarily relying on ANs for packet forwarding.

All the details of the UPMT solution (including the details of the MH-to-FH and MH-to-MH scenarios) cannot fit in this paper and can be found in [5].

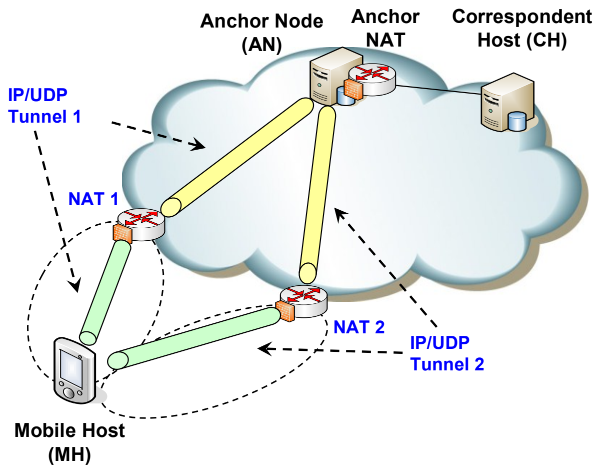


Figure 1: UMPT basic scenario

This demo is about the implementation of the UPMT solution. In the demo we assess the main features of UPMT implementation, for both Linux and Android OS, in particular:

- for each active application we can manually choose the preferred Network Interface Card (NIC) among all possible ones and through all kind of networking technologies, as long as they use IP as networking layer
- for each active application we can set up handover policies based on priority lists or active measurements
- the system reacts to IP reconfigurations of the available interfaces to satisfy the connectivity policies set up by the users
- on the anchor node we run a monitoring tool that keeps tracks all active tunnels and for each Mobile Host the aggregated bandwidth per NIC.

In order to meet our goals we had to implement some enhancements to the IP networking modules of the Linux kernel. These enhancements support the identification and the control of legacy applications sockets in the Mobile Host. Moreover, while the implementation of the IP in UDP tunneling described in [4] was based on a user-space module, we have later on implemented the tunneling in a kernel-space module. The performance evaluation of the UPMT solution, including the comparison between the user-space and the kernel space solution is reported in [10]. Our UPMT implementation is open source and it is available at [11].

2. TECHNOLOGY DESCRIPTION

As shown in Figure 2 the proposed Linux/Android implementation is composed by user space and kernel space components. The coordinating entity is called UPMT Control Entity (UCE), it runs in user space and it has a Graphical User Interface that can be used for user interaction, especially useful for testbed and demo purposes. The UCE and the associated GUI are written in Java. The GUI component is different for the Linux and Android implementation. The Linux UCE GUI (Figure 3) presents the user the list of available network interfaces, the list of active applications and the list of the sockets open by each application. The user is able to decide the interface (i.e. the tunnel) to be used for an application or even for each single socket of each application, "manually" controlling the handovers. The UCE receives notifications about the presence and the status of network interfaces from the Network Manager (used in both Linux and Android) and it communicates with remote UPMT entities (e.g. the Anchor Node) using the UPMT Signaling Agent.

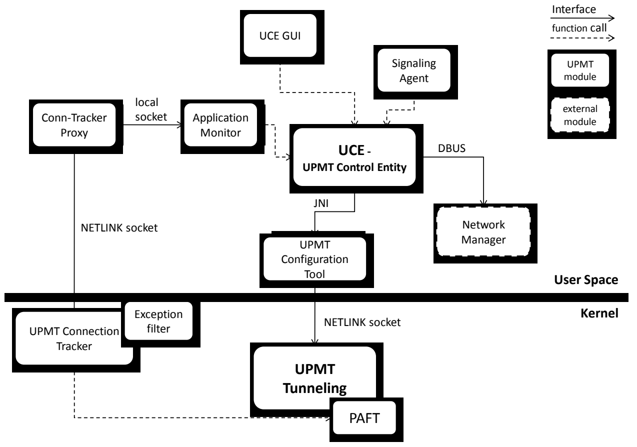
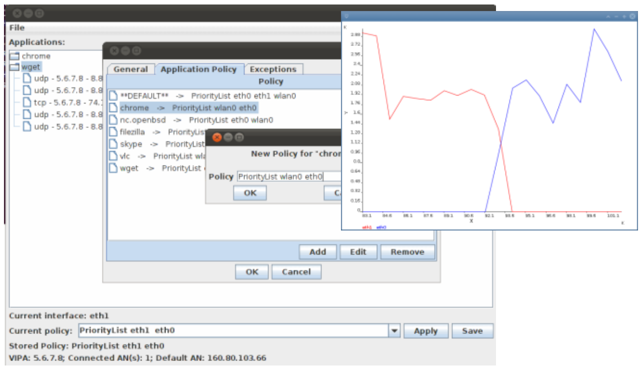


Figure 2: Architecture of implementation on Mobile Host

The UPMT tunneling module is implemented in kernel space for maximum performance. It can be controlled from user space using a so-called Netlink Socket of the Linux OS. Using this channel, the UCE sends commands to the UPMT tunneling kernel module as needed to configure the PAFT (Per-Application Forwarding Table) and to drive the handover process of the applications and of the single flows. Note that the PAFT works at the level of the single flow, i.e. identified by the 5-tuple (protocol, IP src, IP dst, src port, dst port), storing the correspondence between flows and tunnels.



```
chrome
wget
  udp - 5.6.7.8 - 8.8
  udp - 5.6.7.8 - 8.8
  tcp - 5.6.7.8 - 74.
  udp - 5.6.7.8 - 8.8
  udp - 5.6.7.8 - 8.8

**DEFAULT** -> PriorityList eth0 eth1 wlan0
chrome -> PriorityList wlan0 eth0
nc.openbsd -> PriorityList eth0 wlan0
filezilla -> PriorityList
skype -> PriorityList
vlc -> PriorityList wla
wget -> PriorityList

Policy PriorityList wlan0 eth0

Current interface: eth1
Current policy: PriorityList eth1  eth0
Stored Policy: PriorityList eth1 eth0
VIPA: 5.6.7.8; Connected AN(s): 1; Default AN: 160.80.103.66
```

Figure 3: Linux UCE GUI

3. INTERFACE SELECTION POLICIES

The choice of the interface to be used by a given application is controlled by a set of configurable policies. A configuration file is used to configure the policies, which can also be manually changed using the UCE GUI. An example UPMT policy configuration file is shown in Figure 4. A policy can be based only on information about the *availability* of a given interface or of an Anchor Node or it can use *performance* metrics dynamically gathered, for example related to packet delay, packet loss rate, estimates of available bandwidth.

```
***IP BASED POLICIES***
160.80.54.34/32  static wifi0
160.81.0.0/16      noUPMT
160.80.80.15 -AN="ANstatic 160.80.80.150" default
***APPLICATION POLICIES***
firefox priorityList eth0 wifi0 ppp0 any
ssh -AN="ANstatic 160.80.80.150" static wifi0
chrome priorityList   eth0 wifi0 ppp0
app-x     noUPMT
app-y     default
skype VoIP eth0 ppp0
app-z PerfThreshold eth0 ppp0 200:10
```

Figure 4: Example policy configuration file

As shown in Figure 4, the policies can also be applied to both destination IP addresses and to applications (the former ones having the priority if both ones match). It is also possible to indicate a specific Anchor Node to be used, instead of the default one for a specific IP destination address or a specific application. The following set of policies is available:

- **Block**: the packets of the selected application will be not forwarded in any tunnel.
- **Static**: an interface is indicated and will be used for every packet of the selected application. If not available, the policy will be set as Block
- **PriorityList**: the user gives a set of interfaces in order of preference. The first interface available on the list will be chosen, if no one of them is available, the Block policy is selected.
- **Random**: A random interface between the ones available is selected and used for the application.
- **PerfThreshold:** The user provides two thresholds with the maximum allowed value of RTT and Packet Loss rate and a list of interfaces. The first interface among them that fulfil the requirements will be selected. If no interface fulfils the requirements, the policy will be read as a normal PriorityList.
- **VoIP:** This Performance Policy will heuristically combine the RTT and Packet Loss rate to score the different interfaces for VoIP applications. The heuristic is derived from the information reported in [12].

4. LINUX TEST-BED DEMO

The Linux test-bed used for this demonstration consists of the following components:

1. an anchor node *AN* at public IP address 160.80.103.66. *AN* is a PC with Pentium M 1.2 GHz processor, 1.256 GB RAM and Linux kernel 2.6.35.4-upmt.
2. a mobile host *MH* with three network interfaces. *MH* is a laptop with Linux kernel 2.6.35.4-upmt, Core 2 Duo 1.83 GHz, 2 GB RAM. *MH* is equipped with 3 NICs: (i) *wlan0,* a wireless 802.11g NIC connected to a AP on the LAN 192.168.100.0/24; (ii) *eth0,* a Ethernet 100Mbs NIC on the LAN 192.168.100.0/24; (iii) *ppp0,* a HSPA USB card connected on a PPP link and IP address 95.75.196.58. In addition *MH* has a UPMT virtual interface has a fixed virtual IP address *VIpA_fix* 5.6.7.8.
3. a number of legacy correspondent hosts placed over the Internet. In particular, along the demonstration time, MH will connect to a WEB server *CH1* at *www.torvergata.tv,* a backup file server *CH2* at *ubfsrl.ath.cx*; a streaming server *CH3* at *vipnrj.yacast.net;* a FTP server *CH4* at *ftp.archlinux.org* and a laptop *CH5* placed in another LAN running a skype client.

Figure 3 shows the Linux UPMT control GUI that allows to handover an application or even a single socket of an application from one interface to the other. Moreover, the GUI allows the users to set up the per-application policies and display the aggregated traffic over each active interfaces. In particular, in Figure 3 it is showed a handover of the whole traffic generated from the interface eth1 to the interface eth0.
The demonstration described in the reminder of this section is intended to show UPMT capability of performing per-application independent handover and it can be summarized as follows:

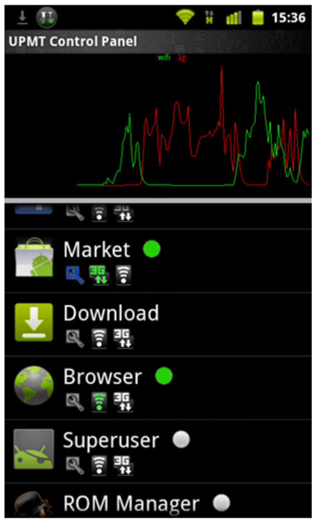
1. UPMT Control Entity is launched. MH associates with *AN* and obtain the association unique virtual IP address *VIpA_an 1.2.3.104*.
2. For each active network interface *MH* automatically creates a tunnel with *AN*.
3. Five applications are started in the same sequence as they are listed in *Table 1*. Four applications are put under UPMT control; one application (firefox) is not handled by UPMT and is routed through *eth0* for the whole demo time. The initial interface for each application is the first interface in the "Handover" column.
4. The applications under UPMT control are handed over as described in *Table 1*, in the same sequence as they are listed.

App	Activity	Traffic type	Handover	CH
skype	video conference	RT video/audio (UDP) + control	ppp0→eth0	CH5 + others
vlc	video streaming	MMS streaming (TCP)	wlan0→ppp0	CH3
scp	remote backup	SSH (TCP)	eth0→wlan0	CH2
chrome	ftp download	FTP (TCP)	wlan0→eth0	CH4
firefox	web tv streaming	HTTP (TCP)	eth0 (noUPMT)	CH1

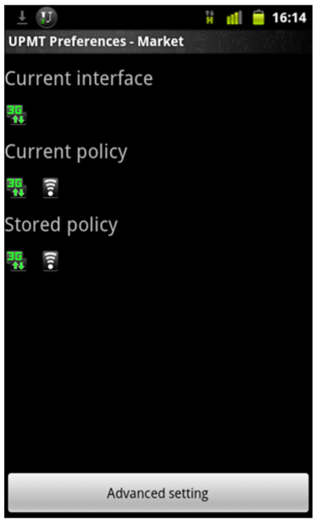
Table 1 Application list in the reported experiment

5. ANDROID TEST-BED DEMO

In the UPMT implementation for Android the MH is an Android device, a HTC Desire HD (ARM Snapdragon S2 processor) with Android 2.3.7 based on CyanogenMod 7.1.0 with the UPMT kernel and all UPMT Java and C applications. Internet access is provided by the two built in NICs of the device: (i) a 3G NIC connected to a 3G operator, (ii) a 802.11g NIC connected to a WiFi access point. The AN is the same as in the Linux demonstration described above. We had to modify the default logic of Android OS which turns off the 3G data connection when WiFi connection is active.
We put a number of legacy applications (among which, Skype, Android Market, Android Browser etc.) under UPMT control and we can perform manual handovers. In *Figure 5.a* the UPMT application manager is shown. The active applications are listed in the UPMT control panel with the green dot. For each active application the outgoing interface can be manually changed by clicking on the NIC icon. In the upper part of the panel, the traffic monitor shows the aggregated traffic generated over the WiFi interface (green line) and the 3G interface (red line). In this picture four different handovers are clearly visible. In conclusion, *Figure 5.b* shows the policy setting for the Android Market application. From this screen it is possible to set up and store the handover policy for each application. In this case the policy is a priority list with the 3G preferred over the WiFi NIC.



(a)



(b)

Figure 5: UPMT Android GUI

6. CONCLUSIONS

In this paper we presented the implementation of UPMT mobility management for Linux and Android OS. To the best of our knowledge, UPMT is the first per-application mobility management solution implemented on real devices, capable to support all legacy applications and to run over existing networks. The demo shows per-application handover executions for data traffic generated by legacy applications like web browsers, video conference software and media players.

7. AKNOWLEDGEMENTS

This work was supported in part by the EU under the project FP7 – 224024 "PERIMETER".
This work was supported in part by OpenTech ENG under the research contract with CNIT "Supporto alla progettazione e realizzazione di una piattaforma mobile innovativa".

8. REFERENCES

[1] E. Gustafsson, A. Jonsson,. "Always Best Connected", IEEE Wireless Communications, Feb 2003.

[2] D. Le, X. Fu, D. Hogrere, "A Review of Mobility Support Paradigms for the Internet", IEEE Communications surveys, 1s t quarter 2006, Volume 8, No. 1

[3] H. Chan (Ed.) "Requirements of distributed mobility management", Internet Draft, draft-ietf-dmm-requirements-01, work in progress, July 2012.

[4] M. Bonola, S. Salsano. "UPMT: Universal Per-Application Mobility Management using Tunnels", IEEE GLOBECOM 2009

[5] S. Salsano, M. Bonola et al., "The UPMT solution (Universal Per-application Mobility Management using Tunnels)", technical report available at http://netgroup.uniroma2.it/TR/UPMT.pdf

[6] H. Schulzrinne, E. Wedlund, "Application-layer mobility using SIP". SIGMOBILE Mob. Comput. Commun. Rev. 4, 3 (July 2000)

[7] P. Vixie (eds.) et al., "Dynamic Updates in the Domain Name System (DNS UPDATE)," RFC 2136, Apr. 1997.

[8] D. Johnson, C. Perkins, J. Arkko, "Mobility Support in IPv6," IETF RFC3775, Jun 2004.

[9] H. Soliman et al., Hierarchical Mobile IPv6 Mobility Management (HMIPv6), IETF RFC4140, Aug 2005.

[10] M. Bonola, S. Salsano, "Per-application Mobility Management: Performance Evaluation of the UPMT Solution", 7th International Wireless Communications and Mobile Computing Conference, IWCMC 2011, 5-8 July 2011, Istanbul.

[11] UPMT home page, http://netgroup.uniroma2.it/UPMT

[12] A. P. Markopoulou, F. A. Tobagi, M. J. Karam, "Assessment of VoIP Quality over Internet Backbones", IEEE INFOCOM 2002