Heterogeneous Mobility in Next Generation Devices: An Android-based Case Study

Ricardo Silva[1], Paulo Carvalho[1], Pedro Sousa[1], and Pedro Neves[2]

[1] University of Minho, Department of Informatics, Braga, Portugal
[2] Portugal Telecom Inovação, Aveiro, Portugal

Abstract. The fast growing of mobile Internet users with the ability of using a wide diversity of access technologies such as Wi-Fi, WiMAX and UMTS/LTE, and the increasing proliferation of mobile devices with heterogeneous network interfaces, require versatile mobility mechanisms providing seamless roaming across those access technologies. Mobility agents such as Mobile IP and Fast MIPv6 are common, however, these solutions still have limitations when dealing with multiple link-layer technology. In this context, the emerging standard IEEE 802.21 provides a framework which enables mobile agents and network operators to improve the handover process in heterogeneous networks. This paper debates and presents a case study of heterogeneous mobility on an Android device, using the IEEE 802.21 framework. Resorting to an experimental testbed, including a modified Android user terminal, the obtained results show that the proposed solution is a first step to successfully accomplish seamless mobility of Android-based devices operating on 3G and Wi-Fi networks.

Key words: Wireless communication; Heterogeneous mobility; Android

1 Introduction

According to the International Telecommunication Union (ITU), in 2008 there were 4.1 billions of mobile cellular subscribers and 1.5 billion of Internet users [1], and, nowadays, this number is still growing up. Most of mobile devices, such as cell phones and notebook computers, have more than one network interface, e.g. Wi-Fi, 3G, WiMAX (Worldwide interoperability for Microwave Access) or Bluetooth [2]. Users want to be "always best connected" to network services and Internet, and with other new emerging concepts, such as Cloud Computing, the need for ubiquitous access is even stressed.

Moreover, there is a wide deployment of wireless networks in public places, homes and enterprises, where many of these Radio Access Technologies (RATs) are overlaid. On the other hand, both operators, service providers and manufactures exhibit a clear trend toward the convergence of communication networks, which will lead to an All-IP network [3]. This can be addressed with vertical handovers (VHO) between the different RATs in order to provide service continuity. Providing heterogeneous mobility allows the mobile device movement to other

network in order to satisfy the service requirements, such as QoS, and provide the best experience to the user.

In this context, the IEEE workgroup 802.21 has defined a framework to support seamless handover over heterogeneous networks. This framework, called Media Independent Handover (MIH), is a middleware in the 2.5 layer and defines a set of mechanisms to facilitate the interaction between the network technologies and the upper layers (e.g. IP) that support VHO. Thus, it provides an abstract layer to interact with lower layers, i.e. hiding the characteristics of each particular technology, e.g. WiMAX, Wi-Fi and 3G. The Media Independent Handover Function (MIHF) has services that provide static and dynamic information about the state and characteristics of the link, helping the mobility management protocols such as Mobile IP (MIP) and Fast MIPv6 (FMIPv6) in the handover decision, as well as services to execute those decisions. This framework helps to decrease packet loss and delay during the handover process, which is important for real-time services, such as Voice over IP (VoIP), that have strict quality requirements in this process.

This paper is organized as follows: the IEEE 802.21 framework and its architecture, including the main functional components and their benefits to the mobility process, are described in Section 2; the Android system is briefly presented in Section 3; a case study with Android supporting heterogeneous mobility between Wi-Fi and 3G networks is presented in Section 4; the experimental tests are reported and discussed in Section 5; and the main conclusions are included in Section 6.

2 IEEE 802.21

Media-Independent Handover (MIH) is the framework defined by the IEEE 802.21 workgroup to support the handover in heterogeneous networks, including both 802 and non 802 networks [4]. This framework aims to facilitate the network discovery and selection process by gathering information locally from the mobile network interfaces and remotely from another MIH framework in the network infrastructure. In addition, it provides a unified interface to get the information and control the network interfaces. MIH users (upper layers) use this unified interface to control the handover, being normally executed based on user or network defined policies.

This framework has three different components: MIH Function (MIHF), Service Access Points (SAP) and MIH users. The MIHF is a logical entity that has three services: the media independent command service (MICS), the media independent event service (MIES) and the media independent information service (MIIS). SAP is an interface that defines a set of primitives to communicate between the MIHF and the other layers. And finally, MIH Users are the entities that use the MIH services and manage the mobility process (handover), being normally a layer 3 mobility protocol (L3MP). These MIH users can be located in the mobile node or in the network.

The MIHF has three SAPs to communicate with other entities: MIH-SAP provides a unified interface to communicate between the MIHF and the higher levels (IP, transport and application layers); MIH-LINK-SAP provides an interface between the MIHF and a specific link layer; and the MIH-NET-SAP provides an interface to exchange information between the local MIHF and a remote MIHF. This communication between two different MIHF entities is made using the MIH protocol, which defines message formats to be used. The advantage of using a unified interface is that the MIH user can manage the handover process regardless of the technology. Therefore, this media-independent framework is a more scalable and efficient method of addressing inter-technology handovers [5]. Moreover, using a common platform to address handovers, each access technology only requires a single extension to ensure interoperability with all other access technologies.

2.1 MIHF Services

In this section, it is described the three services of MIHF that facilitate handovers across heterogeneous networks. These services, interacting with the upper and lower layers, are managed and configured by a fourth service called the management service [5]. Figure 1 shows the architecture of the MIH framework.

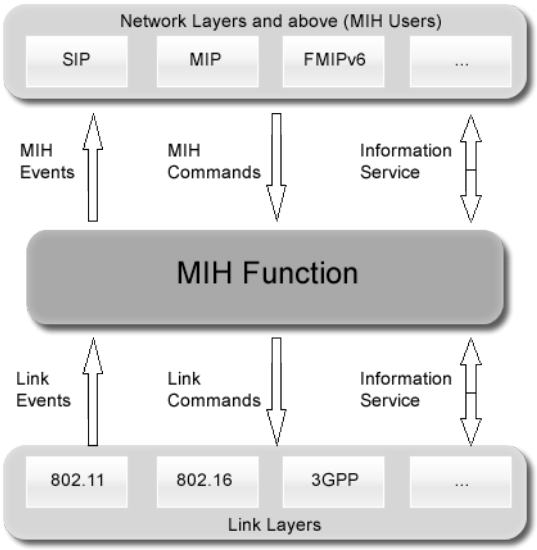

Fig. 1. MIH framework architecture

Media Independent Event Service The MIES provides an asynchronous service to the upper layers by reporting both local and remote events [6]. These

events represent changes that occur in the link layers, such as links status or link quality events. Events can be local or remote: local events occur within the mobile node whereas remote events occur in the network elements. When an event occurs, the MIHF sends an indication to the MIH user that subscribed to that event. The subscribers of the events are the MIH users, which can be a local upper layer or a remote MIHF (or remote upper layer).

There are two types of events: link events and MIH events [4]. Link events are originated from lower layers and propagated to the local MIHF. MIH events can be originated within the MIHF or from lower layers (link events) and are propagated to MIH users. In case that a remote MIH user subscribes to a remote event, the information will propagate from the local MIHF to the remote MIHF by MIH protocol messages and then to the remote MIH user. Some of the events that have been specified by IEEE 802.21 are *Link Up*, *Link Down*, *Link Detect*, *Link Parameter Reports* and *Link Going Down*.

Media Independent Command Service The MIH users use the MICS to control and monitor the link state (lower layers). This service provides commands that can be used to obtain the status of the various links in the node and execute commands to change the state of a link (e.g. power off the network interface). The information obtained from the MICS commands are typically dynamic, such as signal strength, active scan result, etc.

There are two types of commands: MIH commands and link commands. The MIH commands are used by the MIH user to execute commands on the MIHF, and the link commands used by the MIHF to execute commands in the network interfaces. The MIH commands can be local or remote, while link commands are only local [5]. Remote commands are transported by the MIH protocol between two MIHFs.

Media Independent Information Service The MIIS defines a platform where a MIHF located in a mobile device or in the network infrastructure can obtain information about other access networks in the neighbourhood. This information will be used by the MIH user to support the mobility decision. This service aims to obtain information of all access technologies available in the area surrounding the terminal and associated operator, and store this information in an entity called Information Server (IS). This information helps the MIH users in the mobility process. Using the MIIS, the battery power can be conserved, because the mobile devices do not need to perform active scanning of access points.

The MIIS provides dynamic and static information. For example, static information can be the names of the neighbourhood networks, while dynamic information can be network parameters such as radio channel, MAC addresses, security information and other information that help the MIH user handover decision [7].

Management Service This service is composed by three functions that allow managing and configuring MIHF services. These functions are:

– MIH capability discovery: used by the MIH users to discover local or remote MIHF's capabilities in terms of MIH Services.
– MIH registration: used to request access to specific MIH services and to register a mobile device in a MIHF from a network entity (Point of Service).
– MIH event subscription: allows a MIH user to subscribe the reception of events from a local or remote MIHF.

The handover process in the IEEE 802.21 framework is divided into four phases and these are summarized next:

– Handover Initiation: configure handover initiation, based on policies or signal degradation.
– Handover Preparation: scan access points near the terminal, scan available resources, select target network and reserve resources.
– Handover Execution: switch terminal to the target network interface (e.g. trigger MIP).
– Handover Complete: free resources on the previous network.

3 Android

The Android operating system is an open-source platform for mobile devices based on Linux (Android Open Source Project), developed by Google and the Open Handset Alliance (OHA) [8]. This alliance, including major mobile operators, software companies and manufactures, was formed with the aim to innovate the mobile market with new open-source software, and therefore, to allow and foster the cooperation among companies from different areas to respond quickly to the emerging needs of users [9]. Android is available through an open-source license which allows to use and modify the source code.

Android provides a software development kit (SDK), which allows developing Java applications for this platform. An extension to this framework is the Native Development Kit (NDK), which adds C/C++ support (native source code) to Java applications. The Java Native Interface (JNI) lets C/C++ functions be invoked from the Java code and vice-versa. The SDK provides an API to access the system resources and services. These applications run over a Java virtual machine, which is the Dalvik Virtual Machine, optimized for embedded systems [9].

As Android is a Linux-based system, it is possible to compile a C/C++ application to run directly in the system. Using an Advanced RISC Machine (ARM) cross compiler, provided in the Android project tools, it is possible to compile a C/C++ application in Linux. However, given that Android does not use a native window manager, these native applications will not have a graphical interface.

3.1 Architecture

As presented in Figure 2, the Android architecture is composed of five layers: applications, application framework, native libraries, Android runtime and the

Linux kernel [8]. The Linux kernel is the abstraction layer between the hardware and the Android platform. Above this layer, the Hardware Abstraction Layer manages and provides an abstract interface between the hardware drivers and the Android platform, such as 3G interfaces, Wi-Fi or the power management.

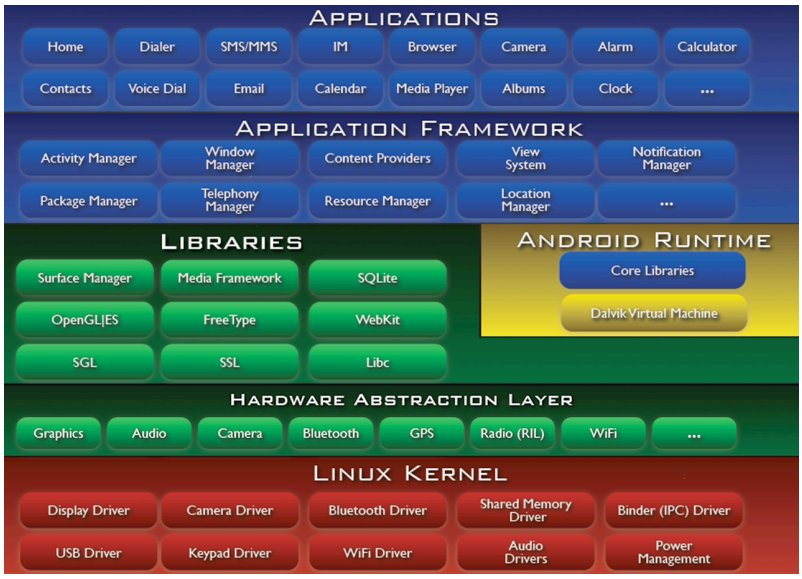

Fig. 2. Android architecture

The set of native libraries supports several standard mechanisms, such as SSL, OpenGL or multimedia codecs. The Android runtime is the core of the Android platform, managing all applications and services running. This layer manages both the lifecycle of the processes (e.g. applications and services) and the corresponding resources (e.g. memory and processing capacity). It consists of two components: the core libraries which are Java APIs to manage data structures or networking access; and the Dalvik Virtual Machine.

The application framework consists of several services that allow managing the system resources, such as window or notification management. Some services are internal to the system and others provide an API to the applications (e.g. GPS or Wi-Fi). Finally, we have the applications that are visible to the user. These applications run over the Dalvik virtual machine and use the application framework to access the services and resources of the system. Each application runs over a single instance of the virtual machine, so the processes are independent of each other.

3.2 Connectivity: Wi-Fi and 3G

The Wi-Fi functionality in the Android system is supported by three components: the driver, the wpa-supplicant and the WifiManager. The driver is

a module that runs in the kernel. The wpa-supplicant is the core component that manages Wi-Fi, provides an API to the Android platform and implements the authentication mechanisms [10]. The WifiManager is a service that runs in the application framework and provides an API to manage the Wi-Fi interface (through the wpa-supplicant).

The 3G functionality is also composed of three components: the TelephonyManager, the Radio Layer Interface Daemon (RILD) and the driver. The driver is a proprietary implementation of the manufacturer (not open-source) and implements an API specified by the Android platform. The RILD is a native service that provides an API between the Android platform and the driver. The TelephonyManager is a service that runs in the application framework and provides an API. This API lacks of important functionality to control the 3G communications [9]. Operations such as turn off the 3G interface or the HSPA connection are not available. These type of operations are only available through internal services, which the TelephonyManager uses but without providing all of its operations.

4 Case Study: Heterogeneous Mobility on Android

This section describes an experimental platform for testing heterogeneous mobility using an Android device on an IPv6 network. The device in use is a Google Nexus One with a custom Android platform. The implementation on Android system is divided into four components: MIHF, Android Interface Manager (3G and Wi-Fi), MIPv6 and Android Mobility Manager (AMM). The MIHF is running as a service in the Android system, allowing the communications among the Android Interface Manager (AIM), the AMM and the remote entities. The other components are explained below.

4.1 Proposed scenario

Figure 3 shows the envisioned scenario considered in this case study. This scenario is composed of a heterogeneous network offering 3G and Wi-Fi access to IPv6 clients. The entities present in the network will be the IEEE 802.21 agent that will control the mobility (Mobility Manager), the Home Agent of the MIPv6 and the streaming server that will provide video over IPv6 to the clients. The handover will be between a 3G network (TMN - a mobile Portuguese operator) and a Wi-Fi network using an Android terminal, with an IEEE 802.21 agent and MIPv6 support (Mobile Node).

Figure 3 shows the main testbed components from the perspective of the mobile terminal. However, from the network perspective there is also one more component, the WiMAX network. In this case study, the WiMAX network provides connectivity between the core network and the Wi-Fi network. WiMAX and 3G allow resource reservation for each client, and also for clients that are connected through Wi-Fi to the WiMAX network. However, the 3G resource

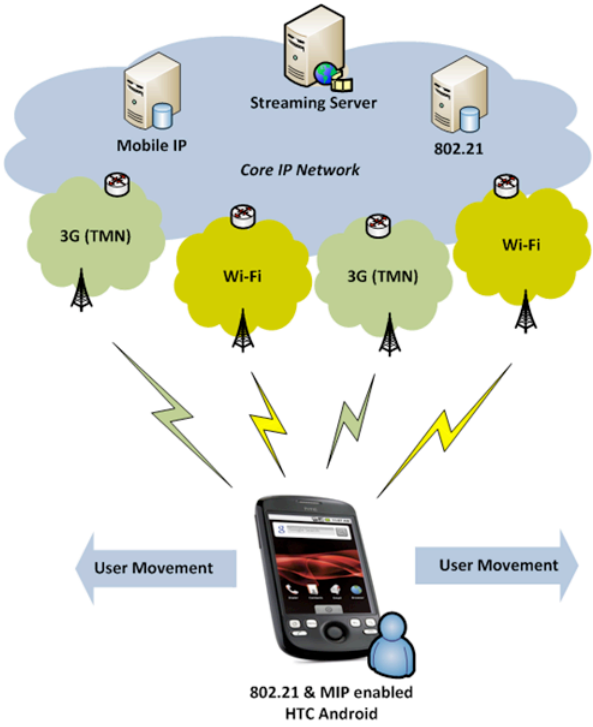

Fig. 3. Proposed test scenario

reservation is based on the operator policies and there is no public API to control the reservation. Therefore, in the present case study, an entity upon receiving a reservation request will simulate the reservation.

4.2 Modifications in the Android Platform

In the normal operation of the Android platform, when the terminal connects to a Wi-Fi network, the platform automatically closes the 3G data connection (UMTS or HSPA). When the Wi-Fi network is unavailable (or the user disconnects from the Wi-Fi network), the platform activates again the 3G data connection. This approach prevents the battery from draining quickly, also freeing resources used in the 3G network. Moreover, it will also prevent the user from spending more money when using the 3G network.

Although the approach mentioned above is convenient to the user, it may be a disadvantage when a make-before-break handover mechanism (controlled by an external entity) is required. In our scenario, there are network entities that control the handover process and therefore, these entities need to control the terminal network interfaces. Moreover, the swap of network interfaces within the Android system is too fast, leading to packet loss. In addition, the system lacks of Mobile IP support to maintain ongoing connections.

In our experiments, the Android source code was modified to avoid disabling the 3G data connection when the Wi-Fi interface connects to a network. Thus, it is possible to have two network interfaces enabled (Wi-Fi and 3G) at the same time. The mobility manager controls when to disable or enable a given connection. As the normal Linux kernel included in the Android terminals does not have all the necessary options enabled, it needs to be recompiled with IPv6, tunnelling and mobility support.

4.3 Android Interface Manager

The AIM integrates the Android network interfaces with the MIHF implementing the Link-SAP interface, as presented in Figure 4. Through the AIM, the MIHF receives events from the network interfaces, such as *Link Up* or *Link Down*, and sends commands to the network interfaces, such as *Power Up* or *Power Down*. The AIM is divided into two components: the AIM for the 3G interface and the AIM for the Wi-Fi interface. Each of these components runs as an independent service, being distinct entities for the MIHF (two Link-SAP). Because the AIM implementation is in Java and the Link-SAP implementation is in C/C++, the JNI is used to integrate the two programming languages.

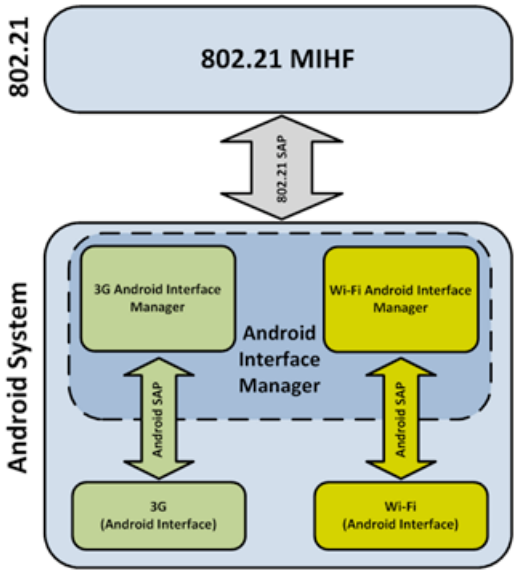

Fig. 4. Android Interface Manager architecture

Wi-Fi AIM This component uses the API provided by the Android service WifiManager. Table 1 shows the integration of the IEEE 802.21 Link-SAP primitives into the Android API.

Table 1. Mapping 802.21 primitives to Android Wi-Fi

Handover Phase	**Function**	**Primitive**	**Android**
HO Initiation	Configure 802.11 thresholds	Link-Configure-Thresholds	MIHF
	Obtain 802.11 link parameters information	Link-Parameters-Report.indication	getRSSI, getLinkSpeed
HO Preparation	Scan the 802.11 Base Station	Link-Action(SCAN)	getScanResults
	Connect to 802.11 network	Link-Action(PowerUp)	WifiEnabled(true)
	802.11 Link Up Event	Link-Up.indication	Wifi-State-Enabled
HO Execution	Trigger Handover	MIH-Net-HO-Commit	AMM
HO Completion	802.11 Link Down Event	Link-Down.indication	Wifi-State-Disabled
	Power-off the 802.11 interface	Link-Action(PowerDown)	WifiEnabled(false)

3G AIM This component uses the API provided by the Android service *TelephonyManager*. However, as explained in Section 3, the *TelephonyManager* lacks of functions that allow the control of the 3G interface. After analysing its source code, it was concluded that this service uses a remote interface to invoke methods on an internal service, through an interface defined in Android Interface Definition Language (AIDL), called *ITelephony*. This internal service, called *PhoneInterfaceManager*, allows a total control of the 3G network interface, such as turn on/off or enable/disable the data connection (UMTS/HSPA).

The *TelephonyManager* has a method called *getITelephony* that allows getting a reference to the *PhoneInterfaceManager*. This method is defined as private. To overcome this problem, the package Reflect was used to change the access permission modifier to public. With this reference and using the *ITelephony* interface, the 3G AIM is able to access all operations provided by the *PhoneInterfaceManager*. Table 2 shows the integration of the IEEE 802.21 Link-SAP primitives into the Android API.

4.4 Mobile IPv6

The used implementation of MIPv6 is the open-source UMIP version 0.4. The normal operation of the MIPv6 [11] does not allow a seamless handover because it only switches to another network when the current network interface disconnects (e.g. stop receiving Router Advertisements). The UMIP version in use was modified to allow a make-before-break handover, with the two interfaces connected, reducing the packet loss during the handover. Thus, our implementation resorts to a TCP socket able to receive commands from other entities in order to

Table 2. Mapping 802.21 primitives to Android 3G

Handover Phase	**Function**	**Primitive**	**Android**
HO Initiation	Configure 802.11 thresholds	Link-Configure-Thresholds	MIHF
	Obtain 802.11 link parameters information	Link-Parameters-Report.indication	getSignalStrength
HO Preparation	Activate UMTS interface	Link-Action(SCAN)	setRadio(true)
	Establish L2 data connection (UMTS/HSPA)	Link-Action(PowerUp)	Enable data connectivity
HO Execution	Trigger Handover	MIH-Net-HO-Commit	AMM
HO Completion	Disable L2 data connection (UMTS/HSPA)	Link-Action(QoS-Deletion)	Disable data connectivity
	Power-off the UMTS interface	Link-Action(PowerDown)	setRadio(false)

obtain the current network interface in use and, if required, trigger the handover to a given interface.

4.5 Mobility Manager

All the mobility process must be managed by an entity which receives information from the terminals and/or other network entities and controls the handover process on the terminals. The Mobility Manager (MM) can be placed in the terminal or in the network, or in both, but only one can make the handover decision. In the present testbed, the MM is placed in the network and controls the handover process of the terminal, called Network Mobility Manager (NMM). This NMM receives information from the terminal, such as signal strength, deciding on the handover strategy to follow.

A MM is also running in the Android terminal (AMM), however, it does not initiate the handover. The aim of the AMM is to receive commands from the NMM and execute the handover locally. These commands trigger the handover on MIPv6 and inform the NMM about the current network interface in use by the MIPv6. The AMM uses specific primitives to communicate with the NMM. The primitive *MIH-Net-HO-Commit* request is sent by the NMM to the AMM to trigger MIPv6 handover. After triggering the handover, the AMM replies with a *MIH-Net-HO-Commit* confirm indicating the result of the operation. The same primitive is used to obtain the current network interface in use.

The NMM is also responsible to reserve resources in the WiMAX and 3G networks. When a client in the 3G network moves to the Wi-Fi network, the NMM reserves resources to this new terminal and then moves the terminal to

the Wi-Fi network, freeing the 3G reserved resources at the end of the handover. In the same way, when moving from Wi-Fi to a 3G network, the NMM reserves resources in the 3G network, moves the terminal and frees the allocated resources in the WiMAX network.

5 Experimental Tests

5.1 Testbed

Figure 5 shows a specific scenario and associated elements in the experimental testbed. The testbed includes three different access technologies: 3G, Wi-Fi and WiMAX. As illustrated, WiMAX provides "last mile connectivity" to/from the Wi-Fi infrastructure.

On the network side, the Home Agent is running the Mobility Manager, the MIHF and the MIPv6. As the testbed is based on IPv6 and the 3G commercial network in use does not support IPv6, a tunnel is created between the Access Router and the Android terminal to transport the IPv6 datagrams over the IPv4 network (IPv6-in-IPv4 tunnelling). The Streaming Server used is the RTSP server called Darwin Streaming Server (open-source) [12]. The video traffic is carried over RTP/UDP. The three components of the network (Home Agent, Access Router and Streaming Server) are running Linux Ubuntu. The Home Agent and the Access Router are configured to send Router Advertisements to the access networks Wi-Fi and 3G, respectively.

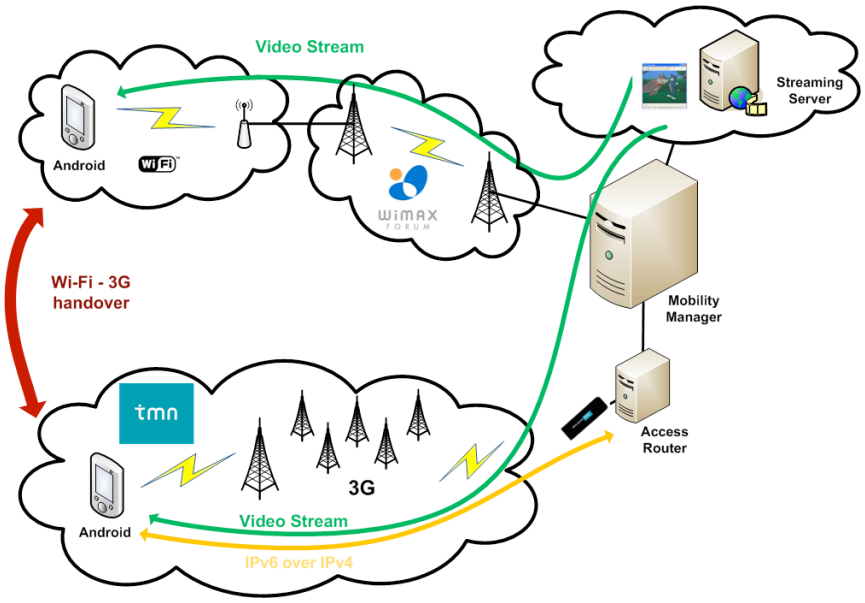

Fig. 5. Testbed

On the client side, the Android terminal, a Google Nexus One with the modified Android platform, version 2.1 and kernel Linux 2.6.32 with IPv6, mobility and tunnelling support, is running the Android Mobility Manager, MIPv6 (MN), MIHF and the Android Interface Manager. The video streaming is received by a simple RTSP client application, with buttons to control the playback and quality of the video. The MM uses the notification service of the Android system to inform the user about both the handover status and the selected target network.

In a normal scenario, the Android terminal is initially in the 3G network (TMN), where the IPv6-in-IPv4 tunnel has to be configured manually between the terminal and the Access Router. The local MIHF registers on the network MIHF, and then the NMM subscribes to events of the terminal and configures the signal thresholds. After the RTSP client is connected to the streaming server, the video stream is sent from the server to the current position of the terminal (initially 3G). When the signal threshold has been exceeded, the NMM initiates the handover preparation (e.g. scan networks, obtain user preferences), reserves the resources in the WiMAX network and connects the terminal to the Wi-Fi network. Then the *MIH-Net-HO-Commit* is sent to the AMM in the terminal, and this triggers the handover on MIPv6 (handover execution). After the handover has been executed, the NMM disconnects the data connectivity in the 3G network (disable UMTS/HSPA) and releases the resources reserved in the 3G network (simulated), finishing the handover (handover completion). The process is the same when the handover occurs from Wi-Fi to 3G.

5.2 Results

In this section, we provide preliminary evaluation results regarding the Android terminal mobility, when the user device moves between 3G and Wi-Fi networks (see Figure 5). Initial experiments showed that when a handover occurs, there is a minimal degradation in quality (loss), without affecting significantly the Quality of Experience (QoE) of the user. To obtain quantitative metrics, such as one-way-delay, packet loss, jitter or data rate, the Distributed Internet Traffic Generator (D-ITG) [13] tool was used to perform test measurements. The Handover Execution Delay, which is the amount of time between sending the Binding Update and receiving the first data packet in the new network interface, is obtained from a program developed within the project. This value represents the time that MIPv6 takes to update its new location and receive data in the new network interface.

To analyse the performance of the solution, the test includes a set of 20 handovers interspersed: 10 from 3G to Wi-Fi and 10 from Wi-Fi to 3G. Each run lasts for 6 seconds, with the handover occurring approximately at half of that time period. These runs are initiated automatically, one after another. The traffic flow in the test simulates a video stream of 1 Mbps, with an average of 128 packets per second. This traffic simulates a real RTSP session with RTP packets of video, with a variable rate.

The charts in Figure 6 show (a) the handover execution delay and (b) the packet loss for each type of handover. The blue bars represent average values

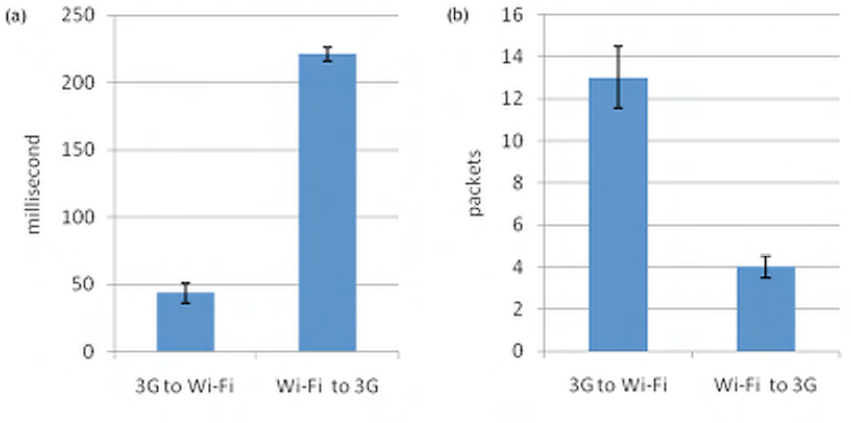

Fig. 6. (a) Handover Execution Delay; (b) Packet loss

and the black bars a confidence interval of 95%. As presented in Figure 6 (a), the Handover Execution Delay difference between the two types of handover is caused by the discrepancy of packet delay in the two technologies. This value is lower from 3G to Wi-Fi because the Binding Update and the first packet received are exchanged in the new interface (Wi-Fi), which have a lower packet delay.

As presented in Figure 6 (b), the packet loss in the handovers from 3G to Wi-Fi is higher than in the handovers in the opposite direction. Note that, D-ITG also counts out-of-order packets as lost packets. As the 3G network has a higher packet delay, when the handover from 3G to Wi-Fi is complete and new packets are arriving in the new interface, some of the old packets continue to arrive to the old interface. The total packet loss ratio is 10.2% in handovers from 3G to Wi-Fi and 3.1% in handovers from Wi-Fi to 3G.

The results above show that the present solution is not yet able to achieve a zero loss during handover. At present, the modified MIPv6 in use still fails to provide a seamless handover. When the handover is triggered on MIPv6, some of the packets addressed to the previous active interface might be dropped, leading possibly to degradation of video quality. However, recall that using the proposed solution packet loss is significantly lower than the one resulting from the normal operation of an unmodified MIPv6 implementation (where the handover only occurs when the current connection is lost). To further improve the current mobility solution, a Fast MIPv6 approach may be used to pursue a seamless handover (no packet loss).

6 Conclusions

In this paper, we have discussed and presented the first steps to achieve heterogeneous mobility of Android-based systems, taking advantage on the recent IEEE 802.21 developments. An experimental testbed has been devised and im-

plemented to assess the handover process across multiple link-layer access technologies. Although our approach still requires improvement to eradicate packet loss during handover events, the obtained results have showed that the proposed solution is a first contribution to successfully accomplish seamless mobility of Android-based devices operating in mixed 3G and Wi-Fi networks.

References

1. ITU - Key Global Telecom Indicators for the World Telecommunication Service Sector 2009, http://www.itu.int/ITU-D/ict/statistics/at_glance/KeyTelecom99.html.
2. de la Olivia, A., Melia, T., Vidal, A., Bernardos, C.J., Soto, I., Banchs, A.: A case study: IEEE 802.21 enabled mobile terminals for optimized WLAN/3G handovers. In: Mobile Computing and Communications Review Volume 11, 2 (2007).
3. Gomes, A., Carapeto, N., Neves, P., Komnakos, D., Sarakis, L.: Handover reference scenarios, requirements specification and performance metrics. Project Deliverable 2.1, HURRICANE - Handovers for Ubiquitous and optimal bRoadband connectIvity among CooperAtive Networking Environments (2008).
4. IEEE Computer Society: IEEE Standard for Local and metropolitan area networks - Part 21: Media Independent Handover Services (2009).
5. Taniuchi, K., Ohba, Y., Fajardo, V., Das, S., Tauil, M., Cheng, Y., Dutta, A., Baker, D., Yajnik, M., Famolari, D.: IEEE 802.21: Media Independent Handover: Features, Applicability, and Realization. In: IEEE Communications Magazine. pp. 112-120 (2009).
6. Dutta, A., Das, S., Famolari, D., Ohba, Y., Taniuchi, K., Kodama, T., Schulzrinne, H.: Seamless Handover across Heterogeneous Networks - An IEEE 802.21 Centric Approach (2006).
7. Tavares da Silva Pinho, A.: Implementation of the IEEE 802.21 in the Network Simulator 3. Faculdade de Engenharia da Universidade do Porto (2008).
8. Android Open Source Project, http://source.android.com/documentation
9. Rogers, R., Lombardo, J., Mednieks, Z., Meike, B.: Android Application Development. O'Reilly (2009).
10. WPA Supplicant, http://hostap.epitest.fi/wpa_supplicant
11. Johnson, D., Perkins, C., Arkko, J.: Mobility Support in IPv6. Request for Comments 3775. IETF (2004)
12. Darwin Streaming Server, http://dss.macosforge.org/
13. Distributed Internet Traffic Generator (D-ITG), http://www.grid.unina.it/software/ITG/