Android-Based Testbed and Demonstration Environment for Cross-Layer Optimized Flow Mobility

Norbert Varga[1(✉)], László Bokor[1], and András Takács[2]

[1] Department of Networked Systems and Services,
Budapest University of Technology and Economics, Magyar Tudósok krt. 2,
H-1117 Budapest, Hungary
varga.norbert89@gmail.com, bokorl@hit.bme.hu
[2] Hungarian Academy of Sciences, Computer and Automation Research Institute,
Kende u. 13-17, H-1111 Budapest, Hungary
andras.takacs@sztaki.hu

Abstract. Nowadays, the spreading and development of multi-access mobile devices together with the proliferation of different radio access technologies make possible to users to actively benefit from the advances of heterogeneous and overlapping wireless networks. This fact and the varying characteristics of mobile applications in means of the required network resources and Quality of Service parameters invoke elaboration of effective flow-based mobility handling algorithms and their cross-layer optimization. Aiming to help research and development in the above topic, we propose an advanced, Android-based testbed and demonstration environment incorporating a cross-layer optimization platform and a flow-aware, client-based mobility management scheme. The testbed relies on MIP6D-NG, which is a client-based, multi-access Mobile IPv6 implementation with different extensions (e.g., Multiple Care-of Addresses registration, Flow Bindings etc.) and an advanced cross-layer communication API. We also introduce an adaptive flow handover system for multi-access environments based on cross-layer information transfer between the applications and the MIP6D-NG core, all implemented and evaluated in the proposed testbed.

Keywords: Android · Cross-layer-optimization · Mobile IPv6 · Flow Bindings · Vertical handover · Heterogeneous network · Mobility management · Testbed

1 Introduction

Recent Android devices are usually provided with multiple network interfaces, thus making able users to reach Internet resources using Wi-Fi or 3G/4G networks. The increasing number of heterogeneous and overlapping radio accesses [1] demand to design and implement algorithms which are able to exploit the available network resources. This motivated us to design and evaluate an extensive, modular, Android-based testbed environment with an advanced flow-aware mobility management framework working in different layers of the TCP/IP stack, a technique for cross-layer information transfer, and an adaptive decision algorithm operating as the engine of this fine-grained mobility management solution. With this system it became possible

V.C.M. Leung et al. (Eds.): TridentCom 2014, LNICST 137, pp. 282–292, 2014.
DOI: 10.1007/978-3-319-13326-3_27

to dynamically bind flows of any Android application to the available access networks and likewise to control the mobility management in the flow level. The decision core of the architecture manages the control of the flows of Android applications by determining the most appropriate interface (i.e., access network) for them. This decision algorithm is an exchangeable module in the testbed and able to optimize the binding of flows and to control the relevant mobility management tasks according to any aspect of the dynamically changing network environment. All the above features of the environment are using a real-time network measurement module continuously providing up-to-date information to the decision engine from the physical, MAC, IP, or even above layers. Based on the current/past network characteristics and the different optimization criteria selected by the mobile user, the decision engine will perform evaluation and in case of need, will send commands to the mobility execution module implemented by MIP6D-NG [2]. We used this testbed to evaluate the performance of different algorithm variants of adaptive cross-layer decision running on Android Smartphones.

The remainder of the paper is organized as follows. In Section 2, we introduce the related work on the existing solutions for cross-layer optimized flow mobility. Both theoretical and practical (i.e., implementation based) researches are depicted here. Section 3 introduces the architecture of our highly customized Android environment. Section 4 in turn details our overall testbed system setup. Section 5 presents our results. In Section 6 we conclude the paper and describe our future work.

2 Background and Related Work

Rapid evolution of wireless networking has provided wide-scale of different wireless access technologies like Bluetooth, ZigBee, 802.11a/b/g/n/p, 3G UMTS, LTE, LTE-A, WiMAX, etc. The complementary characteristic of the above architectures motivates network operators to integrate them in a supplementary and overlapping manner.

Benefits of such multi-access environments can only be exploited if mobility between the different networks is efficiently handled. The Mobile IPv6 [3] protocol family solves the session continuity for mobile nodes on the move. Two MIPv6 implementations are publicly available nowadays. Both the UMIP and the MIP6D-NG are Linux-based open source implementations of MIPv6 and its core extensions (NEMO [4], MCoA [5]). While UMIP [6] is the more mature and widespread solution, MIP6D-NG is a novel, emerging, more extendable implementation comprising some advanced and innovative functions which are not available in UMIP. From this set of pioneering features Flow Bindings [7] and a cross-layer communication API makes MIP6D-NG a promising client based cross-layer optimization supporting multi-access mobility management solution, and that was our motivation to apply it in our testbed architecture. However, further optimization can be achieved by assigning application flows to the appropriate interfaces using intelligent decisions and adaptivity based on the available network resources. Vertical handover and network flow mobility algorithms are the basics of an optimal and cross-layer driven mobility management method for future heterogeneous mobile networks.

2.1 Vertical Handover and Decision Algorithms

The literature on vertical handover (VHO) solutions is extensive. Many papers introduce special handover schemes, network topologies and architectures with decision engines for VHO management (e.g. [8]). One of the most crucial elements of our testbed is the decision engine, thus we start the introduction of the related work on the most important VHO decision techniques and approaches.

We categorize the decision mechanisms based on the input parameters they rely on. A summary about the used parameters for VHO decision can be found in [9], where authors rate VHO algorithms into four categories: RSSI-, bandwidth-, cost-based and combined solutions. In [9], Xiaohuan et al. also present a novel algorithm but it does not rely on other inputs than RSSI. Majority of the existing algorithms use only signal strength as input parameter (e.g., [10]), and authors usually evaluate their solutions based on simulations or analytical calculations. However the RSSI based techniques are the most widespread in the literature, the efficiency of this type of VHO algorithms can be very low, if the parameters of other layers in the TCP/IP stack (e.g., packet loss rate in L3) are not appropriate. Aiming to increase the efficiency of the applied decision scheme we have to use more input parameters, not only signal strength. Authors of [11] and [12] follow this path and also rely on other input parameter types (e.g., monetary cost, bandwidth, and user preferences) beside the RSSI to design more a sophisticated handoff solution. Majority of the existing decision schemes are not capable to support decisions for flow level mobility management, meaning that during the handover all the flows are moved to another interface, hence eliminating the possibility to differentiate between applications neither in the VHO decision nor in the execution phase. For more fine-grained mobility management it is indispensable to define network flows and manipulate them separately during handover events. The concept of network flows allows us to assign flows to applications and link them to different interfaces. We can describe a flow with a 5-tuple: source address and port, destination address and port, protocol type. We focus on the literature of flow mobility in the following section.

2.2 Flow Mobility

Most of the papers in the subject discuss the definition and management of different flows in protocol level [13]. In our testbed the advanced toolset of MIPD6-NG solves all the protocol level questions of flow mobility management by relying on the Flow Binding and MCoA RFCs, so we do not detail this in this paper. Instead, we focus on the flow-aware VHO schemes and existing flow mobility implementations.

In [14], Haw et al. examine a multi-criteria VHO decision mechanism to manage the network flows efficiently. In their flow mobility scenario two flows were defined (FTP and VoIP), however the flow mobility was managed by the operator side and in the context of the mostly theoretical content centric networking (CCN). Contrarily we designed and implemented an IPv6 client-based mobility management which provides more freedom to the end users and relies on the practical IPv6 networking schemes. In [15] also a multi-criteria decision engine is presented. The introduced algorithm supports handover decisions based on network cost, signal strength, packet loss and predefined weight of the flows. This paper introduces theoretical results and doesn't contain evaluation based on real implementations.

The articles above present only recommendations and/or simulation models for flow mobility management. The first publicly available Flow Bindings implementation was designed for Linux distributions by the authors of [16], however their implementation supported only NEMO environments, regular mobile nodes were not able to register or update network flows. Francois Hoguet et al. [17] showed a Linux based flow mobility environment and the possibilities of porting it to Android Smartphones. They used the UMIP's MIPv6 implementation and a proprietary flow binding solution. The authors measured the performance differences between a laptop and an Android Smartphone. This is a real implementation for Android devices, but does provide neither efficient flow mobility management nor complex decision engine. [18] and [19] also introduces a Linux based scheme, but this solution covers only a special NEMO use-case, always moves every flow (its predictive mobility management scheme prohibits separation of individual flows) and does not rely on the Flow Bindings RFCs. Ricardo Silva et al. [20] examine the mobility management on Android systems. They created a custom Android ROM to use the 3G and Wi-Fi interfaces simultaneously. IEEE 802.21 Media Independent Handover framework [21] is applied to support IPv6 based mobility. From this article also the flow mobility and the flow based decision mechanism are missing compared to our architecture.

3 Customized Android Architecture

In our proposed testbed environment the Mobile Node (MN) entity is realized by an Android Smartphone. The overall customized Android Smartphone architecture will be introduced in this section using a bottom-up approach (Fig. 1).

The mobile device must be able to run the MIP6D-NG daemon. MIP6D-NG requires special kernel therefore we modified the kernel part (added Mobile IPv6 support, MIP6D-NG compatibility, modified header files, external modules). To execute these modifications a custom ROM is required. The daemon runs on the native layer of the architecture. The porting MIP6D-NG to Android systems was a non-trivial task, because it required libraries and header files that do not exist on Android OS or if exists, differ from their original GNU Linux implementations. Therefore we created a cross-compiler toolchain which contains the ARM compatible versions of all the necessary components. We extended the NDK stand-alone toolchain[1] with our own libraries and header files. Other important daemons are located in this layer, such as `Pingm6`, `Socat`, and `Lighttpd`. `Pingm6` is a modified Linux `ping6` command which allows testing the flow mobility features. We used `Socat` for the UDP file transfer. `Lighttpd` is an open-source web server optimized for speed-critical environments. We used this for TCP video streaming purposes.

For multi-access communications, the MN needs the ability to communicate via two (3G and Wi-Fi) network interfaces (with IPv6 support) simultaneously. Despite the fact that recent Android devices usually possess multiple radio interfaces, even the newest Android OS versions (Android 4.4) do not allow the simultaneous usage of them. In fact, the built-in mechanisms for network interface management in Android phones are very simple: if a 3G interface is active and Wi-Fi is available, the 3G will

[1] Android NDK toolchain: `http://developer.android.com/tools/sdk/ndk/index.html`

shut down, while if only a 3G network is available, then the Wi-Fi interface will be in down state. Android OS designers are currently pushing a solution which saves battery power so only one interface can be active at the same time. To change the mechanism described above it was necessary to modify the source code of the `Service module` of the Android OS managing network connections. The `Service module` contains the `ConnectivityService.java` where the `handleMessage()` method of `NetworkStateTrackerHandler` class is responsible for the state management of network interfaces: a switch-case statement contains the implementation of each scenario. We implemented a new statement as an extension: if the 3G interface is active and Wi-Fi is available, then 3G should remain active, therefore real multi-access became usable. It meant that the Android OS itself also required modifications.

Another issue to be solved was that the 3G interface doesn't support native IPv6 on most Android devices. In order to solve this problem and also to provide portability of the testbed (i.e., testing and demonstration possibilities over any legacy IPv4 3G network) we were implementing an OpenVPN connection with a bridged interface on the Android Smartphone. Our custom ROM therefore contains the OpenVPN binary and the required `busybox` commands (e.g., `ifconfig`, `route`, etc.). In order to be able to create the bridge interface we needed the following kernel modules loaded: `bridge.ko`, `llc.ko`, `psnap.ko`, `p8022.ko`, `stp.ko`. To configure the environment variables of openvpn, the Smartphone runs the `OpenVPN-Settings` application. The OpenVPN server is located on a router, which provides an appropriate IPv6 prefix for the Android Smartphone 3G connection through the OpenVPN tunnel.

In order to perform the required modifications inside the source code of the Android OS and the kernel, a build environment was created which was able to make a custom ROM image with our MIP6D-NG ready kernel source code and with our modified Android OS code. We used CyanogenMod[2] Android sources and Andromadus[3] kernel tree distribution as a base code platform for our extensions. The result is a custom ROM with Android 4.1.2 and Kernel 3.0.57 with the appropriate patches and settings (MIP6D-NG requires kernel 3.x version, some kernel patches and special configuration). In the Java layer we designed and implemented a modular Android application comprising three main parts. The so called Radio Access Network Discovery Module (RANDM) is designed to measure the different parameters from multiple layers of the available networks (e.g., signal strength, delay, and packet loss). For signal strength measurements we used the `TelephonyManager` API. The packet loss and delay are calculated from the output of `Pingm6`. To run `Pingm6` (which is not a so called system binary) from Java layer we had to use an external library, the RootCommands[4]. RANDM forwards the measurement results to the Handover Decision and Execution Module (HDEM). HDEM is able to direct the Android OS to connect an available WiFi network using the `WifiConfiguration` and `WifiManager` APIs. The Handover Execution module (HEM) communicates with the native MIP6D-NG daemon, creates and sends flow register and flow update messages induced by the advanced decision algorithm. For the cross-layer information exchange a socket based communication scheme was designed and developed. The Handover Decision module (HDM) decides about the necessity of the

[2] CyanogenMod github: `https://github.com/CyanogenMod`

[3] Andromadus github: `https://github.com/Andromadus`

[4] RootCommands external library: `https://github.com/dschuermann/root-commands`

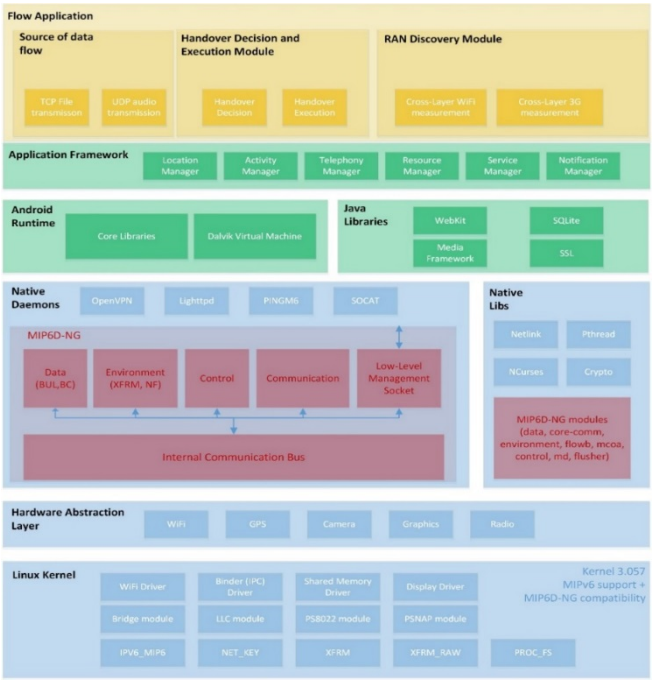

Fig. 1. Highly customized Android architecture

handoff based on the decision algorithm introduced in the next section. HDM directs the HEM to send flow register or update command to MIP6D-NG. The HDM is a modular, exchangeable part of the architecture, thus we can alternate the used decision algorithm very easily.

Currently we have three different decision mechanism implemented in our testbed: Static Flow Assignment (SFA), a purely Signal Strength based [9] (RSSI) and our custom cross-layer optimized algorithm. The SFA is able to register flows using MIP6D-NG, but cannot move them between the available interfaces. The RSSI algorithm reassigns the flows on the basis of the signal strength measurements of the available networks. Contrarily, our scheme relies on cross-layer information. The most important input parameters of our decision algorithm are the actual measurement data, the static information obtained during the network measurements in the currently used networks, and a knowledge database containing the information of all the previously visited networks. The first step of the algorithm is the registration of data flows to the default 3G interface using cross-layer communication between the application and the network layers. After this step starts the phase of passive measurements of Wi-Fi networks. If there are no available networks, the algorithm holds the flows on the 3G interface and waits for the appearance of new Wi-Fi access points. Otherwise starts the cross-layer measurements, in which it measures the signal strength from link-layer, and packet loss, RTT and jitter from network layer. If the parameters of the current measured network are not suitable for the QoS profile, the scheme starts to measure the next available network. If the measured QoS values are appropriate, the MN connects to this Wi-Fi network and moves the corresponding

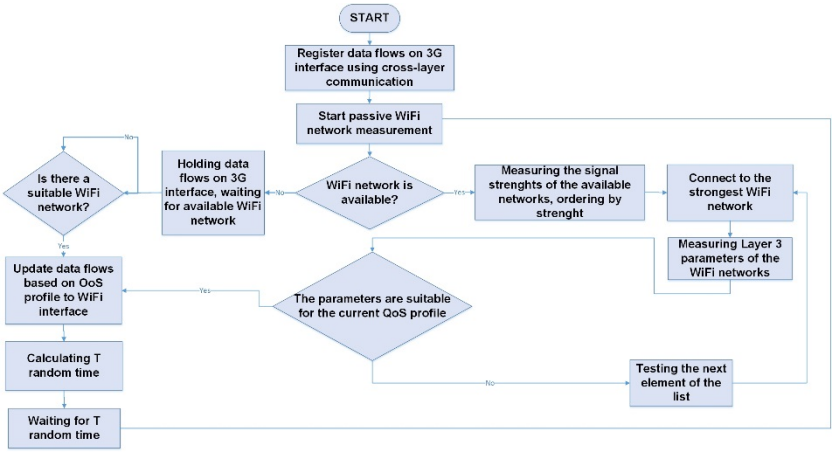

Fig. 2. The proposed cross-layer optimized decision algorithm

flow(s) to the Wi-Fi interface based on the flow(s) QoS profile(s). After that, the application waits for a random time to avoid ping-pong effect of handovers similarly to the solution applied in [22].

The third and last part of the Java layer application is the Source of Data Flows which serves as a simple traffic generator: produces an UPD audio stream and/or a TCP file transfer.

4 Overall Testbed Architecture

Fig. 3 presents the overall architecture of our testbed environment designed and implemented for real-life evaluation of advanced cross-layer optimized, flow level mobility management protocols and algorithms. The Home Agent is realized by a Dell Inspiron 7720 notebook running a MIP6D-NG daemon configured for Home Agent functionality. This entity requires special kernel configuration, which means the need of a MIP6D-NG compatible kernel.

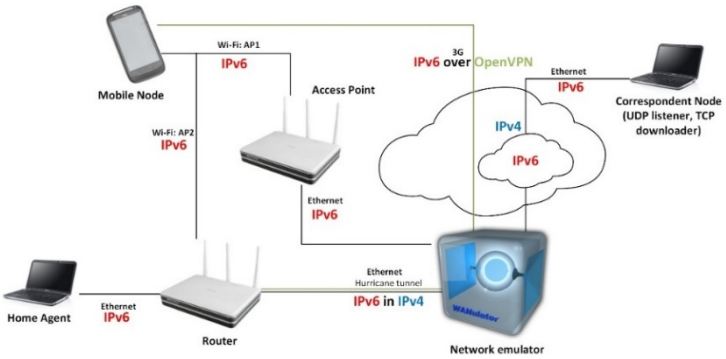

Fig. 3. The overall testbed architecture

A HTC Desire S Smartphone plays the role of the MN. In our testbed the core router is an ASUS WL500 with DD-WRTv24 OS (CrushedHat distribution). Two OpenVPN daemons are running on this router. On one hand an OpenVPN Server provides an appropriate IPv6 address for the 3G connection of Android Smartphone using RADVD[5]. On the other hand an OpenVPN client operates as an IPv6 over IPv4 or IPv6 over IPv6 tunnel, interconnecting the testbed with our IPv6 domain, independently of the router's actual IP access. It means that the overall architecture could be portable and in the worst case only recovers legacy IPv4 connection for the core router. Wanulator[6] network emulator nodes are also applied in the environment. This entity is a Linux distribution which allows us to manipulate the QoS parameters (e.g. delay, packet loss, jitter etc.) of the link to which it is connected (i.e., the wireless connections in the depicted setup). Using Wanulator we are able to evaluate different decision algorithms in any set of network QoS parameters.

5 Results

In order to present the feasibility of our testbed and to evaluate the proposed flow mobility decision algorithm, we implemented three measurement scenarios. In the first test case the significance of the lack of flow level mobility management is presented by the static flow assignment scheme that involves the following:

- the Smartphone connects to the Internet using only the 3G interface
- the Flow Application registers two different type of flows (e.g., TCP and UDP) to the MIP6D-NG, both statically assigned to the only available 3G interface
- a new Wi-Fi AP appears and the application connects to this new AP
- newly started flows, that favor Wi-Fi by their QoS profile can be registered to this newly available Wi-Fi, but ongoing sessions cannot be moved to use the novel access networks: they remain on the 3G connection

Relying on SFA we could communicate simultaneously with two different types of network, but because of the lack of fine-grained mobility, the data flows remain on the interface they were statically assigned to. Using such a static flow assignment algorithm we cannot exploit optimally the available network resources. On the contrary, our algorithm is able to handle the registered flows separately (e.g., we move only the TCP flow to the Wi-Fi (which has higher bandwidth value) and hold on the 3G interface the VoIP flow (which is reactive to the frequent handoffs). By running TCP and UDP tests we measured the amount of the transmitted data (audio and video files) of SFA and our proposal during transmissions of 90 seconds. Fig. 4 left part shows clearly that an algorithm which is able to dynamically move flows between interfaces (i.e., access networks) can transfer much more data. In case of the evaluated schemes in our scenario the average gain was 100.8%. Majority of handover decision algorithms in the literature are (purely) signal-strength based. The efficiency of this type of vertical handovers can be very low if the chosen network shows degraded QoS parameters in network layer of the TCP/IP stack (e.g., packet loss or high jitter occurs). In order to highlight this, our second measurement scenario

[5] Router Advertisement daemon: `http://www.litech.org/radvd/`
[6] Wanulator Network Simulator: `http://wanulator.de/`

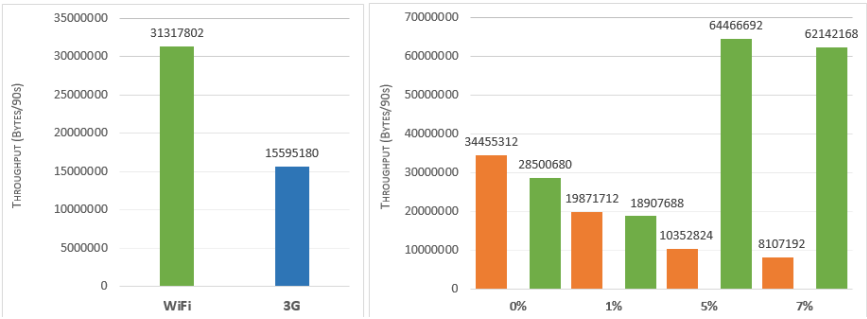

Fig. 4. Comparison between SFA (blue) and our algorithm (green) [left], comparison between RSSI (orange) and our algorithm (green) [right]

manipulates the network level parameters (e.g., packet loss) using the Wanulator box. This scenario from the RSSI algorithm's point of view:

- 3G network is available and the application connects to the 3G network
- the application registers two different types of flow to the 3G interface
- a new Wi-Fi network with good signal strength but high packet loss appears
- the application connects to this Wi-Fi network
- flows favoring Wi-Fi by their QoS profile will be moved to Wi-Fi

In this case the RSSI algorithm moves the data flows to the Wi-Fi interface, but because of the degraded network layer parameters it has much lower efficiency. On the contrary, our algorithm measures the packet loss and holds the flows on 3G interface until it finds an appropriate Wi-Fi network. Fig. 4 right part compares the two schemes: the amount of transmitted data over the TCP flow as a function of the packet loss is depicted. In the two cases (packet loss = 0% and 1%) the performance of RSSI algorithm is better, because our algorithm keeps the data of flows on 3G interface during the measurement session, while RSSI starts to use the Wi-Fi (which has a bigger bandwidth in the first and second test case) earlier. The measurement phase of our algorithm takes 20 seconds, but this will be enhanced in the future. The cumulative average gain of the cross-layer scheme in this scenario was 139%.

6 Conclusions

In this paper we aimed to present a highly customized Android-based testbed and demonstration environment involving a cross-layer optimization platform and a flow-aware, client-based mobility management scheme based on MIP6D-NG. We confirmed the applicability of our testbed by evaluating our flow mobility management proposal with the help of real-life measurements. We performed a comparison between our algorithm and two other scheme implemented from the literature (SFA, RSSI). As a part of our future activities in the designed testbed we are planning to refine our algorithm (e.g., decreasing the measurement period) and combining our client-based approach with network-based mobility management techniques.

Acknowledgement. The research leading to these results has received funding from the European Union's Seventh Framework Programme ([FP7/2007-2013]) under grant agreement n° 288502 (CONCERTO project). The authors are grateful to the many individuals whose work made this research possible.

References

1. Cisco Visual Networking Index, Global Mobile Data Traffic Forecast Update, 2013–2018 (February 05, 2014)
2. Takács, A., Bokor, L.: A Distributed Dynamic Mobility Architecture with Integral Cross-Layered and Context-Aware Interface for Reliable Provision of High Bitrate mHealth Services. In: Godara, B., Nikita, K.S. (eds.) MobiHealth. LNICST, vol. 61, pp. 369–379. Springer, Heidelberg (2013)
3. Johnson, D., Perkins, C., Arkko, J.: Mobility Support in IPv6. IETF (2004)
4. Devarapalli, V., Wakikawa, R., Petrescu, A., Thubert, P.: Network Mobility (NEMO) Basic Support Protocol. IETF (2005)
5. Wakikawa, R., Devarapalli, V., Tsirtsis, G., Ernst, T., Nagami, K.: Multiple Care-of Addresses Registration. IETF (2009)
6. UMIP, Mobile IPv6 and NEMO for Linux (2013)
7. Tsirtsis, G., Soliman, H., Montavont, N., Giaretta, G., Kuladinithi, K.: Flow Bindings in Mobile IPv6 and Network Mobility (NEMO) Basic Support. IETF (2011)
8. Salsano, S., Veltri, L., Polidoro, A., Ordine, A.: Architecture and testbed implementation of vertical handovers based on SIP session border controllers. Wirel. Pers. Commun. **43**(3), 1019–1034 (2007)
9. Yan, X., Şekercioğlu, Y.A., Narayanan, S.: A survey of vertical handover decision algorithms in Fourth Generation heterogeneous wireless networks. Comput. Netw. **54**(11), 1848–1863 (2010)
10. Mahardhika, G., Ismail, M., Mat, K.: Multi-criteria vertical handover decision in heterogeneous network. In: 2012 IEEE Symposium on ISWTA (2012)
11. Kim, J., Morioka, Y., Hagiwara, J.: An optimized seamless IP flow mobility management architecture for traffic offloading. In: NOMS (2012)
12. He, D., Chi, C., Chan, S., Chen, C., Bu, J., Yin, M.: A Simple and Robust Vertical Handoff Algorithm for Heterogeneous Wireless Mobile Networks. Wirel. Pers. Commun. **59**(2), 361–373 (2011)
13. De La Oliva, A., Bernardos, C.J., Calderon, M., Melia, T., Zuniga, J.C.: IP flow mobility: smart traffic offload for future wireless networks. IEEE Commun. Mag. (2011)
14. Haw, R., Hong, C.S.: A seamless content delivery scheme for flow mobility in Content Centric Network. In: 2012 14th Asia-Pacific Network Operations and Management Symposium (APNOMS), pp. 1–5 (2012)
15. Wang, Q., Atkinson, R., Dunlop, J.: Design and evaluation of flow handoff signalling for multihomed mobile nodes in wireless overlay networks. Comput. Netw. **52**(8), 1647–1674 (2008)
16. Ropitault, T., Montavont, N.: Implementation of Flow Binding Mechanism. In: Pervasive Computing and Communications, PerCom 2008 (2008)
17. Hoguet, F.: Network mobility for multi-homed Android mobile devices. Nicta, Eveleigh, Sydney, NSW, Australia, France (2012)

18. Kovacs, J., Bokor, L., Jeney, G.: Performance evaluation of GNSS aided predictive multihomed NEMO configurations. In: 2011 11th International Conference on ITS Telecommunications (ITST), pp. 293–298 (2011)
19. Jeney, G., Bokor, L., Mihaly, Z.: GPS aided predictive handover management for multihomed NEMO configurations. In: 2009 9th International Conference on Intelligent Transport Systems Telecommunications (ITST), pp. 69–73 (2009)
20. Silva, R., Carvalho, P., Sousa, P., Neves, P.: Enabling Heterogeneous Mobility in Android Devices. Mob. Netw. Appl. **16**(4), 518–528 (2011)
21. IEEE, IEEE Standard for Local and metropolitan area networks- Part 21: Media Independent Handover. IEEE (January 2009)
22. Inzerilli, T., Vegni, A.M., Neri, A., Cusani, R.: A Location-Based Vertical Handover Algorithm for Limitation of the Ping-Pong Effect. In: WIMOB 2008 (2008)