Deployable application layer solution for seamless mobility across heterogeneous networks

GIANNI A. DI CARO[1], SILVIA GIORDANO[2], MIRKO KULIG[2], DAVIDE LENZARINI[3], ALESSANDRO PUIATTI[2], FRANÇOIS SCHWITTER[2] AND SALVATORE VANINI[3]

[1] *Istituto "Dalle Molle" di Studi sull'Intelligenza Artificiale (IDSIA)*
`gianni@idsia.ch`
[2] *University of Applied Sciences of Southern Switzerland (SUPSI)*
{`silvia.giordano,mirko.kulig,alessandro.puiatti,francois.schwitter`}`@supsi.ch`
[3] *Forward Information Technologies SA*
{`davide.lenzarini,salvatore.vanini`}`@forit.ch`

Galleria 2, CH-6928 Manno, Ticino, Switzerland

The steady increase in user demand for "all the time, everywhere access" asks for effective and portable solutions for mobility management. Traditionally, global mobility has been addressed by solutions at the network layer. This is the case of Mobile IP and related approaches. On the other hand, this class of solutions presents several drawbacks, such as high overhead, poor scalability, inefficiencies at the transport layer, and inability to take into account user preferences and QoS specifications. These drawbacks seriously affect the ability of the systems to provide seamless handovers, that is, transparent management of mobility and continuity of service when changing network connection point. Moreover, most of the proposed solutions cannot deal with vertical handovers, which involve access points equipped with different radio technology (e.g., WLAN and GPRS). In this paper we propose *WiOptiMo*, an application layer solution for mobility management and seamless handover in heterogeneous multi-technology networks. Working at the application layer, and making use of active and passive cross-layer monitoring, WiOptiMo can overcome most of the drawbacks of previous solutions. It is an integrated approach that can deal effectively with both macro- and micro-mobility. Its design has been driven by the objective of realizing a system which is fully portable, energy and cpu aware, and fully compliant to current standards. The inclusion of autonomic components in WiOptiMo's design empowers the system with basic self-tuning and self-configuring capabilities. We present a detailed description of the system and discuss its pro and cons. Results from real-world experiments are reported to assess the efficiency and the efficacy of the approach.

[*] This work was partially funded by the CTI Swiss programme under the KTI-7640.1 ESPP-ES *Optimised always-on solution* project.

Keywords: Seamless mobility, cross-layering, optimized horizontal/vertical handovers

1. INTRODUCTION

Rapid advances in radio technologies and the increasing in user demand to be always-on even when traveling are giving impetus to the large-scale deployment of wireless services. The moment when users will be put in the conditions to experience the *"all the time, everywhere access"* foreseen in the view of pervasive computing [1] it is not far away. In current and near future scenarios, access to this global Internet is and will be granted by a set of heterogeneous wireless technologies, such as Bluetooth, 802.11x wireless local area networks (WLANs), WCDMA 3G cellular networks, and satellite networks, all relying on the use of the IP protocol. The central challenge consists in providing the mobile user with *seamless connectivity respectful of her/his expectations in terms of received QoS and charged costs*. Seamless connectivity means that terminal mobility is transparent to the user in terms of IP connectivity. The mobile terminal (MT) can change its point of attachment to the network while still being reachable with the same IP address and without incurring in the disruption of the ongoing communications: while moving, (re)connection with new network access points (APs) must be dealt with automatically and transparently, and correct routing of the IP datagrams must be ensured. In case of high mobility and/or demanding applications, this might be a task hard to accomplish meeting user's expectations. During the *handover* from one network connection point to another, service provisioning might incur in significant interruptions and delays, such that the user might experience undesired oscillations and breaks in the received QoS.

So far a number of works have addressed the different aspects of mobility management and seamless handovers. The large majority of the proposed solutions are all based on the use, or are extensions of, *Mobile IP* [2, 3, 4], which is a network layer protocol designed to add mobility management capabilities to the basic IP. However, in spite of its popularity, Mobile IP and related solutions inherently present a number of flaws, limitations, and practical problems. For instance, Mobile IPv4 generates a significant per packet overhead, it has serious problems with firewalls, onward and backward packets can follow different routes creating problems to the TCP, it needs specialized support to implement optimized handovers within the same administrative domain, in its basic form it cannot deal with handovers between access points equipped with different radio technologies (e.g., GPRS and WiFi), etc.. Mobile IPv6 alleviates some of these problems but the full deployment of IPv6 is still far from being a reality. Similarly, alternative solutions based on totally new protocols at the transport or network layers need significant changes in the current implementations of the TCP/IP stack, such as they cannot be immediately deployed on the Internet. On the other hand, *application layer* solutions like the *Session Initiation Protocol (SIP)* [5] are from a practical point of view way more interesting since they do not require any modification of current standards, such as they can be immediately put to work on the global Internet. Moreover, application layer solutions can act as middleware, with the propertyt of dynamically taking into account applications' QoS requirements as well as user preferences. This is not feasible with approaches at the lower layers, since they intrinsically miss the fundamental ability to match user needs and expectations with the ever changing characteristics of the wireless environment she/he is interacting with while moving.

In this paper we present *WiOptiMo*, an integrated application layer solution for mobility management and seamless handovers across heterogeneous IP networks. More specifically, WiOptiMo is designed to: (i) fully support user mobility across wired and wireless networks equipped with different technologies by providing seamless and persistent connectivity, (ii) offer to the user both fully automatic and semi-automatic/assisted ways to choose among the different available network accesses according to her/his preferences and QoS expectations, (iii) avoid unnecessary networking overhead in order to optimize the performance and favor scalability, (iv) run unchanged on the majority of the most popular commercial and network devices.

WiOptiMo has been designed to overcome the main drawbacks of previous work (i.e., Mobile IP and related network layer approaches, reviewed in Section 2), and to be immediately deployable on the Internet and accessible to a very large number of users. In practical terms this means that the installation of WiOptiMo components does not require any modification or addition to the current standard implementations of the TCP/IP protocol stack, nor impose constraints on the side of the MT in terms of computational power, operating system, characteristics of network interface cards (NICs), or on-board radio technology. We pragmatically followed the approach to design a middleware that rely only on *existing technologies, deployed standards, lightweight calculations, and minimal network overhead*. The core of WiOptiMo's architecture is empowered by the active and passive *cross-layer monitoring* of variables of interest, and by the use of *autonomic* components to provide transparent adaptivity and basic self-tuning and self-configuration capabilities.

Different variants of WiOptiMo have been presented in the past in a number of conferences [6, 7, 8, 9]. Here we report a more complete and up-to-date description of the system and we also include some new, unpublished, experimental results from real-world scenarios.

The rest of the paper is organized as follows. Section 2 and its subsections introduce some basic definitions and contain an overview of current literature on mobility management and seamless handovers. We show the general flaws of the solutions proposed so far and provide the rationale to better understand some of the minimalist but pragmatic choices adopted in the design of WiOptiMo. Section 3 and all its subsections are devoted to the description of WiOptiMo and of its components. Section 4 and its two subsections discuss respectively the advantages of WiOptiMo with respect to other solutions (e.g., Mobile IP) and the constraints and limitations for its practical use. Section 5 reports results for real-world experiments concerning measures of efficiency (overhead) and efficacy (timed and robust handovers) of the system, as well as measures to evaluate the possible use of techniques based on active probing packets to estimate the traffic load. Finally, Section 6 provides some conclusive remarks and highlights future work.

2. BASIC DEFINITIONS, MAIN CHALLENGES, AND RELATED WORK

Mobility management protocols and functions aim to support the notions of reachability and seamless handover in mobile environments. Terminal handover between two network access points can happen within the same administrative domain (*micro-mobility*) or between different domains (*macro-mobility*), as well as between access points using equivalent radio technologies (*horizontal handover*) or different radio technologies (*vertical handover*). Most of the proposed schemes for mobility management follow a common general pattern: movement detection at layer 3 or below, selection of next attachment point, discovery and configuration of new IP address, signaling to redirect incoming data packets. An efficient solution is one that provides transparency and low handover latencies, is robust with respect to a wide range of situations, scales well, minimizes user's costs and resource usage, guarantees security, and can be easily deployed given existing protocols and technologies. In emerging 4G IP networks, which are inherently multi-domain, multi-technology, and highly dynamic, reaching these objectives is a complex task. Mobility management protocols and decisions have to react to and/or take into account a variety of conflicting events/issues such as user QoS expectations, locally available network resources and services, policies, security constraints, monetary costs, etc..

In the following of this section we review some of the most important solutions proposed for mobility management. We first discuss the general characteristics of popular protocols for mobility management, such as Mobile IP, mainly pointing out their inherent flaws. Then, in Subsection 2.2, we focus the discussion on the solutions devised to optimize the specific aspects of movement detection and network selection during handover, pointing out the limits for their deployment in practice. The structure of the discussion mirrors that of WiOptiMo, which consists of a general architecture to deal with mobility and reachability (the CNAPT and SNAPT components), and of specific procedures to implement fast and optimized handovers inside these components.

2.1. Macro-mobility and Micro-mobility Protocols

The vast majority of the proposed solutions for mobility management are based on the use of *Mobile IP* [2, 3, 4]. Mobile IP is a network layer protocol that builds on the existing IP to provide global reachability and to make the handover transparent to the layers above the IP layer for each role of the node (client or server). Hiding the change of the IP address when the mobile host is moving between IP subnets is needed to keep TCP connections alive. Mobile IP provides the mobile node with two IP addresses. One of these addresses is its static IP address (home address), which uniquely identifies the node, whereas the second is a dynamic IP care-of address, which provides information about its current point of attachment to the Internet. The assignment of the two addresses allows the IP data packets to be rerouted to the current address of the mobile node. Mobile IP provides signaling procedures for registering the care-of address with a Home Agent, guaranteeing constant reachability. Mobile IP presents slightly different characteristics according to the underlying IP version in use. Limiting the discussion to Mobile IPv4 (since Mobile IPv6 has still a long way before it could be widely deployed due to the tremendous delays in the deployment of IPv6), the basic actions of Mobile IP can be summarized as: (a) use of a discovery procedure from the mobile terminal (e.g., periodic sending of ICMP packets) to identify home and foreign agents, (b) implementation of an authenticated registration protocol to inform the home agent of the care-of address, (c) use of encapsulation and tunneling to forward datagrams from the home address to the address in the foreign network. In its basic form, Mobile IP presents serious problems in terms of performance and scalability (in both of its versions, even if the extended address space and the additional functionalities of IPv6 alleviates some of these drawbacks). The most known of these problems are: (i) triangular routing [10], which is the phenomenon that a packet to a mobile host travels via the home agent whereas a packet from the mobile host is routed directly to the destination, introducing routing inefficiency and creating difficulties for the TCP; (ii) excessive overhead generated by packet encapsulation (IP in IP encapsulation is about 20 bytes [11], that can also result in IP packet fragmentation) and establishment of new tunnels to forward datagrams from the home address to the care-of address; (iii) high overhead and long latencies for the handover, which are due to the procedures of foreign agent discovery and registration of the care-of address with the home agent, and which adversely affect TCP/IP connections [12] and make Mobile IP unsuitable for micro-mobility [13]; (iv) packet dropping by firewalls and ingress routers which get confused by the use of IP addresses which seem either to originate behind the firewall or not being destined for the local network, (v) need for a permanent home IP address and home agent services, which is usually not the case for non-business users. Some of these problems can be partially overcome, but at a price (e.g., see [10]), while for others possible solutions are still under analysis. See also [14] for an extensive analysis on the impact of the use of Mobile IP on both network and transport layer performance.

To the problems just discussed we have to add also the fact that being a network layer solution, Mobile IP acts blindly at application layer, where on the other hand the utilization of the highest available bandwidth, fast handover, and low latency, are essential. The importance of an application-aware approach to future wireless access provisioning is well detailed in [15], which reports an extensive study of the correlation between application-level throughput and physical layer propagation properties. Furthermore, in the current multi-provider scenario, with the growing deployment of hot spots in public places and the availability of wireless WANs such as GPRS, EDGE, UMTS, HSDPA, CDMA 1x2000, EV-DO/EV-DV, etc., a user is faced with the possibility of multihoming, that is, direct attachment to multiple networks. For enterprises this solution offers both performance and availability benefits [16]. On the other hand, multi-homing, and, more in general, the possibility to switch from one provider/technology to another through the use of multi-radio devices or overlay networks [17], raises the problems of provider/technology selection according to user's expectations and of seamless execution of vertical handovers. For instance, while moving between open access WLANs, invoking the DHCP can be sufficient for reconfiguring at the new site. On the other hand, if the user moves from a WLAN to GPRS the reconfiguration is not automatic. Mobile IP has not been designed to manage multi-homing and seamless vertical handovers, such that it needs to be complemented by additional protocols and components. Finally, in current multi-provider scenarios, the need for enabling mobile users to maintain their transport connections and provide permanent reachability while securely connecting back to their target home networks protected by Virtual Private Networks (VPN) is increasing. Again, current Mobile IP standards fall short of this promise for the important customer segment of corporate users behind VPN gateways [18], requiring the support of additional protocols [19].

From the short discussion so far it is apparent that in spite of its popularity, Mobile IP has serious problems in terms of performance and scalability and presents several deficiencies, requiring the inclusion of additional mechanisms and the integration with other protocols. A number of such extensions to the basic behavior and integration with different protocols can be found in literature [20, 19]. For instance, several protocols for micro-mobility that rely on Mobile IP only for macro-mobility have been proposed: *IDMP, Hierarchical Mobile IP, Fast Handoff, Proactive Handoff, TeleMIP, HAWAII, Cellular IP*, etc. (see [21, 13] for overviews). All these solutions introduce additional complexity and require appropriate design/adaptation of the protocols and of their interactions. Moreover, they all work at the MAC and/or network layers, and this is their main constraint in terms of scalability and performance. On the other hand, even if these micro-mobility protocols can boost up Mobile IP's performance for horizontal handovers, they do not solve the problems that Mobile IP has with respect to vertical handovers since it is not designed to deal with technology access switching. This is a problem common to most of the existing mobility management protocols. Working solutions from telecommunication companies (e.g., see [22, 23, 24]) rely on Mobile IP, such that they are all conditioned by the highlighted limitations of Mobile IP and had to implement and manage all the coordination issues of the protocol. Additionally, they had to include a lot of low-level components to make the whole system working with supported operating systems and radio technologies. This also means that any change in network standards or the introduction of new technologies requires substantial updating of the system. Clearly, this is a deprecate situation. An improvement will result from the completion of the specifications contained in the IEEE 802.21 standard, which will support algorithms enabling seamless vertical handover.

A general way to improve over the network layer approach of Mobile IP consists in introducing mobility awareness on a higher layer, where it can be possible to make use of knowledge about traffic and current user expectations to issue optimized roaming decisions. This is for instance the way followed with the popular *Session Initiation Protocol (SIP)* [5], which is an application layer protocol originally designed for establishing and tearing down real-time multimedia sessions but that can also support terminal, personal, session, and service mobility. Application-layer mobility can be deployed much easier than Mobile IP since it does not require changes to the operating system of the participants nodes nor the installation of home agents on the home ISP or dynamic DNS updates. SIP's main drawback consists in the fact that in its basic form it is not really suitable for TCP-based applications but only for real-time interactive and streaming traffic based on RTP/UDP (but for this class of applications it can be very effective and fast intra-domain handovers can be obtained incurring only in minimal packet losses [25]).

Several solutions act at the transport layer and are based on modifications of the basic TCP behavior. *TCP Migrate* [26] is an example of such a way of proceeding. The TCP is modified on both the mobile and correspondent nodes such that it can deal with changes in the IP address during a connection. Using DNS, the correspondent node learns the current address of the MT, with the DNS being updated every time the terminal moves. Apart from the serious drawback of requiring modification of the TCP (drawback common also to most of other proposals working at the transport layer), TCP Migrate can only handle client-server applications but not peer-to-peer ones.

A totally different way to mobility management consists in accepting to revise the current Internet architecture to let it reflect the new challenges posed by terminal mobility and heterogeneity. This is the case of the *Internet Indirection Infrastructure (i3)* [27], which proposes an architecture that offers a communication abstraction based on rendez-vous points in an overlay network. Packet sending and receiving are kept separate at the rendez-vous points: packets are associated to identifiers rather than sent to a destination address. The receiver makes use of the identifier to get the packet delivered. Inside this framework the *Robust Overlay Architecture for Mobility (ROAM)* [28] is built at the application layer without any modification to the TCP/IP legacy stack. From simulation results the approach seems quite effective and robust.

2.2. Techniques for Handover Optimization

Let us first introduce some terminology. The handover of the mobile terminal is said *network executed* if it is totally under the control of the network (e.g., as it is the case between UMTS/GSM/GPRS cells). *Mobile executed handover* is the case in which the handover decision is autonomously taken by the MT. This is the modality prescribed by the currently deployed 802.11 standard for WLANs and this is also the main focus of this work. The handover can be either *soft* (or *alternative*) when it is executed for the sole purpose of *optimization* of the connection cost or QoS, or *hard* (also termed *imperative*), when it is executed due to imminent or present loss of connectivity.

The handover process can be in practice decomposed in three functional components (e.g., see [29]): (i) *Handover Initiation*, (ii) *Network Selection*, and (iii) *Handover Execution*. Handover Initiation consists of the *proactive monitoring* of the current connection and/or of possible alternative connections in order to: (i) effectively *anticipate* or explicitly deal with imperative handovers, or (ii) trigger alternative handovers in order to *optimize* costs and performance. In our case, the *Search* and *Check Activities* of WiOptiMo's CNAPT components (see Section 3) both participate to the Handover Initiation process, which is however mostly focused on the treatment of imperative handovers. Network Selection comprises the procedures to select the new connection point according to the current network context and user's requirements. That is, according to decision metrics like quality of the signal, coverage, access technology, monetary cost, bandwidth, reliability, security, traffic load, etc.. Information about these metrics can be gathered either proactively and/or reactively. In our case, Network Selection is supported by the results provided by the Search Activity. Handover Execution stands for the set of procedures to be executed for the authentication and reassociation of the MT. This pertains to WiOptiMo's CNAPT/SNAPT switching procedure.

All the activities involved in the handover can have a major impact on the overall handover latency, and, consequently, on the QoS perceived by the user. It is apparent that in order to effectively trigger a handover and take an optimized decision in terms of network selection, it is important to collect fresh and reliable information about the ongoing communications and the characteristics of the different candidate networks. This can be effectively realized through *cross-layer monitoring* of the communications status. Layers 1 and 2 can give information on the quality of the connection in terms of signal, errors, losses, and collisions. Layer 3 can report about the existence and the quality of the routing path toward the destination. Layer 4 and upper can provide important information concerning local and end-to-end traffic load. It is clear that gathering such a collection of multidimensional information cannot be realized effectively by the mobile terminal alone. A certain level of information handshaking between the network APs and the MT is required. Nevertheless, at the moment of writing, the current 802.11x standards do not include much help and feedback from and between the APs and the MT in the perspective of supporting timed and optimized decisions. The situation is made even worse by the fact that widely used commercial drivers for 802.11 network interface cards (e.g., Windows' NDIS drivers) do not give standard access to potentially useful variables (e.g., MAC's network allocation vector, discussed later), nor the implementation of current 802.11 standards shows consistent behavior across the different commercial devices [30]. As a result of this situation, the most effective proposals for handover optimization reported in literature are mostly ad hoc solutions that to be widely deployed would require significant changes in standard protocols and drivers. A different way of proceeding is that we followed with WiOptiMo, renouncing to change the standards and/or adapting the single drivers. We trade a bit of efficacy for the possibility of an actual extensive deployment *now* given the current characteristics of implemented standards and commercial devices.

2.3. Overview of Solutions for Handover Initiation and Network Selection

The basic challenges for a handover decision systems consist, in order of priority, in the *robust anticipation* of the interruption of the current connection and in the *optimized selection* of the new connection according to the *network context*. Building-up on previous work for handover in GSM networks, the majority of the approaches for horizontal handover (which have mainly considered WLANs) try to anticipate connection interruption relying on *threshold-based schemes*, including *hysteresis margins* [31] and *dwell timers*, to enhance their robustness (e.g., see [32]). That is, a new AP is selected if it is verified that for a certain dwell time Δ_d, both the quality q_{curr} of the current connection and that of the alternative AP, q_{new}, are respectively below and over assigned threshold values: $q_{curr} < \Delta_r$ and $q_{new} > \Delta_h$. Where Δ_h is such that $\Delta_h = \Delta_r + \delta_h$, with δ_h playing the role of hysteresis margin. If more alternatives are present, the one with best estimated quality is selected. The quality of a connection can be expressed by means of any desired combination of metrics related to physical layer measures of the quality of the received signal, and/or to upper layers measures of the available throughput. It is common practice to restrict the use to measures of signal strength and/or signal-to-noise ratio of the received signal, and to measures of packet losses (e.g., Bit Error Rate (BER), and Frame Error Count (FEC)). Unfortunately all these measures are inherently noisy and show important fluctuations in space and time [33, 34], such as they are often filtered and/or used together with trend measures [35, 36] in order to add robustness to the mentioned threshold mechanisms. An handover is executed only if the new AP seems to provide better connection quality over some robust stability period, avoiding in this way to overreact to temporary falls of the signal and avoiding a *ping-pong effect*. That is, repeatedly switching from one AP to another when the MT crosses the overlapping edges of two cells and the signal from the connected AP shows significant fluctuations. Considering that each AP switch involves time-consuming *authentication* and *reassociation* procedures [37], it is very important to avoid unnecessary handovers by assigning carefully the values of all the involved parameters and thresholds. The problem is that in principle these values should be continually *adapted* to the changing speed of the MT [38] and load status of the APs. However, given currently implemented standards, it is very hard for the MT to derive sound estimates of the AP load status (or, equivalently, of available throughput and end-to-end delay). At this aim, the majority of the approaches are based on the assumption of the knowledge of the 802.11 MAC's *Network Allocation Vector* (NAV), whose total occupation over a time window reflects the busy status of the wireless channel (i.e., the occupied bandwidth) [35, 39, 40]. Unfortunately, NAV information is hardly returned by standard NIC drivers like Windows's NDIS driver. Therefore, its use should be ruled out for practical, commercial implementations. This means that at the moment, when multiple alternative networks are available, it is not possible at network selection time to rely on load information to take a possibly optimized decision in this respect.

Also in the case of vertical handover, similar threshold-based schemes have been adopted in the majority of the studies. However, some important differences exist with respect to horizontal handover in WLANs. The signals from the two different types of networks are not anymore directly comparable, such that it is necessary to adopt independent thresholds Δ_r and Δ_h to assess the quality of the different connections. On the other hand, using dual-mode or multiple NICs, it is possible to monitor alternative connection points while keeping using the current network connection (at the expenses of on-board energy consumption). This can facilitate data collection for robust and well-timed handover decisions. Due to the bandwidth and coverage differences existing between WLANs and WWANs, as rule of thumb most of the decision schemes tend to always favor WLAN to WWAN handovers (*upward link* handovers) in case of low speed mobility, while tend to favor WWAN to WLAN handovers (*backward link* handovers) for high speed mobility. Often, load/bandwidth information is not even took into account, relying on the fact that WLANs bandwidth outperforms WWANs bandwidth even under quite high loads such that it is always preferable to switch to a WLAN (see [41] for an argument against this strategy) in these cases.

2.4. The New Possibilities Open by Forthcoming 802.11x Standards

The activities of the 802.11r, 802.11k, 802.11e and 802.21 IEEE working groups are a joint effort to overcome the major problems arising during the handover process, and to pave the way toward seamless mobility and wireless IP telephony and multimedia streaming. The protocol being drafted by the 802.11r group is expected to dramatically reduce horizontal handover latencies in WLANs. According to 802.11r, the MT will have the possibility to use the current AP to communicate with other APs in range. This will allow the MT to get from candidate APs all the information necessary to take timed and optimized decisions, as well as to set up a security and QoS state at the new AP before making the transition, providing in this way much smoother and faster completion of the handovers. The specifications of the future standard IEEE 802.11r will be complemented by those of the IEEE 802.11k. In fact, in current WLANs MTs cannot rely on any feedback (e.g., load status) from the APs. This fact puts strong limitations on the actual optimization of mobile executed handovers. The standard 802.11k will precisely define what information in terms of radio measurement can be exchanged and the related transmission protocols. Finally, the activities of the 802.11e are addressed at enabling QoS while the already mentioned 802.21 considers the thorn issue of vertical handovers.

3. WIOPTIMO: APPLICATION LAYER SOLUTION FOR SEAMLESS MOBILITY

WiOptiMo [6, 7, 8, 9] is an application layer solution for seamless mobility across heterogeneous networks/providers. It aims to transparently provide persistent connectivity to users moving across different wired and wireless networks. WiOptiMo's design is entirely based on the use of currently deployed network protocols and drivers. It does not require any ad hoc modification or adaptation in current 802.x standards, even if it can be easily adapted to accommodate and exploit their future improvements (discussed in Subsection 2.4). Moreover, it has been designed to have minimal impact on CPU load and, consequently, on on-board energy consumption. All these design characteristics make WiOptiMo immediately deployable at a large scale over a wide range of mobile devices, and it has been patented [42] precisely to commercially exploit these facts.

The system is designed for network applications based on the client-server model. Hereafter, we will consider this model, unless explicitly indicated otherwise. However, as it is discussed later, the extension for peer-to-peer models is straightforward, as it is inherent in the architecture's design. In most of client-server scenarios, the client is located on the mobile node and the server is in the fixed network. While this is not a requirement for WiOptiMo, without loss of generality hereafter we assume this configuration. Mobility management and seamless handover are realized by two application layer modules, the *Client Network Address and Port Translator (CNAPT)* and the *Server Network Address and Port Translator (SNAPT)*. The CNAPT and the SNAPT collectively act as a *middleware* that interfaces the communication between the client and the server applications hiding the mobility to them. They make the client to believe it is running either on the same host where the server is or on a host directly connected to the server in a stable way. The CNAPT can be installed on the same host as the client application or on a different host in the same mobile network. E.g., for a group of people, as a team of consultants or auditors that require mobility while working together, the CNAPT can be installed on only one of the available mobile devices and the whole team can share the seamless handover functionality provided by it. Equivalently, the SNAPT application can be installed on the same host of the server application, on a different one belonging to the same network, or on any Internet server (e.g., in a corporate front-end server, in the home PC, or in any Internet node or router). Thanks to this flexibility, the mobility of multiple users can be handled either using a *star topology*, with central servers managed by telecommunication companies or ISPs and providing large computational capabilities and bandwidth, or using a *distributed topology*, in which every user manages its own mobility by installing the SNAPT in accessible nodes, saving in terms of transmission costs and with the freedom to make at any time the most convenient choice. WiOptiMo's architecture does not suffer from Mobile IP's problems of encapsulation, redirection and signaling. Not a single byte of data is added to the original payload, as no encapsulation is used and no tunneling is required. This not only does not increase data transmission costs, but also solves the IP fragmentation problem that may arise in case of encapsulation of large packets, which can dramatically reduce wireless connection throughput.

WiOptiMo's CNAPT component provides the user with seamless service continuity while she/he moves across different networks based on possibly different radio technologies. Compatibly with available network resources, the handover is realized in an automatic or semi-automatic way without interrupting active ongoing communications and completely avoiding or minimizing user intervention. The system relies on both *soft* and *hard handovers*, also termed here respectively *expected* and *unexpected switches*. Soft handovers are meant for both optimization purposes (e.g., a network with a better tradeoff between costs and performance than the current one happens to be in range), and to anticipate possible interruptions in the current connection (e.g., signal strength is dramatically decreasing and/or frame losses are increasing). On the other hand, hard handovers have to be executed when there is an unexpected interruption in the connection (e.g., a large obstacle is suddenly shielding the signal). In this case, if no alternative available networks are in range, the system freezes the application and periodically keeps monitoring the radio environment, reestablishing the connection and reactivating the application when possible compatibly the application's timeout.

WiOptiMo's behavior is summarized in the diagram of Figure 1. From this figure it is evident that, by decoupling the traditional client/server application from the network connection, WiOptiMo acts as a distributed proxy. Thus, for example, it is possible to use networks like IPV6, ATM, etc., between the CNAPT and the SNAPT, and continue to use IPv4 for the application sides.

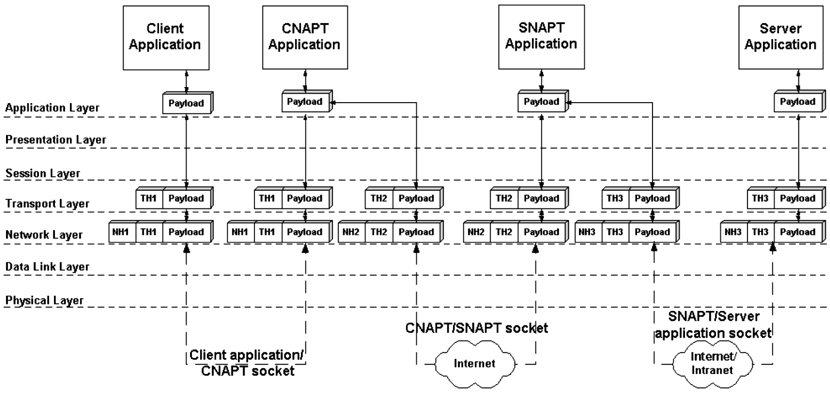


FIGURE 1
CNAPT and SNAPT components. The traditional single socket between Client and Server applications is substituted by three sockets: (1) a local (reliable) socket between the Client and the CNAPT (*Client application/CNAPT socket*); (2) an (unreliable) socket between the CNAPT and the SNAPT (*CNAPT/SNAPT socket*); (3) a (reliable) socket between the SNAPT and the Server application (*SNAPT/Server application socket*).

3.1. The Client Network Address and Port Translator (CNAPT)

The CNAPT (Figure 2) is a software application that at the same time plays the role of servers on the client side and that of clients on Internet side. Figure 2 refers to the case where CNAPT and client application are installed on the same device. In general, they can simply be installed on the same mobile network.

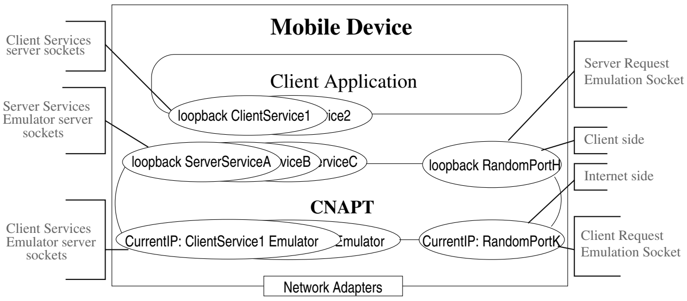

Mobile Device
Client Services server sockets
Client Application
loopback ClientService1
ice2
Server Services Emulator server sockets
loopback ServerServiceA
viceB
rviceC
loopback RandomPortH
CNAPT
Client Services Emulator server sockets
CurrentIP: ClientService1 Emulator
mulator
CurrentIP: RandomPortK
Network Adapters
Server Request Emulation Socket
Client side
Internet side
Client Request Emulation Socket

FIGURE 2
WiOptiMo's CNAPT component and its interactions with the client applications.

When the CNAPT and the client application run on the same machine they use the loopback address to communicate to each other. This way, their communication is not affected by the mobility of the device. In fact any traffic that an application sends on the loopback interface is addressed to the same device independently from the status of the network adapters. Under IPv4 the loopback network has the CIDR address of 127.0.0.0/8. The most commonly used IP address on the loopback network is 127.0.0.1 for IPv4 and ::1 for IPv6. The standard domain name for this address is "localhost". A loopback interface is a type of "circuitless IP address" or "virtual IP" address, as the IP address is not associated with any one particular interface (or circuit) on the device.

The CNAPT emulates the client and the server applications behavior by providing different sockets to handle the different situations.

- A server socket on the client side for each service the client can request via Internet. This server socket listens on the real server service port. It is termed Server Service Emulator Server Socket (SSESS).

- A client request emulation socket on the Internet side for each service request sent via the Internet. This socket is bound to the current IP address of the node and relays packets to the right SSESS provided by the SNAPT.

- A server socket on the Internet side for each client service[1]. This socket listens on the client service emulator port. This port is different from the client service real port in order to avoid a bind error if the CNAPT is installed on the same machine as the client. The socket is termed Client Service Emulator Server Socket (CSESS).

- A server request emulation socket on the client side for each client service request. This socket relays packets to the real client service server socket.

The CNAPT can be installed either on the same machine of the client, or on another mobile host that could act as a connectivity box able to manage a high number of network access technologies and which is linked to the original mobile client through a broadband IP connection. In this case, the user sets up his/her mobile personal area network with access switching capabilities. The CNAPT can emulate simultaneously and in parallel more than one server and can be used by more than one client.

3.2. The Server Network Address and Port Translator (SNAPT)

The SNAPT (Figure 3) is a software application that at the same time emulates the client on the server side and the server on the Internet side. It provides the following sockets:

- A server socket on the Internet side for each server service. This server socket listens on the server service emulator port (the SSESS). This port is different from the server service real port in order to avoid a bind error if the SNAPT is installed on the same machine as the server.

[1] The client provides a set of services that can be used by the server for Publish/Subscribe communication models which are particularly relevant for peer-to-peer networks.

- A client request emulation socket on the server side for each server service request. This socket relays packets to the real service server socket.
- A server socket on the server side for each client service. This server socket listens on the real client service port (the CSESS). The client service emulator server sockets are grouped by a CNAPT ID. If, they use the same port they are bound to different virtual IP addresses to avoid binding errors.
- A server request emulation socket on the Internet side for each client service request. This socket relays packets to the right CSESS provided by the corresponding CNAPT ID.
- An interceptor server socket on the Internet side. It is used by the CNAPT to communicate to the SNAPT the request to dynamically instantiate a server service.
- A control server socket on the Internet side which is used by the CNAPT to communicate with the SNAPT. This connection is used to transmit handshaking packets during the changing of IP address and to optimize the interaction between the client and the server.

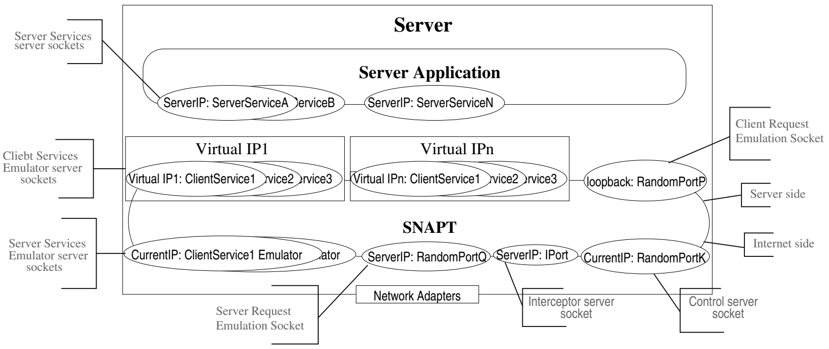

Server Services server sockets
Server
Server Application
ServerIP: ServerServiceA
erviceB
ServerIP: ServerServiceN
Client Request Emulation Socket
Cliebt Services Emulator server sockets
Virtual IP1
Virtual IP1: ClientService1
rvice2
ervice3
Virtual IPn
Virtual IPn: ClientService1
rvice2
ervice3
loopback: RandomPortP
Server side
SNAPT
Internet side
Server Services Emulator server sockets
CurrentIP: ClientService1 Emulator
ator
ServerIP: RandomPortQ
ServerIP: IPort
CurrentIP: RandomPortK
Network Adapters
Server Request Emulation Socket
Interceptor server socket
Control server socket

FIGURE 3
WiOptiMo's SNAPT component and its interactions with the server applications.

The SNAPT can emulate simultaneously and in parallel more than one client request, and it can be used by more than one server. It can be installed anywhere on the Internet. In particular, it can be installed on the same host of the server, or on an additional host placed in the same network of the server or in a different network. Figure 4 shows the case of one client connected to multiple servers through one SNAPT, which can be installed in any Internet node.

3.3. Use of CNAPT and SNAPT for Mobility Management

During regular communication phases (i.e., outside of an handover phase), when a client wants to communicate with a remote server, the CNAPT component processes the client requests and relays them to the SNAPT that manages

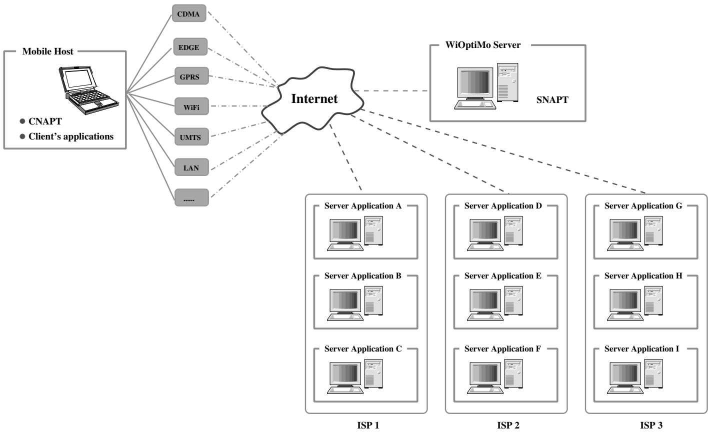


FIGURE 4
Example of one client CNAPT that makes use of multi-radio to get connected to several servers on the Internet through one SNAPT.

the server. CNAPT's intervention is totally transparent to the client, which acts as it is communicating with the server in the usual way. On receiving client requests, the SNAPT processes them and, on its turn, relays them to the corresponding servers. The server response path mirrors the client request path. If the client's mobile device is equipped with multi-radio, such as a GPRS, EDGE, UMTS modem, and an independent WiFi adapter or several independent WiFi adapters, the client can hold simultaneously multiple network connections. In any case, through WiOptiMo, the client and the corresponding remote server exchange data via Internet using exactly one network connection at a time.

During a handover phase, the CNAPT and the SNAPT perform all the required actions for accessing and switching to the new selected connection in a transparent way. The application's data flow is interrupted at the level of the CNAPT, that stops forwarding the outgoing IP packets generated by the client. On its side, the SNAPT stops forwarding all the outgoing packets generated by the server. The packets already stored in the transmission buffers of both the CNAPT/SNAPT sockets will be forwarded respectively to the SNAPT/CNAPT after the completion of the handover.

The *TCP window mechanism* for flow control is exploited to pause the application, on a *local* connection, when no global connection is available. This way of using the TCP and its *buffer space* avoids the needs to allocate extra buffer space (potentially large in case of several running applications) for the outgoing packets during the handover. The use of buffering is common in network layer approaches like Mobile IP to avoid losses during the handover. However, it requires extra resources and does not result in high performances since the applications end up stopping when the TCP window of the socket becomes full, waiting for acknowledgments from the remote end. Moreover, acting at network layer and using buffering has a direct impact upon the end-to-end TCP connection parameters (e.g., retransmission timeout) and can seriously degrade the performance of the application after the handover (see also the results reported in [14, Chapter 4]). On the other hand, for the duration of the handover, WiOptiMo only suspends the *local* TCP connections. It acts on the local connections between the client and the CNAPT and between the SNAPT and the server. The real remote end-to-end connection CNAPT/SNAPT is kept hidden to the client and to the server. The client/server socket is replaced at the application layer by a client/CNAPT socket, a CNAPT/SNAPT socket, and a SNAPT/server socket. With the client/CNAPT and the SNAPT/server sockets, WiOptiMo does not require any application buffer since it can use the buffers provided by the TCP/IP implementation. For the CNAPT/SNAPT socket, WiOptiMo allocates an application buffer to protect the data that are present in its transmission buffer and that can get lost during a hard handover. The dimension of this application buffer is equal to or greater than the TCP congestion window. Therefore, differently from network layer approaches, no buffer space other the minimal, standard one, is allocated for the sockets.

3.4. Cross-layering and Autonomic Design for CNAPT's Handover Initiation and Network Selection

The CNAPT application acts as an application relay system, but also aims at providing persistent and optimized Internet connectivity. The CNAPT implements a handover decision process which consists of two concurrent components: the *Search Activity*, for soft handovers, which proactively searches for new network providers and connectivity, and the *Check Activity*, for hard handovers, which continuously monitors reliability and performance of the current Internet connection. In addition to these two automatic handovers, at any time the user can also manually ask to switch to another available network connection. For instance, this can be useful when the user wants to use a specific network that would not be selected otherwise.

The Search Activity periodically searches for available network connections. The handover to a new network can be triggered in either manual or automatic mode. In manual mode, when the Search Activity finds at least one network that could provide an Internet connection better than the current one (based on the parameters input by the user), it asks the user whether she/he wants to execute the handover. In automatic mode, the Search Activity automatically decides to execute or not the handover on the basis of a set of preferences assigned by the user. However, it might still need some user interaction in case of a connection requiring user authentication inputs. After the handover has been performed, the user can choose to keep the old Internet connection, otherwise this will be closed in order to reduce power consumption.

Similarly to the Search Activity, also the Check Activity follows a periodic activation scheme. It verifies reliability and performance of the current connection every second. If the defined reliability or performance indexes go below a critical threshold (possibly set by the user), or the current network connection is actually experiencing an interruption, the Check Activity tries to switch to a new network provider or to set up a new Internet connection from the same provider if the signal is back (i.e., it was only a temporary problem, like when crossing a small uncovered area during a UMTS connection). If no available network connections are found, the Check Activity notifies the user about the situation and changes its operational mode to *Tunneling mode*, which consists of a continuous search for an available network connection. When it eventually finds it, the connection is re-established and the Check Activity goes back to its normal operational mode. After the handover has been performed, the user can still choose to close the old Internet connection, if it is still alive, in order to reduce power consumption. As discussed in more general terms also in Section 2.3, the main challenges for the Check Activity consist in the *robust anticipation* of the interruption of the current connection and in the *optimized selection* of the new connection according to the *network context* (which is a particularly difficult task in the case of vertical handovers).

To face the complexity and the dynamic and stochastic aspects intrinsic to the task at hand, we had no choice but to adopt for both the Check and the Search Activity a design based on *cross-layering* [43] and *autonomic* components [44]. The need for using cross-layer information in this context has been discussed in Section 2.2, where we also reported about related literature. On the other hand, the characteristics of flexibility, accessibility, continuity, and transparency that are expected to be delivered by a system for seamless mobility, are intrinsic to the paradigm of autonomic computing. Seamless internetwork roaming is a very complex task that requires, transparently to the user, proactive (cross-layer) monitoring of the system performance and behavior for continual adaptation and self-optimization of internal thresholds and decision policies. Both the Check Activity and the Search Activity have been designed according to this combination of cross-layering and autonomic approach. However, so far we mostly focused on the optimization of the Check Activity, which is more critical since it deals with hard handovers. The functional components of a CNAPT module are illustrated in Figure 5. *Cross-layering monitoring* performs active and passive monitoring activities at the different layers. In particular, the Check Activity includes both passive monitoring at the physical and data link layer and active monitoring at the network and application layers. The *User Interaction Module* gets user feedback for the selected handovers and allows the user to input her/his preferences profile through an intuitive interface. The *Optimization Module* is intended for *self-optimization* and *learning*, it will make use of a repository of past experiences to better adapt internal parameters, exploit possible regularities in the user's geographic movements (e.g., this can be relatively easy with an on-board GPS), learn automatically about her/his preferences, derive statistical measures of trend, etc. (however, most of this is still part of our long-term future work; similar issues have been investigated, for instance, in the context of the "Personal Router" project at MIT [45, 46]).

3.4.1. The Check Activity in Detail: Cross-layering at Work

The Check Activity includes both passive monitoring at the physical layer and passive/active monitoring at the network/application layer. We use these measures in cross-validation. In fact, experimental results (see Section 5) seem to indicate that the information obtainable at the application layer can fruit-

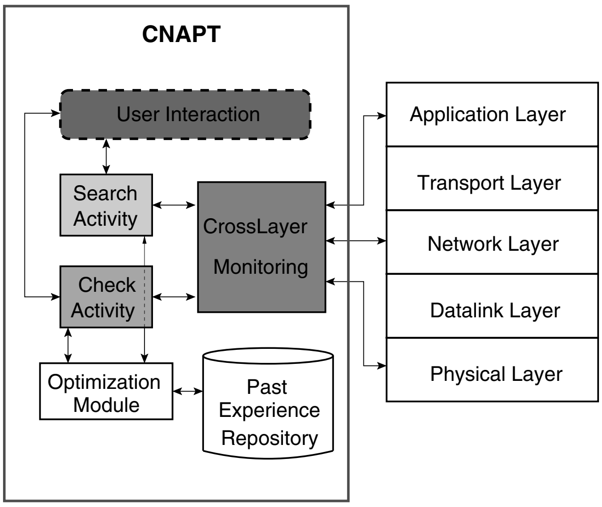


FIGURE 5
Functional components in the WiOptiMo CNAPT module.

fully complement the information from the lower layers to derive more robust estimations of the status of a connection link.

In terms of *physical layer measures*, we made several tests with earlier versions of WiOptiMo to identify the most significant and reliable measures [8, 47].[2] The result was that only measures of received signal strength (RSS) and frame check sequence (FCS) are trustworthy and consistent across the different tested cards. On the other hand, FCS does not really provide more or alternative information with respect to the RSS. Therefore, we decided to focus only on the periodic sampling of raw RSS values in order to keep the computational burden as low as possible and to guarantee high portability of the system.

On the other hand, we decided not to take into account monitoring of potentially useful *data link variables* like the Frame Error Count or the previously cited 802.11's NAV, since we observed that not all NICs allow to read these values using standard APIs. Therefore, for sake of portability, we dropped data link layer monitoring.

Concerning *application layer monitoring*, since WiOptiMo conveys all the traffic exchanged between the client and the server applications on the CNAPT/SNAPT connection, the CNAPT knows at each time whether traffic is being exchanged or not. Therefore, WiOptiMo can easily monitor the connection at the application layer. Unfortunately, passive monitoring of all exchanged data packets would be too expensive (especially for light handheld devices). Therefore, we opted for active application layer monitoring, realized by injecting few control packets in the connection, in the form of ICMP packets, and observing their behavior to derive information about status and quality of the connection.

The flowchart of the actions of the Check Activity is reported in Figure 3.4.1.

[2] In the earlier versions of WiOptiMo [6, 7, 8] we did not make use of application layer measures, which we introduced for the first time in [9] to add robustness to the system without sacrificing neither performance nor portability.

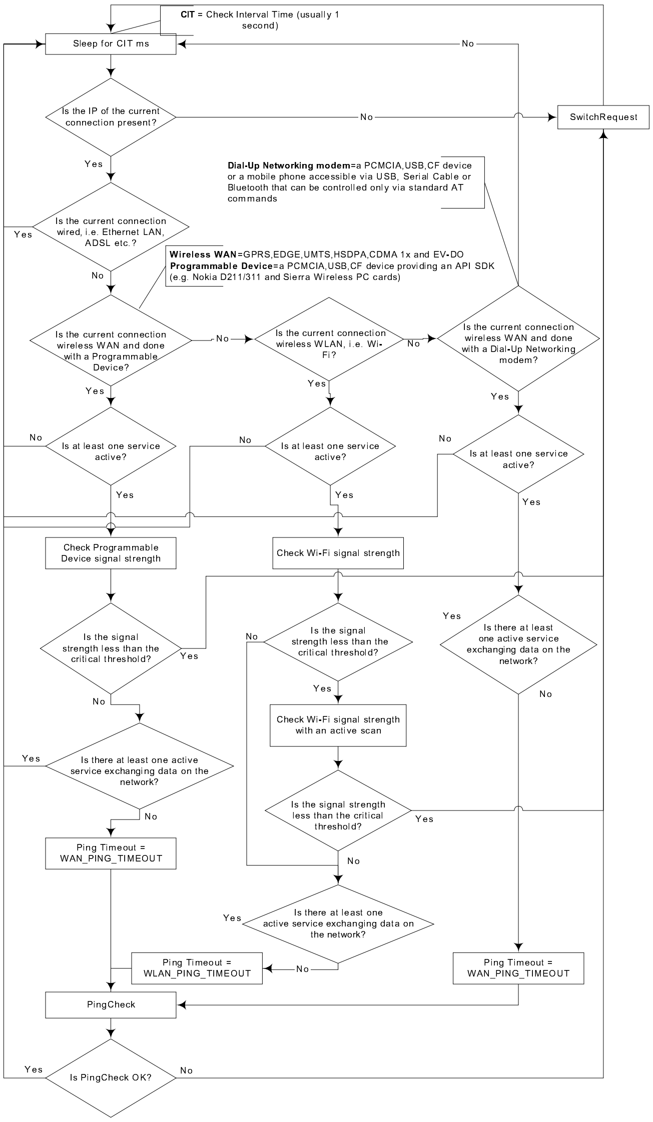

CIT = Check Interval Time (usually 1 second)
Sleep for CIT ms
No
Is the IP of the current connection present?
No
SwitchRequest
Yes
Dial-Up Networking modem=a PCMCIA,USB,CF device or a mobile phone accessible via USB, Serial Cable or Bluetooth that can be controlled only via standard AT commands
Yes
Is the current connection wired, i.e. Ethernet LAN, ADSL etc.?
Wireless WAN=GPRS,EDGE,UMTS,HSDPA,CDMA 1x and EV-DO
Programmable Device=a PCMCIA,USB,CF device providing an API SDK (e.g. Nokia D211/311 and Sierra Wireless PC cards)
No
Is the current connection wireless WAN and done with a Programmable Device?
No
Is the current connection wireless WLAN, i.e. Wi-Fi?
No
Is the current connection wireless WAN and done with a Dial-Up Networking modem?
Yes
Yes
Yes
No
Is at least one service active?
No
Is at least one service active?
No
Is at least one service active?
Yes
Yes
Yes
Check Programmable Device signal strength
Check Wi-Fi signal strength
Yes
Is the signal strength less than the critical threshold?
Yes
No
Is the signal strength less than the critical threshold?
Is there at least one active service exchanging data on the network?
No
Yes
No
Check Wi-Fi signal strength with an active scan
Yes
Is there at least one active service exchanging data on the network?
No
Is the signal strength less than the critical threshold?
Yes
Ping Timeout = WAN_PING_TIMEOUT
No
Yes
Is there at least one active service exchanging data on the network?
Ping Timeout = WLAN_PING_TIMEOUT
No
Ping Timeout = WAN_PING_TIMEOUT
PingCheck
Yes
Is PingCheck OK?
No

FIGURE 6
Flowchart of the CNAPT's *Check Activity* function.

The flowchart shows only the stable core of the behavior, omitting the parts still under development.

The Check Activity first checks the presence of the IP address for the current connection (*network layer check*). If the IP address is present (i.e., from the operating system point of view, the communication device miniport is connected), it checks whether the current connection is a wired (e.g., Ethernet LAN, ADSL, Token ring, FDDI) or a wireless one. This is the *first physical layer check*. In case of wired connection, no additional checks are needed, as the wired connections are considered as always good and reliable connections.

In case of a wireless connection, the Check Activity verifies if the connection is currently in use or not. That is, if there is at least one application or service that needs to access the wireless channel (*first application layer check*). If this is the case, a number of additional actions are performed depending on the type of radio device in use. Radio devices are grouped in three categories according to the level and type of control on the device which is made available from the operating system[3]:

1. *Wireless WAN programmable devices*: GPRS, EDGE, UMTS, HSDPA, and CDMA-1x EV-DO devices providing an API SDK (e.g., Nokia D211/311 and Sierra Wireless PC cards) that gives the possibility to directly establish/destroy the wireless connection and control all its parameters;

2. *WiFi devices*;

3. *Wireless WAN Dial-Up Networking (DUN) modems*: GPRS, EDGE, UMTS, HSDPA, and CDMA-1x EV-DO devices that can be controlled only via standard AT commands.

For the third category, once the DUN connection has been established, it is not anymore possible, in general, to access the connection variables (e.g., the signal strength), as the COM port used for the interaction is locked by the operating system to provide the WAN connection. Therefore, the Check Activity cannot perform in this case any further physical check. On the contrary, for the other two classes of devices, a check on the *received signal strength* is performed (*second physical layer check*). If the RSS is greater than a *critical RSS threshold*, or if it is the case of a modem from group 1, the Check Activity inspects the presence of active applications or services using the network (*second application layer check*). If data are being correctly exchanged, the current connection is considered as *good*. Otherwise, a *ping check* action is started (*third application layer check*). The ping check sends three ICMP `ECHO REQUEST` packets to a reachable host and starts a timeout timer (if it is not feasible to use ICMP, it establishes three TCP connections with a reachable host). If all the three attempts are not echoed within the requested timeout, the connection is considered lost and a handover is requested. Actually, since it is expensive in terms of time and cost (e.g., in case of GPRS connections), the ping check is not performed at every single run of the Check Activity, but with a frequency modulated by the number of timeouts observed during previous runs. This simple but rather effective strategy is similar to that adopted in [48], where the authors show some empirical evidence that the fall of a connection can be robustly assessed after three consecutive frame losses (and without considering any additional metric).

The adopted cross-layer design overall allows robust and timed detection of broken connections. It turns out to be particularly effective in the case of DUN modems and WiFi devices. In fact, for DUN modems physical monitoring is

[3] Hereafter, without loss of generality, the operating system we explicitly refer to is Windows, which is by far the most used operating system for personal computing. Accordingly, the driver for the communications with the network interface card is the Windows NDIS driver.

not possible. Therefore, the application layer monitoring becomes an effective way to bypass this limitation while still proving sound detection of broken connections. On the other side, it is very common for WiFi devices to experience dramatic fluctuations in the value of variables monitored at physical layer. This is particularly true for RSS values. Sudden drops in RSS values can be caused by reasons which can be viewed either as permanent (e.g., the user has moved away from the coverage area of the current connection point) or temporary (e.g., a large truck has just passed close to the user). If the interruption is only temporary, such as it lasts for a very short time, it is definitely counterproductive to try to switch to another connection point as soon as the RSS signal goes below threshold. Therefore, it is very important to get a clear and timed understanding of the nature of the interruption. The fact that the different network cards available on the market show a quite different behavior in case of a sudden drop in the RSS value (e.g., some of them abruptly set the value of RSS to -200 dBm, while others keep on indicating the RSS value before the drop), makes the whole issue even more complex than necessary. We experimentally observed (see Section 5) that the adopted cross-layer design avoids to overreact and/or to react incorrectly to the temporary fluctuations in the value of variables monitored/reported at the physical layer. In fact, the evidence that a short train of ICMP packets has experienced a timeout, once combined with the evidence of a drop in the RSS, provides a robust validation of the fact that the current link is really down and will not be back in the immediate future.

In principle, the use of ICMP packets as probing packets can be envisaged to derive also information about the throughput which is available from the network connection. However, this approach is either unreliable or computationally unfeasible, as it is discussed in Section 5 where we report experimental data concerning this use of ICMP packets.

3.4.2. Autonomic Components

In the previous subsection we have discussed the joint use of application and physical layer measures to build a robust trigger system for handover initiation. The effectiveness of the system depends on the setting of parameters like the number of generated ICMP packets and the timeout threshold. The challenge consists in finding the right balance between fast reaction to loss of connection and minimization of useless (counterproductive) handovers. How long a link should be down before declaring broken the connection? Intuitively, longer is the monitoring period of an apparently broken link, more sound will be the final assessment about the connection status. On the other hand, longer monitoring also means slower reaction times in case of an actually broken connection. The duration of the observation window we used in the ran experiments (resulting from 3 ICMP packets and 1.2 second timeout) seems to be appropriate for walking speeds in indoor environments but they might turn out to be inappropriate for outdoor high-speed mobility (e.g., see [38], where a fuzzy logic scheme is used to adapt the length of an averaging window for the RSS values to user's speed). In more general terms, all the used parameters should be adapted runtime to the changing scenarios in relationship to mobility, ongoing applications, user preferences, available connections, and characteristics of on-board radio devices. For instance, since different NICs have quite different sensitivity, the critical RSS threshold needs to be set according to the specific characteristics of the on-board hardware, and also the relative importance given to physical layer measures should depend on it.

The ultimate target of our work is to make all the adaptation, optimization, and tuning procedures automatic and fully transparent to the user. So far we only made some preliminary but important steps in this direction. Since WiOptiMo is intended to run unchanged on the majority of the portable devices in commerce, the system can *self-detect* the characteristics of the on-board hardware and take the appropriate action flow. That is, it can distinguishing among the three class of devices (DUN modems, WiFi, and WAN programmable devices) and act consequently (see also the flowchart of Figure 3.4.1). This fundamental *self-configuration* feature allows runtime re-configuration after plug and play of NICs. Another important autonomic component regards the *self-tuning* of the critical RSS threshold used in both the Check and Search activities to verify connection reliability. The choice of an effective value for this threshold depends on the current context, in terms of signal characteristics at the specific location and hardware and software configuration. In order to achieve adaptive context-dependent tuning, the CNAPT checks, after each detection of lost connection, if the associated RSS value is lower than that of the currently stored RSS threshold value. If this is the case, the CNAPT assigns this new (lower) value to the RSS threshold, assuming that this is a more appropriate value to represent the inferior limit for successful communications at the current location given the current hardware and software configuration. This adaptation can be started automatically by the system under reception of an event of loss connection, or explicitly by the user after the installation of new hardware to speedup the tuning process. In this case, the user has to move around for a while in order to gather some minimal amount of data.

In the future we plan to empower WiOptiMo with additional autonomic capabilities. In particular, we want to learn from possible regularities existing in the user behavior. At this aim, we intend to maintain a repository of past experience, in order to predict and anticipate to some extent changes in the environment and in the user activities/mobility. For instance, if an on-board GPS is available, the system can store and learn geographical maps identifying good coverage areas together with the characteristics of the network access points providing the coverage. This might result quite useful (and relatively easy to learn) in the case of users traveling along the same routes over and over. The presence of a GPS could also make relatively easy to derive *speed* estimates, which could be used in turn to adapt other parameters like those regulating the duration of the monitoring of an apparently broken link before activating a new handover. Speed estimates could be also used for the tuning of the handover policy, always preferring WANs over WLANs in case of high-speed (on the other hand, without a GPS or other dedicated hardware, speed estimates might result either unreliable or computationally expensive [49, 50]).

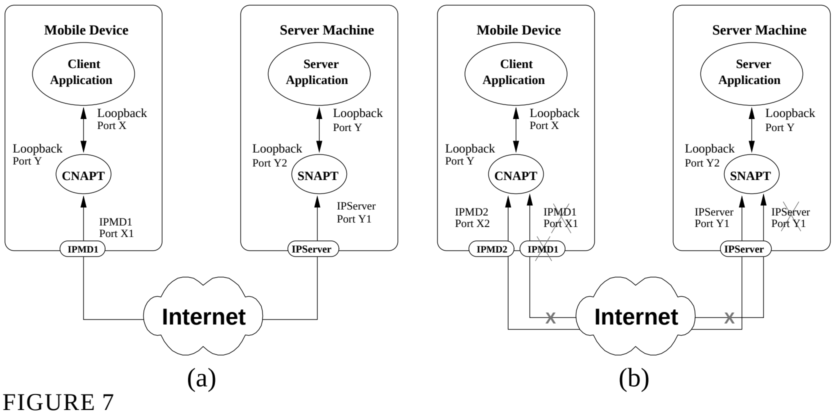
Neverthless, since regularities in the user mobility patterns and/or the availability of a GPS module cannot be given for granted, we assume that a certain degree of user interaction is necessary to optimize system's behavior in relationship to user's conditions and expectations. At this aim we provide an intuitive and friendly *user interface* to facilitate the input and updating of *profiling information*. For instance, through this interface the user can runtime notify the system about her/his speed (e.g., walking, driving) and current needs (e.g., the ongoing download is very urgent, therefore throughput maximization is the priority rather than cost minimization). The user interface can be used also by WiOptiMo to ask for assistance from the user in case of very ambiguous situations.

3.5. A Practical Example

Let us explain the overall behavior of WiOptiMo with the use of a complete example. We consider the general case in which the client's mobile host is able to provide N different Internet connections through N independent network adapters, as in the case of Figure 4.

The CNAPT and the SNAPT are installed respectively on the same host of the client and of the server. The server may provide one or more services: for each of them a server socket listens for client requests. Let the mobile device be connected to the Internet through an access technology T_X via the network provider X, and let IPMD1 be its IP address. Besides provider X, a second network provider, Y, that offers an access technology T_Y, is available at the same time.

The interaction between client and server and CNAPT and SNAPT is shown in Figure 7a (see also Figure 1). The client uses a socket to send to the corresponding server a service request. The CNAPT associates to this request its IP address as source IP address and forwards it to the SNAPT. The SNAPT associates its IP address to this request as source IP address and forwards it to the right server. The server replies in the usual way to the SNAPT and the SNAPT sends back packets having IPMD1 (the CNAPT IP address) as destination IP address. The CNAPT then forwards these packets to the client.



(a)

(b)

FIGURE 7

Example of the functioning of WiOptiMo's CNAPT and SNAPT components. The left-side diagram (a) shows data flow and used IP addresses in the case of stable connectivity between the client and the server. The diagram (b) shows the same situation after a handover has happened.

Let us consider now what happens in the case the network service received from provider X becomes unavailable. The interruption can be sudden (e.g., a LAN/Ethernet cable is unplugged) or can happen gradually, through a progressive degradation of the connection quality. For instance, this is the case when the mobile host is slowly moving away from the coverage area of the wireless provider. Let us assume that while the network connection through provider X is broken, the network service of provider Y is available (see Figure 7b). This can be detected by the Check Activity, which then executes the handover to provider Y. If the handover can be executed before the timeout for the client and server applications expires, then the handover becomes totally transparent to the user's applications, which do not realize that the network (and, possibly, the access technology) has changed. In fact, the IP address used for the remote communication (now changed from IPMD1 to IPMD2) are fully under the control of CNAPT and SNAPT, and are not directly seen by the client and the server.

4. DISCUSSION OF WIOPTIMO'S CHARACTERISTICS AND PRACTICAL IMPLEMENTATION ISSUES

4.1. Positive Features and Advantages Over Mobile IP

In this section we summarize and highlight the positive features of the WiOptiMo approach to seamless handover. We also point out its advantages with respect to other approaches and in particular to network layer approaches like Mobile IP.

All-in-one solution and wide portability: WiOptiMo is an *integrated solution* that combines traditional global mobility management (macro-mobility) with the faster and smaller scale intra-domain mobility (micro-mobility). Moreover, it provides automatic access to comprehensive *vertical mobility* since it is basically technology-independent being an application layer approach: all the network access technologies supported by the mobile device's operating system (both packet- and circuit-switched) can be used by the CNAPT. This means that WiOptiMo can be easily configured to work with newly developed technologies (e.g., LEO satellites) and network standards (e.g., 802.16a). Furthermore, since WiOptiMo has been implemented in *Java* precisely for *portability* purposes, it can be installed on every device (operating system) providing a Java Virtual Machine. Moreover, the computational overhead generated by WiOptiMo's Java processes is very light, such that WiOptiMo can efficiently run on a vast range of handheld devices with different on-board resources. Calculations have been purposely designed to be as light as possible in order to guarantee similar responsivity across different devices and minimize at the same time energy consumption. Portability results also from the fact that WiOptiMo works at the application layer: it does not require modifications to the current implementation of the OSI protocol stack nor requires the introduction of additional sub-layers. WiOptiMo can work over IPv4 and IPv6, as well as over Mobile IP (either v4 or v6) or any micro-mobility solution based on Mobile IP.

Scalability and high performance: Mobility management can be *fully distributed*, as the SNAPTs do not have to be installed on the same network as the host providing the services requested by the user. The SNAPT can operate on any Internet node, i.e., wherever an Internet access is available. This allows high scalability since SNAPTs can be distributed throughout the network and each CNAPT can refer to any SNAPT available on the Internet, e.g. the one with less IP hops to the destination or the one with more available resources. Considered that each CNAPT and the SNAPT can handle multiple client and server services and requests, the possibility to physically distribute CNAPT and SNAPT components can really allow for significant performance optimization.

Scalability and performance result also by the much *reduced overhead* with respect to network layer solutions based on tunneling: not a single byte of data is added to the original payload, as no encapsulation is needed. From one side this does not increase data transmission costs, and from the other side it reduces IP fragmentation problems that may arise in case of encapsulation of large packets, and which might dramatically reduce the throughput of the wireless connection. Moreover, also in terms of signaling, WiOptiMo's overhead is minimal, since it requires only one initial registration between CNAPT and SNAPT, and not a registration for each new connection (as mobile IP imposes for each new visited IP domain). Better performance and adaptivity compared to network layer solutions also come from the use of the TCP window mechanism to locally suspend the connections and avoid the allocation of extra buffers and end-to-end connection interruption, as discussed in 3.3.

Transparent management of security: The most popular tunneling protocols used by commercial VPNs (which are an important segment of the network market) *encrypt* everything at the network layer or above. Therefore they can have troubles with mobility solutions that need to access network information. On the other hand, they can easily work with application layer approaches such as WiOptiMo. More in general, being an application layer solution, WiOptiMo does not require the implementation of any custom security feature and can be integrated with all the most popular commercial products at transport or lower layers to address security issues. It can work in a transparent way over IPSec, L2TP (Layer Two Tunneling Protocol), or other VPN protocols.

Application/user awareness: WiOptiMo is application aware. Thus it can adapt to different scenarios [16] and take into account at network selection time user's expectations in terms of received QoS versus both monetary costs and available network resources. WiOptiMo provides both automatic and semi-automatic/assisted ways to select network connections. That is, it allows direct user intervention if she/he wants it. Generally speaking, the overall architecture of a WiOptiMo's CNAPT consists of several functional components, each dealing with a specific issue relevant to provide seamless and optimized mobility management (see Figure 5). In particular, the Search and Check activities rely on data resulting from *cross-layer monitoring* (i.e., active and passive monitoring and data collection at physical, network, and application layers), from *user interaction*, who can input her/his preferences, and from a *repository of past data* to smoothly adapt internal parameters.

4.2. Application Domains and Practical Implementation Issues

WiOptiMo's CNAPT and SNAPT have a CNAPT ID and a SNAPT ID that are used for authentication purposes every time a CNAPT/SNAPT connection has to be established or re-established (reconnection after a switch). At each time the SNAPT knows the CNAPT ID and the current IP of all connected CNAPTs.

WiOptiMo is able to grant the seamless handover to three types of services:

1. Internet HTTP/HTTPS/FTP browsing services accessed via a Web browser or via any application that can use a Web browser proxy setting (e.g., Windows Media Player, WinAmp and other applications that can use the Microsoft Internet Explorer proxy settings). The proxy setting can be done automatically and transparently by WiOptiMo (e.g., using Microsoft Internet Explorer) or can be done manually by the user, who only needs to assign the HTTP/HTTPS/FTP proxy to "127.0.0.1:12345" (IPv4 case). In this way all the web traffic is conveyed through the Virtual Web Channel (see [7] for details).

2. Dynamic services (black box services), which are the TCP/UDP services that are unknown in terms of used ports and IP addresses and that, in general, do not allow to set a proxy (e.g., Skype, MSN Messenger, etc.). These services are not pre-allocated by the CNAPT and SNAPT and are created on-demand. To handle dynamic services, WiOptiMo uses a client-side software module, specific for each operating system, called *Interceptor*. This module intercepts all the network activity generated by each client application and interacts with the local CNAPT. For instance, the Interceptor signals to the CNAPT that an application wants to open a socket and redirects the application socket to it. Making use of a control channel, the Interceptor sends to the CNAPT the real destination IP address, the port, and the type (TCP or UDP) of the application socket. CNAPT and SNAPT then dynamically interact to provide the connection to the real server and continue to work as usual, supporting seamless handovers. When the Interceptor is used, the client does not have to change the IP address of the server: it can continue to use the original IP address of the server but its outgoing packets are redirected to the local CNAPT. With the Interceptor, the server service information (IP, port and type) is not statically known in advance by CNAPT and SNAPT, but it has to be retrieved dynamically.

3. Known and configurable services, which are the TCP/UDP services for which the user knows the ports and the server IP addresses to be used, and for which it is possible to set the server IP address to the CNAPT IP

address (e.g., some games, e-mail, FTP clients, corporate market applications and, more in general, each type of services in which a proxy can be set). For this kind of services an off-line joint setup of CNAPT and SNAPT is requested because they have to pre-allocate these services. The constraints to allow the use of WiOptiMo in these cases are:

(a) The IP address of the server can be modified in the client. That is, it is not hard-coded such that it is possible to define the server IP address either through a configuration file or at run time through a client input mask filled in by the user.

(b) The source IP address of the provided client services is not hard-coded. It must be possible to assign this IP address to the loopback address through a configuration file or a client input mask, as in the previous case.

(c) The SNAPT knows the server IP address, the server service ports and the server service types (e.g., connectionless or connection-oriented). When the SNAPT is installed on the same machine as the server, in order to avoid bind errors, it emulates server services using ports that are different from the ports used by the actual services.

(d) The CNAPT knows the IP address of the SNAPT that it has to use to connect to a certain server.

(e) The CNAPT knows the server service ports, the SNAPT service emulator ports, and the server service type.

(f) The CNAPT knows the client IP address, the client service ports, and the client service types. The CNAPT emulates the client services providing a set of emulation services operating on emulation ports which are different from the real service ports to avoid bind errors when the CNAPT is installed on the same host of the client.

(g) The SNAPT, for each CNAPT that provides client services, knows the client service ports, the CNAPT service emulator ports, and the client service types. These data are grouped by means of the CNAPT ID.

(h) If client services are used, the SNAPT must have a set of Virtual IP addresses belonging to the same network to which it belongs to. These addresses are used to accept simultaneous connections from different CNAPTs and manage clients providing their client services on the same port while avoiding bind errors. This constraint holds also for the previous case of Dynamic services.

Condition (a) is not very restrictive, as it is the commonly used approach with proxies. Furthermore, this is a requirement for the client side, which is traditionally more open source even when distributed for commercial purposes.

Condition (b) must hold to avoid client service interruption: if the source IP address would be set automatically as the IP address of the mobile device, when this address changes after a handover, client services would be interrupted. On the other hand, condition (b) not need to hold if the clients are running on hosts different from CNAPT's host. In this case, the source IP address is the address of the client device and it does not change during a handover (only the CNAPT Internet IP address changes).

Conditions (c), (d), (e), (f) and (g) refer to the fact that server and client services must be known in advance, from both SNAPT and CNAPT.

Conditions (f), (g) and (h) refer to the possibility that also the client could provide a set of services that can be used by the server for Publish/Subscribe communication models particularly relevant for peer-to-peer networks. Condition (h) explicitly deals with the situation where N identical CNAPTs, managing a client providing a client service on a certain port, are connected to a single SNAPT. A different Virtual IP address is used for each one of the identical CNAPTs. Clearly, the number of Virtual IP addresses must equal the number of the identical CNAPTs that can simultaneously access the SNAPT.

WiOptiMo does not require modifications of the client and server source code unless the client IP address, retrieved from the incoming packets, is used for critical internal activities (e.g., in case of an authentication scheme based on the client IP address).

5. PERFORMANCE EXPERIMENTS

In this section we report the results of three different sets of experiments. In the first set we show that the overhead introduced by WiOptiMo during the regular course of operations (i.e., not during a switch) is very low, negligible in practice. The second set of experiments show the robustness of the CNAPT's Check Activity mechanism detecting a broken connection. Finally, the third set of experiments regards the previously mentioned use at the application layer of ICMP packets to derive estimates of available throughput/delay. This last set of experiments reports in some sense a “negative” result, showing the inadequacy of such “tempting” approach.

5.1. Efficiency of WiOptiMo

We considered the simple but quite general scenario shown in Figure 8. The CNAPT is installed on the client mobile host and the SNAPT is installed on the same network as the server applications. The client runs on a laptop (Acer Aspire 2001WLCi) equipped with a LAN Ethernet adapter (Realtek RTL8139/810x, Family Fast Ethernet), a built-in WiFi card (Intel PRO/Wireless LAN 2100), and a GPRS card (Sierra Wireless Air Card AC750 Modem and GPRS Adapter). The server is located inside a corporate LAN. The SNAPT is installed on the same machine as the server in the case it is an FTP server and on a different machine in the case of a Microsoft Terminal Services Server and TightVNC Server. The client can connect through GPRS to a Telco bridge and then gets on the server via the Internet. The WiFi and LAN connections are managed by a 802.11b Wireless Access Point/DSL Router that is connected to the Internet. WiFi and LAN IPs are fixed. The testbed runs in 2 modalities: with a VPN active (realized with Checkpoint VPN-1/FW-1 NG FP3 HotFix 2 over Nokia IP120) and without a VPN.

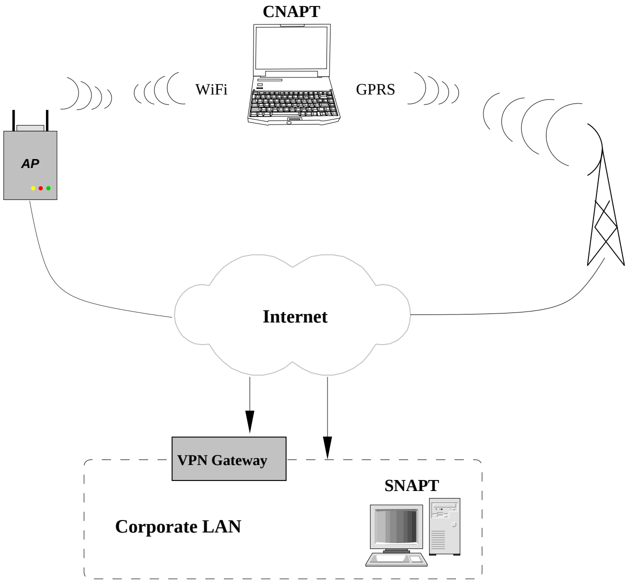
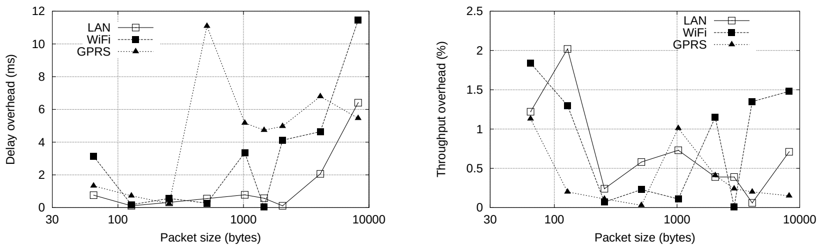


FIGURE 8
Configuration used for efficiency experiments.

Using this configuration we ran hundreds of experiments to evaluate the added *overhead*, in terms of delay and throughput, introduced by WiOptiMo. Referring to Figure 8, the application on the laptop uploads UDP/TCP packets of fixed size generated from the server inside the corporate LAN. All the experiments refer to the situation when no handover is necessary. In this case, WiOptiMo's impact on performance is caused by the presence of the two additional sockets which are used by CNAPT and SNAPT, which determine additional overhead because of opening, acknowledgments, and fragmentation operations.

Figure 9a reports averaged results for the delay overhead introduced by the use of WiOptiMo for different packet sizes and for the different radio technologies considered. Figure 9b shows the same results but in terms of throughput (reported as percentage of the regular data throughput). Concerning the delay, we can see that the large majority of the points in graph stay below the 8ms value. The behavior of the delay curves is a bit irregular but the range of variability is anyway short for all the different technologies. This tells that the relative impact of the presence of WiOptiMo is minimal, especially for larger packet sizes. The same reasonings apply to the throughput results: the behavior is slightly irregular but the negative impact in absolute terms is in practice almost negligible since it does not go over the 2% of the regular data throughput.

Contrary to solutions at network layer, WiOptiMo does not introduce any encapsulation or tunneling. Therefore, it adds a low computational overhead, as shown by the throughput results, but no networking overhead, such that the end-to-end delay is independent from the number of transmitted packets.



(a) (b)

FIGURE 9
(a) Delay overhead, and (b) throughput overhead introduced by WiOptiMo.

5.2. Efficacy of the Cross-layering Approach for WiOptiMo

The soundness and performance of the WiOptiMo system in the case of using only simple physical monitoring was assessed in previous work [6]. In [8] we tested, in a special testbed that did not include the use of WiOptiMo, the efficacy of a number of different variables at the *physical* and *MAC layers* to be used to recognize the need for a handover. The results of the test suggested that among these measures only the RSS can be robustly used to provide useful indications of a broken connection. Here we report some experimental results showing the effect of the combined use of *physical* and *application layer monitoring* in WiOptiMo to detect timely and robustly a broken connection and initiate a handover. The experiments were only aimed at gathering data from the monitored variables and not at dealing explicitly with the handovers at runtime. Collected data from these variables were given in input to WiOptiMo offline to tune Check Activity's ability to correctly detect the situations where a handover is appropriate.

We ran several tests using different hardware and considering a variety of user applications with different characteristics. We report the results from three scenarios. In all scenarios a mobile user is slowly moving in and out the coverage area of an AP connected to the Internet. Data shown in the graphs are sampled at regular intervals of 500 ms.

In the first scenario, the mobile host is equipped with a TrendNet TEW-401 PCMCIA card and uploads CBR data from the Internet at different data rates ranging from 2 Mbps, up to a maximum of 15 Mbps, with 802.11g cards. The movement was linear (in a corridor), moving forward and backward with respect to the AP. Results are reported in Figure 10. The upper graph shows the behavior in terms of correctly transmitted packets. The lower graph shows the evolution of the RSS (the irregular curve) and the behavior of the Check Activity in terms of triggering a handover initiation (the two-levels step-like curve). The purpose of the graph is to show that when both the RSS is below threshold and packet transmission dramatically falls down, that is, when the connection is likely broken, the Check Activity is able to spot the problem and invoke the handover. In the graph this is represented by the presence of a vertical bar. If the need for a handover persists over contiguous intervals, a horizontal line joins the vertical bars delimiting the interval. From Figure 10

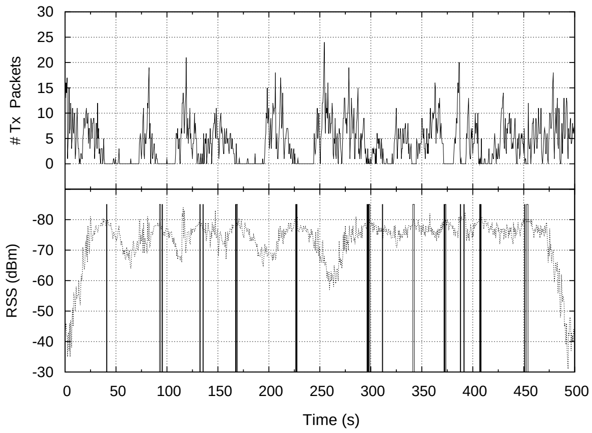


FIGURE 10

WiOptiMo's ability to robustly detect a broken link situation due to user mobility using cross-layer data. The meaning of the curves is explained in the text. Data refer to the case of a TrendNet PCMCIA card uploading CBR data to Internet through an AP.

we can see that WiOptiMo can detect broken connections in a way which is both timed and robust. Small fluctuations do not cheat the system, while all the situations of a clearly broken link are promptly detected.

A similar behavior is shown also in Figures 11 and 12, which refer respectively to the same TrendNet card but this time used for a Skype connection, and an Acer card Intel Pro/Wireless 2200 b/g used for Internet browsing. The application traffic shows substantial differences among the different scenarios, as well as the characteristics of the different cards are rather different from each other. Nevertheless, WiOptiMo's behavior is consistently robust and effective across all the different tested situations.

5.3. On the Use of Probing Packets for Throughput/Delay Estimates

As discussed in Section 3.4, WiOptiMo makes use of a cross-layer approach to derive timed and robust information about the status of a connection. The experimental results of the previous section show the effectiveness of the approach for what concerns the handover initiation phase. On the other hand, in order to take an optimized decision at network selection time, it is needless to say that the system would profit from the availability of information about the effective throughput that would be available from each candidate network. As pointed out in Subsection 2.4, future 802.11 standards will hopefully provide an efficient solution to this issue. In the meantime, other possible solutions should be investigated. At this aim, we envisaged the use of ICMP packets not only to check the effective status of the connection, but also to derive information on the quality of the connection in terms of available throughput and/or end-to-end delay (or, equivalently, in terms of current traffic load). In cases of excessive congestion, the connection might seem on the edge of an interruption if only physical layer measures are used. Therefore, application layer measures can be used in cross-validation with physical ayer measures to help to discrim-

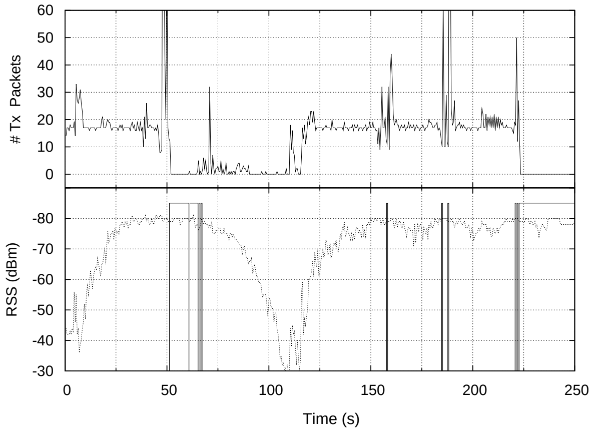


FIGURE 11
Check Activity's ability to robustly detect a broken link situation due to user mobility using cross-layer data. The meaning of the curves is explained in the text. Data refer to the case of a TrendNet PCMCIA card connected through an AP to a Skype server on the Internet. The user experiences a large reconnection delay after connection breaking happening at time 50s. In fact, since WiOptiMo was not used during the experiment, the user was forced to re-issue a request for connection after 15s of interruption, and then Skype took about 30 more seconds to re-establish the communication.

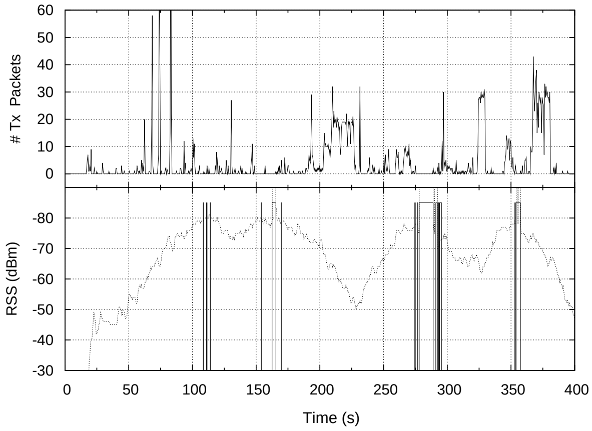


FIGURE 12
Check Activity's ability to robustly detect a broken link situation due to user mobility using cross-layer data. The meaning of the curves is explained in the text. Data refer to the case of an Acer card used for browsing the Internet through an AP.

inate between congestion and actual broken link situations. As pointed out in Section 2.3, the availability of MAC layers variables such as the NAV would simplify this task, but unfortunately most the NICs on the market do not allow access to these variables.

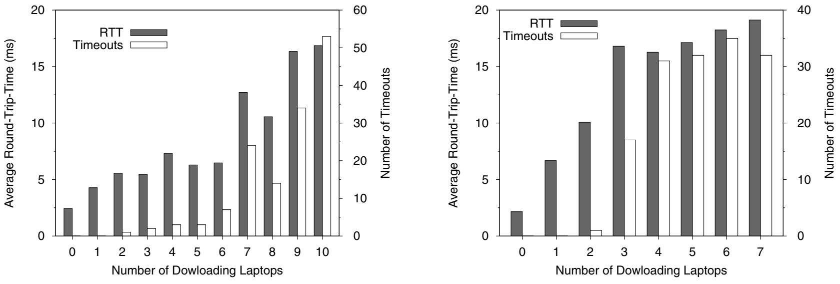
All the experiments reported in this section were conducted in conditions of strong and stable RSS to check if under favorable physical conditions it is possible to observe significant variations at the application layer in relationship to load variations. More specifically, we report results concerning the use of streams of ICMP packets as *probing packets* to actively derive load estimates at the application layer as a function of variations in the load offered to the wireless environment. Probing techniques have been widely used in the wired Internet for similar purposes. However, published results seem to indicate that in order to obtain really reliable estimates, rather intrusive and repeated probing actions are needed in addition to careful parameter setting (e.g., see [51]). On the other hand, the same probing techniques (and/or they adaptation) once used in wireless environments seem to loose most of their efficacy and reliability [52, 53]. We wanted to check if they could be used to give weak but still helpful indications without incurring in excessive overhead.

In the ran experiments, ICMP messages are proactively sent to the AP in order to evaluate the connection status in terms of: (i) *throughput*, on the basis of the observed RTTs, and (ii) *channel congestion* according to the number of experienced timeouts. We considered the following four traffic-intensive scenarios experiments where an increasing load is offered to the WLAN: (1) an increasing number of laptops (from 1 to 10) continuously download data from one PC connected via LAN to the AP, while in the meantime another laptop sends ICMP messages to the AP; (2) an increasing number of laptops (from 1 to 7) continuously download data from each other passing through the AP while another laptop sends ICMP messages to the AP (in this way the network load was doubled with respect to the previous case); (3) an increasing number of laptops (from 1 to 10) continuously download data from the Internet through the AP while another laptop sends ICMP messages to the AP; (4) 11 laptops with an increasing number of downloading processes (from 1 to 5 on each laptop) download data from the Internet through the AP while another laptop sends ICMP messages to the AP. In each scenario a continuous train of 1000 ICMP messages (size of 32 bytes) was generated during the experiment. The size of each single download was of 1.5 Gbytes. The WLAN is based on 802.11g devices.

As shown in Figure 13, in the first two scenarios the average RTT and the number of experienced ICMP timeouts increased each time that a new laptop joined the network with a new download. Moreover, the number of timeouts rose up very fast when the number of laptops in download grew from 7 to 10 in the first scenario, and from 3 to 7 in the second scenario.

In these scenarios, the RSS was always above the critical threshold. Therefore, no handover indications could have been derived from this physical measure. However, from Figure 13 it is clear that the throughput was significantly decreasing, and the channel congestion increasing, with the increase of the active users in the WLAN. Therefore, the combined analysis of the RSS and RTT values seems to be necessary to become aware of a situation of pure traffic congestion.

On the contrary, in the third scenario (Figure 14a) both the average RTT and the number of ICMP timeouts remain low almost independently from the number of laptops in download. This was probably due to the available bandwidth on the Internet side, that was lower than that available in the WLAN. In



(a)

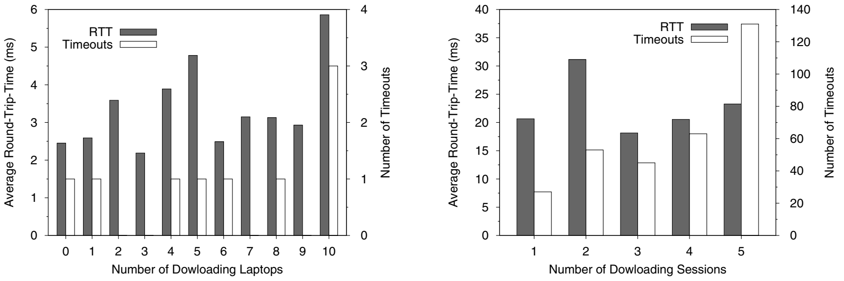
(b)

FIGURE 13

Experimental results for the first (a) and second (b) load estimation scenarios. In the first scenario an increasing number of laptops download data from one PC connected to an AP via LAN. In the second scenario, the laptops download data from each other passing through the AP. In both cases an additional laptop sends ICMP messages to the AP.

the last scenario we overloaded the WLAN with a large number of downloads (11 up to 53) from the Internet. As shown in Figure 14b, the average RTT remained more or less the same but the number of timeouts rose up very fast reaching in the worst case more than 10% of the total number of ICMP messages sent. Therefore, in these last two scenarios, the usefulness and reliability of the approach using ICMP packets seems to be much less evident.

In more general terms, it is necessary to point out that even if some tests showed a clear increase in the average RTT as well as in the number of ICMP timeouts in relationship to an increase in the connection traffic, these are values obtained averaging 1000 ICMPs. Single measurements, and/or averages over few packets, show large variances and provide very contradictory results. It is clear that the system cannot rely on generating such a large additional overhead without causing undesired throughput reduction and latency increase. Furthermore, considering that WiOptiMo has been designed to be used mainly



(a)

(b)

FIGURE 14

Experimental results for the third (a) and fourth (b) load estimation scenarios. In the third scenario an increasing number of laptops download data from the Internet passing through an AP. In the fourth scenario, data download from the Internet are performed by an increasing number of sessions running on a fixed number of laptops (11 in the experiments). In both cases an additional laptop sends ICMP messages to the AP.

for Internet and not for intranet, we can conclude that it is not feasible to use probing packets for the sole purpose of carrying out load/throughput estimate. Accordingly we ruled this alternative out of future WiOptiMo's improvements.

6. CONCLUSIONS AND FUTURE WORK

Widespread availability of wireless services and their use by mobile devices is becoming a reality and is expected to play an ever increasing role in our daily life. The demand for efficient and flexible solutions for micro- and macro-mobility management is increasing accordingly. When changing the network connection point and possibly also the access technology, mobile users must be provided with a totally transparent management of mobility, continuity of service respectful of the expected QoS, and minimization of costs and used resources. Most of the current solutions based on network layer approaches like Mobile IP fail to reach these objectives. We discussed the reasons behind this fact, and pointed out why an application layer solution like our WiOptiMo can overcome most of the limitations presented by Mobile IP and related approaches and can provide efficient and effective mobility management in heterogeneous, multi-technology, networks.

Our general target was to design an integrated system for all types of mobility, that could provide optimized performance, show a good degree of adaptivity and robustness, and be immediately and widely deployed on the Internet. At this aim we presented an architecture based on the use of so-called CNAPT and SNAPT applications, that collectively act as a middleware interfacing the communications between the client and the server applications hiding the mobility to them. We extensively described and discussed WiOptiMo's characteristics, considering, for sake of clarity, the special case of a client-server architecture. However, the system can be used with only relatively minor adaptations also for other general communication paradigms like the peer-to-peer one. To provide timed and optimized handover choices, WiOptiMo makes use of both active and passive cross-layer monitoring, as well as of autonomic components for self-tuning and self-configuring purposes. In order to allow immediate and widespread deployment, we strictly relied on the use of standard protocols and driver implementations. Our a software implementation is easily portable across the different operating systems and at the same time is very parsimonious in terms of CPU and battery resources, such as it can be used on a wide range of portable devices available on the market.

We reported few experimental results showing the efficiency of WiOptiMo in terms of very low overhead, and its efficacy in terms of the ability to provide robust, timed and correct handovers. We also reported about the possible use of probing packets to derive estimates of the load status of an access point, to be used to take performance-optimized handover decisions.

Future work includes the investigation of cross-layer measures of the connection quality alternative to the signal strength, which suffers of excessive variability. Moreover, we plan to investigate more in depth the issue of the feasibility of deriving sound load estimates without generating unacceptable overhead. Also the improvement of the autonomic components, as well as of the user interface are an important part of our current/future plans. Finally, we are working to the refinement of the software implementation in order to improve its efficiency and to favor full portability across different platforms.

References

[1] D. Saha and A. Mukherjee. Pervasive computing: A paradigm for the 21st century. *IEEE Computer*, 36(3):25–31, March 2003.

[2] C. Perkins (Editor). IP mobility support. RFC 2002, Network Working Group, 1996.

[3] C. Perkins (Editor). IP mobility support for IPv4, revised. RFC 3344bis-02, IETF – Mobility for IP4, October 2005.

[4] D.B. Johnson, C. Perkins, and J. Arkko. Mobility support in IPv6. RFC 3775, IETF, June 2004.

[5] H. Schulzrinne and E. Wedlund. Application-layer mobility using SIP. *ACM SIGMOBILE Mobile Computing and Communications Review*, 4(3):47–57, 2000.

[6] S. Giordano, D. Lenzarini, A. Puiatti, and S. Vanini. WiSwitch: seamleass handover between multi-provider networks. In *Proceedings of the 2nd Annual Conference on Wireless On demand Network Systems and Services (WONS)*, 2005.

[7] S. Giordano, D. Lenzarini, M. Schiavoni, and S. Vanini. Virtual web channel: Flow aggregation for enhanced ubiquitous web access. In *Proceedings of IEEE WirelessCom*, 2005.

[8] S. Giordano, M. Kulig, D. Lenzarini, A. Puiatti, F. Schwitter, and S. Vanini. WiOptiMo: Optimised seamless handover. In *Proceedings of IEEE WPMC*, 2005.

[9] G.A. Di Caro, S. Giordano, M. Kulig, D. Lenzarini, A. Puiatti, and F. Schwitter. A cross-layering and autonomic approach to optimized seamless handover. In *Proceedings of the 3d Annual Conference on Wireless On demand Network Systems and Services (WONS)*, 2006.

[10] C. Perkins and D.B. Johnson. Route optimization in Mobile IP. Internet draft, IETF, November 1997.

[11] C. Perkins. IP encapsulation within IP. RFC 2003, IETF, October 1996.

[12] F.C. Choong. TCP Performance in mobile-IP. Technical report, National University of Singapore, Department of Electrical Engineering, 2003.

[13] A. T. Campbell and J. Gomez. IP micro-mobility protocols. *ACM SIGMOBILE Mobile Computing and Communications Review*, 4(4), October 2001.

[14] P. Vidales. Seamless mobility in 4G systems. Technical Report UCAM-CL-TR-656, University of Cambridge, Computer Laboratory, Cambridge, UK, November 2005.

[15] B. E. Henty and T. S. Rappaport. Throughput measurements and empirical prediction models for IEEE 802.11b wireless LAN installations. Technical Report MPRG 01-08, Virginia Polytechnic Institute, Department of Electrical and Computer Engineering, 2001.

[16] T. Bates and Y. Rekhter. Scalable support for multi-homed multi-provider connectivity. RFC 2260, 1998.

[17] U. Varshney and R. Jain. Issues in emerging 4G wireless networks. *IEEE Computer*, 34(6):94–96, 2001.

[18] F. Adrangi and H. Levkowetz. Problem statement: mobile IPv4 traversal of VPN gateways. Internet draft, IETF, 2002.

[19] S. Vaarala and E. Klovning. Mobile IPv4 traversal across IPsec-based VPN gateways. Internet draft, IETF, November 2005.

[20] R. Koodli and C. Perkins. Fast handovers and context transfers in mobile networks. *ACM SIGCOMM Computer Communication Review*, 31(5):37–47, 2001.

[21] P. Reinbold and O. Bonaventure. IP micro-mobility protocols. *IEEE Communications Surveys and Tutorials*, 5(1), 2003.

[22] Swisscom Mobile AG. Method and system for mobile IP nodes in heterogeneous networks. *International Application N. WO 02/103978 A2, published under the Patent Cooperation Treaty (PCT)*, 2002.

[23] Nortel Networks Limited. Method and system for switching between two network access technologies without interrupting active network applications. *EU Patent Application N. EP 1 089 495 A2*, 2001.

[24] Nokia Mobile Phones LTD. Method for coupling a wireless terminal to a data transmission network and a wireless terminal. *EU Patent Application N. EP 0 998 094 A2*, 2002.

[25] A. Dutta, S. Madhani, W. Chen, O. Altintas, and H. Schulzrinne. Fast-handoff schemes for application layer mobility management. In *Proceedings of the 16th Annual IEEE International Symposium on Personal Indoor and Mobile Radio Communications (PIMRC)*, Spain, 2004.

[26] A. Snoeren and H. Balakrishnan. An end-to-end approach to host mobility. In *Proceedings of the 6th ACM MOBICOM*, Boston, MA, August 2000.

[27] I. Stoica, D. Adkins, S. Zhuang, S. Shenker, and S. Surana. Internet indirection infrastructure. In *Proceedings of the ACM SIGCOMM Conference on Applications, Technologies, Architectures, and Protocols for Computer Communication*, pages 73–86, 2002.

[28] S. Zhuang, K. Lai, I. Stoica, R.H. Katz, and S. Shenker. Host mobility using an internet indirection infrastructure. *Wireless Networks*, 11(6):741–756, 2005.

[29] K. Murray and D. Pesch. State of the art: Admission control and mobility management in heterogeneous wireless networks. Deliverable 3.1 D1.1 of the *M-Zones* Project, May 2003.

[30] A. Mishra, M. Shin, and W. Arbaugh. An empirical analysis of the IEEE 802.11 MAC layer handoff process. *ACM SIGCOMM Computer Communication Review*, 33(2):93–102, 2003.

[31] N. Tripathi, J. Reed, and H. Vanlandingham. Handoff in cellular systems. *IEEE Personal Communications Magazine*, December 1998.

[32] M.-H. Ye, Y. Liu, and H.M. Zhang. The mobile IP handoff between hybrid networks. In *Proceedings of the 12th Annual IEEE Int. Symposium on Personal Indoor and Mobile Radio Communications*, 2002.

[33] G. Gaertner and V. Cahill. Understanding link quality in 802.11 mobile ad hoc networks. *IEEE Internet Computing*, 8(1):55–60, 2004.

[34] D. Lymberopoulos, Q. Lindsey, and A. Savvides. An empirical analysis of radio signal strength variability in IEEE 802.15.4 networks using monopole antennas. Technical Report 050501, ENALAB, Yale University, New Haven, CT, USA, 2005.

[35] C. Guo, Z. Guo, Q. Zhang, and W. Zhu. A seamless and proactive end-to-end mobility solution for roaming across heterogeneous wireless networks. *Journal of Selected Areas in Communications* Special Issue on Advanced Mobility Management And QoS Protocols for Next-generation Wireless Internet, 22(5), June 2004.

[36] W. Zhang, J. Jaehnert, and K. Dolzer. Design and evaluation of a handover decision strategy for 4th generation mobile networks. In *Proceedings of the 57th Semiannual Vehicular Tech. Conf. (VTC)*, 2003.

[37] S. Pack, H. Jung, T. Kwon, and Y. Choi. SNC: A selective neighbor caching scheme for fast handoff in IEEE 802.11 wireless networks. *ACM SIGMOBILE Mobile Computing and Communications Review*, June, 2005.

[38] A. Majlesi and B.H. Khalaj. An adaptive fuzzy logic based handoff algorithm for hybrid networks. In *Proceedings of the International Conference on Signal Processing*, August 2002.

[39] D. Gu and J. Zhang. A new measurement-based admission control method for IEEE 802.11 wireless local area networks. In *Proceedings of the IEEE Int. Symposium on Personal, Indoor and Mobile Radio Communications (PIMRC)*, volume 3, pages 2009–2013, September 2003.

[40] G. Bianchi and I. Tinnirello. Kalman filter estimation of the number of competing terminals in an ieee 802.11 network. In *Proceedings of INFOCOM*, April 2003.

[41] M. Ylianttila, M. Pande, J. Mäkelä, and P. Mähönen. Optimization scheme for mobile users performing vertical handoffs between IEEE 802.11 and GPRS/EDGE networks. In *Proceedings of Global Telecommunications Conference 2001*, 2001.

[42] D. Lenzarini. Method and system for seamless handover of mobile devices in heterogeneous networks, *Patents N. PCT/EP2005/050599 and PCT/EP2004/050111, International Publication Numbers: WO2005/076651 and WO2005/076649, Forward Information Technology (Switzerland)*, Publication date: August 18, 2005. http://ep.espacenet.com.

[43] M. Conti, G. Maselli, G. Turi, and S. Giordano. Cross-layering in mobile ad hoc network design. *IEEE Computer*, 37(2):48–51, February 2004.

[44] J. Kephart and D. Chess. The vision of autonomic computing. *IEEE Computer magazine*, 36(1):41–50, January 2003.

[45] G.J. Lee. Automatic service selection in dynamic wireless networks. Master's thesis, Computer Science and Artificial Intelligence Laboratory, Massachusetts Institute of Technology, Boston, USA, 2003.

[46] G.J. Lee, P. Faratin, S. Bauer, and J. Wroclawski. A user-guided cognitive agent for network service selection in pervasive computing environments. In *Proceedings of 2nd IEEE International Conference on Pervasive Computing and Communications (PerCom)*, 2004.

[47] M. Kulig and F. Schwitter. Monitoraggio dei parametri per ottimizzare una soluzione di always-on (in Italian). Semester Project Report, University of Applied Sciences (SUPSI), Manno (Switzerland), 2005.

[48] H. Velayos and G. Karlsson. Techniques to reduce IEEE 802.11b MAC layer handover time. Technical Report TRITA-IMIT-LCN R 03:02, Kungliga Tekniska Hskolan (KTH), Stockholm, Sweden, April 2003.

[49] K. Kawabata, T. Nakamura, and E. Fukuda. Estimating velocity using diversity reception. In *Proceedings of the IEEE Vehicular Technology Conference*, pages 371–373, March 1994.

[50] M.D. Austin and G.L. Stuber. Velocity adaptive handoff algorithm for microcellular systems. *IEEE Transactions on Vehicular Technology*, 43:549–561, August 1994.

[51] N. Hu and P. Steenkiste. Evaluation and characterization of available bandwidth probing techniques. *IEEE Journal on Selected Areas on Communications* Special Issue on Internet and WWW Measurement, Mapping, and Modeling, 21(6):879–894, 2003.

[52] A. Johnsson. *Bandwidth Measurements in Wired and Wireless Networks*. PhD thesis, Mälardalen Research and Technology Centre (MRTC), Mälardalen University, Mälardalen, Sweden, April 2005.

[53] Z. Yang, C. Chereddi, and H. Luo. Bandwidth measurement in wireless mesh networks. Course Project Reports, Wireless Netwirking Group, University of Illinois, Urbana-Champaign, 2004.