Mobility Management for VoIP on Heterogeneous Networks: Evaluation of Adaptive Schemes

Massimo Bernaschi, Filippo Cacace, Giulio Iannello, *Member, IEEE,* and Massimo Vellucci

***Abstract*— The introduction of the IP multimedia subsystem on 3G cellular networks and the integration with other widely deployed wireless networks based on the IEEE 802.11 protocol family require support for both mobility and quality of service. When mobile systems move across heterogeneous networks, ongoing real-time sessions are affected not only by handoff delay but also by different packet delay and bit-rate. In this paper we propose a cross-layer mechanism that takes into account mobility at different layers of the network stack in order to yield better quality for VoIP, videoconferencing and other real-time applications. We describe our cross-layer architecture, adaptation techniques, a prototype implementation and experimental results.**

***Index Terms*— Mobility, Adaptivity, Voice over IP.**

I. Introduction

THE Third Generation Partnership Project (3GPP) is currently developing standards in order to support seamless mobility among integrated 3G and 4G wireless architectures, Wireless Local Networks (WLANs) based on the protocols of the IEEE 802.11 family and other networks [1]. In the recent past, the 3GPP has defined also the IP Multimedia Subsystem (IMS) to support IP-based services such as streaming, voice over IP (VoIP) and videoconferencing [2]. In particular, the IMS provides real-time services among hosts connected both to cellular networks and the Internet. In such a way, mobile systems (like cellular phones, PDAs and notebooks) should be able to offer real-time services across different access technologies in a seamless way. However, the support to mobility raises new issues related not only to handoff management, such as low disruption time [3], but also to the quality of the service (QoS).

Wireless technologies provide network access with substantial differences with respect to bit-rate, delay, packet-loss, *etc.*. As a consequence, it is crucial that real-time applications include the capability to adapt to such variations in order to fulfill QoS requirements. An end-to-end adaptive mechanism can offer a good contribution toward supporting real-time QoS on heterogeneous networks that do not have a uniform QoS enforcement policy and, in the medium term, it might be the only viable option. We show how this approach can be implemented by introducing mobility awareness and management at different layers of the network stack.

The results presented in this paper extend our previous work on cross-layer mobility support. Initially, we developed and validated a cross-layer approach to improve the performance of heterogeneous handoffs through link-layer triggering [4], a solution already in use for horizontal handoffs [5]. We soon realized that this approach could be extended to support content adaptation at session layer. In particular we showed in [6] that it allows to achieve good performance with little overhead for mobile streaming applications. To move a step forward, we considered real-time applications like VoIP, videoconferencing and network games, where delay constraints are important. The present paper extends our cross-layer approach to VoIP applications.

The contribution of this paper is to describe this approach and its implementation in a working prototype. Experimental results about its performance at network, session and application layers in a 3G/WLAN scenario are presented. Our main conclusion is that with the proposed approach the expected performance of a VoIP application in mobility situation is very close to the one reported in literature for VoIP applications on wired geographical networks [7]. We also show that this result can be obtained with no modification of existing protocols.

In Section II we describe the internetworking and mobility scenario in which our approach is applicable and the requirements of real-time mobile applications like VoIP. In Section III we present our cross-layer approach to improve mobility support at network layer. The mobility protocol we employ is Mobile IPv6. The Application Programming Interface of the mobility module, which allows to provide mobility related services to upper layers of the network stack, is also presented. We then describe how to introduce adaptation at session and application layers (Section IV and V) and experimental results obtained with this approach. Section VI concludes the paper.

II. Background

A. Scenario overview

The chance of using different wireless access networks during a VoIP session is a promising opportunity in many different scenarios. For vehicular mobility it makes it possible to use high speed wireless connections when they are available maintaining the possibility to resort to ubiquitous, albeit slower, cellular networks in areas that are outside of WLANs coverage. Also in smaller areas where the distribution of access points cannot achieve a complete coverage, the seamless migration of a real-time session to a more pervasive 3G radio access enhances the service availability that the user perceives and can promote the service usage.

The IMS is the key element in the 3G architecture that makes it possible to provide ubiquitous cellular access to all

M. Bernaschi is with the Istituto per le Applicazioni del Calcolo "Mauro Picone" part of the Italian National Research Council. F. Cacace, G. Iannello, and M. Vellucci are with Università Campus Bio-Medico of Rome.

Digital Object Indentifier 10.1109/TMC.2007.1003

Internet services [8]. More specifically, IMS is the component that must provide support for multimedia services (*e.g.*, voice and/or video) based on packet switching with quality of service and authentication, authorization, and accounting (AAA) provisioning. To achieve its goals, IMS uses open Internet protocols, like SIP [9][1] as session control protocol and RTP/RTCP to transport real time media, such as video and audio. IMS design is based on IPv6, the new version of the IP protocol, even though early implementations may still use the current IPv4 version.

Inter-networking issues between IMS entities and generic Internet hosts, as well as roaming between IMS and non-IMS networks are still a subject of study. According to [1], [10] five interconnection levels can be considered: *i)* common billing and customer care; *ii)* 3G system-based access control and charging; *iii)* access to 3G system IMS-based services; *iv)* service continuity; *v)* seamless services. These interconnection levels refer to a *tightly-coupled* approach, where WLAN users are within the 3G domain and 3G providers operate both networks, but they are relevant also in a future loosely-coupled scenario, where mobile users access the network through a public WLAN.

3GPP included the first three levels in Release 6, whereas the last two will be developed in future releases. In particular, level (iv) provides the ability to maintain an active service session when the user moves between two access networks (from a WLAN to a 3G UTRAN and vice-versa), although the handoff process may be perceptible to the user (due to data loss or delays). The fifth level provides seamless continuity, with a service interruption comparable to that perceived in *intra*-3G handoffs.

Our work is particularly relevant to the issues raised by these two last levels. We aim at allowing seamless service continuity across distinct wireless access networks by means of a cross-layer, end-to-end, approach compatible with the IMS framework. The requirements we want to meet are: *(i)* adequate performance; *(ii)* no change to existing network infrastructures; *(iii)* no modification to existing protocols.

B. Requirements for VoIP services

Mobility support of VoIP applications has different requirements with respect to Web or streaming applications, as they are related to specific features of VoIP services:

- VoIP applications are *delay sensitive*. An acceptable VoIP call must have a total end-to-mouth delay (E2M) not exceeding 150-200 ms [7]. However, wireless access networks exhibit significant variations in packet delay. For a WLAN a RTT in the order of 20-50 ms can be expected, but present 3G cellular networks have values up to 150-250 ms. Enforcing a limit on the E2M delay is not easy when passing from one access network to another.
- VoIP applications must synchronize the speech playout. With a limited amount of buffering, due to E2M delay limitations, a change in the network delay might cause a loss of synchronization, thus deteriorating the quality of a VoIP call also in the absence of packet loss. This is the case when packets arrive too late to be played back, or when they arrive too soon and the application cannot perform packet reordering.
- When moving to a network that offers a lower bit-rate, a VoIP application may experience congestion situations, with packet loss and service degradation. For example, a 64 kbps codec for human voice can be used on a WLAN, but it is beyond the current capabilities of a UMTS network whose maximum bit-rate in up-link is just 64 kbps. It is a task of the application to detect the available bit-rate and to adapt its behavior when changing access network. Even if this specific problem will be probably solved in forthcoming generations of HSDPA/HSUPA access networks, future videoconferencing applications might experience the same problem in mobility situations, thus we argue that adaptation is an important issue for this class of applications.
- Service disruptions due to handoff execution must be contained under 0.1-0.2 s for a seamless service. Packet losses corresponding to this interval are acceptable, but handoff frequency should be low: in particular, the ping-pong effect between two access networks must be avoided.

We claim that these requirements cannot be satisfactorily met by acting on a single layer of the network stack. We therefore propose to address these issues in an integrated, cross-layer framework within which we deal with:

- delay and synchronization issues at the application layer;
- bandwidth requirements at the session layer;
- vertical handoff optimization at the network layer.

In the subsequent sections we propose our solution for each layer, along with experimental results, background and a comparison with related approaches.

III. Mobility Management for VoIP Services

A. Mobility on heterogeneous networks

Mobility in wireless networks can be managed at different layers:

- *link* layer mobility is specific for each wireless access network. For WLANs it allows roaming between access points belonging to the same subnet, whereas in the UMTS environment it supports handoffs from one cell to another.
- *network* layer mobility has been proposed as an extension of the IP protocol [11]. Mobility at network layer provides a uniform mobility management that makes handoffs and network changes invisible to upper layers.
- *transport* layer mobility uses extension to transport protocols, like in the case of mobile SCTP [12]. The drawback of this approach is that mobility is restricted to applications using a specific transport protocol. The performance is also usually worse than in the network case.
- *application* layer mobility is application specific. For real-time services, an interesting opportunity is to introduce mobility management in the SIP protocol [13], [14], so

[1]The SIP version used by 3GPP in the IMS design has special extensions with respect to the "basic" SIP as defined by the IETF.

that it becomes part of the signaling protocol, with no extra cost. Indeed, SIP is capable of handling terminal, session, personal and service mobility. The main drawbacks are performance and lack of mobility support for non SIP-based applications. The analytical study reported in [14] shows that, when the wireless link is congested or subject to errors, the disruption time in SIP horizontal handoffs increases much more with respect the Mobile IPv6 case. More recent results [15] for vertical handoffs provide a value of 1.5-2 s for SIP WLAN-to-UMTS handoff delay when the Frame Error Rate (FER) is between 0.01 and 0.1 on the UMTS link. Most of this delay is due to the transmission of handoff signaling messages on the error-prone UMTS link.

Network layer mobility offers two main advantages, namely: *(i)* better performance with respect to solutions implemented at upper layers of the protocol stack, and *(ii)* a common mobility management that is transparent to any application or higher level protocol. Network applications always use just one source address, the *Home Address*, and are unaware of access network changes. For this reason, in the present paper we follow the network layer approach, through the use of Mobile IPv6, which is designed to support mobile nodes in IPv6 networks. When a mobile node (MN) moves through other networks, the home agent keeps track of the current binding of the mobile node. When a handoff takes place, the mobile node sends messages, known as *Binding Updates* (BU), both to the home agent and to any node with which it is communicating, usually indicated as *Correspondent Nodes* (CN). The above mechanism is called *horizontal handoff* if the migration is between homogeneous networks. The same procedure can also be used for mobility through heterogeneous networks for mobile nodes equipped with more than one network interface. In this case the change of active interface is called *vertical handoff*, since it takes place in presence of a hierarchy of overlaid networks with different features (*e.g.*, bandwidth, power consumption, cost, *etc.*)

B. Mobility Manager

The performance of Mobile IPv6 with respect to horizontal handoffs has been studied thoroughly in recent years [16], [17]. Even though vertical handoffs can be implemented through Mobile IP, their peculiarities require a specific analysis [18], [19]. First of all, horizontal handoffs are typically required when an access router becomes unavailable due to mobile hosts movement. On the contrary, being overlaid is a common situation for heterogeneous networks. An implication of the above observation is that the handoff can be initiated for convenience, rather than connectivity reasons. It is often possible to have loss-less handoffs by performing all the configuration and signaling steps on the new network before actually leaving the old one. This is called *make before break* handoff, as opposed to a *break before make*, that happens with connection disruption, usually because the active access router becomes unreachable before the handoff execution.

The efficiency of vertical handoffs depends mostly on movement detection, and precisely on the capability of detecting that the current access network is going to become unusable before it actually does. The ordinary IPv6 method to discover router reachability, based on the reception of Router Advertisements messages, is not well suited to this task. A software module that monitors the link status can react more quickly.

Figure 1 describes the cross-layer approach proposed to support handoff decisions. The Mobility Manager (MM) is a user-level application in charge of: *i)* monitoring connection parameters through appropriate system calls; *ii)* performing handoff decisions with appropriate timings using MobileIPv6 primitive operations; *iii)* dispatching notifications about connectivity changes and network parameters to any application residing on the mobile system that notifies its interest in these events through an Application Programming Interface (API). The MM scope is limited to the network layer. The application/session adaptation is performed separately by each adaptive application that: *i)* receives notifications from the MM; *ii)* decides how to adapt.

The use of link layer triggers to improve handoff management at network level is the approach used in the new IEEE 802.21 standard, that aims at improving handoff and interoperability among heterogeneous networks (both 802 and non-802 network types). The 802.21 standard is not finalized yet, but our MM basically shares the same approach even though it is implemented in user-space rather than inside the network stack. Our work provides experimental evidence that this approach is indeed feasible and efficient. Moreover, the MM module has one more feature, namely the support for adaptive applications.

The MM can be used to provide mobility support to any kind of application, for instance streaming applications [6]. For legacy applications, the MM delivers event notifications to an Application Mobility Interface (AMI), in charge of performing adaptation at session layer and/or issuing adaptation commands for the legacy application.

The description of the MM module is beyond the scope of the present paper (see [19] for details). All experimental results obtained in our testbed when using the MM show considerable performance improvements for handoffs from higher priority interfaces with packet loss (the typical situation is an up handoff from a WLAN to a UMTS access network). The overall handoff delay at network layer is reduced from 2.0 s to about 0.2 s, that is, to (about) 10% of the original time. Handoff decision algorithms in the MM can be used to implement network selection algorithms driven by user preferences [20] and to reduce undesirable ping-pong effects.

The API used by the adaptive VoIP application to receive notifications about connectivity events contain the primitives listed in Table I (we omit implementation details):

- *handoff_notification*. The application requires to receive a notification, through the *pHandoff_f* callback function, whenever a vertical/horizontal handoff is performed.
- *prehandoff_notification*. The application requires to receive a notification, through the *pHandoff_f* callback function, when the MM decides to perform a handoff, but *before* the handoff is actually executed. The MM waits for the completion of *pHandoff_f* (up to a pre-determined time interval) before performing the handoff.

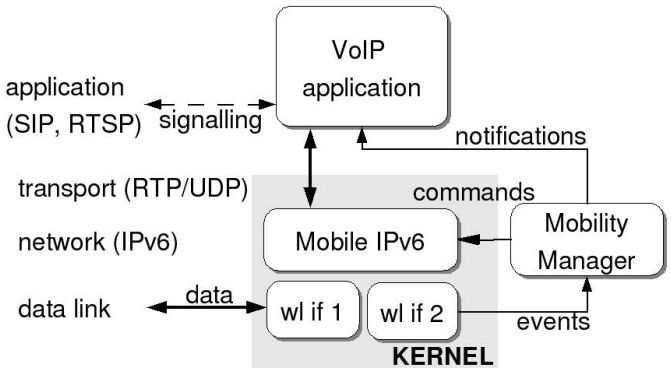

Fig. 1. Cross-layer mobility support

TABLE I
API FOR THE MOBILITY MANAGER

Function name	Input	Output
handoff_notification	pHandoff_f	exit_code
prehandoff_notification	pHandoff_f	exit_code
warning_notification	pWarning_f	exit_code
get_interfaces	void	If_info **info_l
get_curr_interface	void	If_info *info
void (*pHandoff_f) void (*pWarning_f)	If_info *Old_If, If_info *New_If	void

- *warning_notification*. The application requires to receive a notification, through the *pWarning_f* callback function, whenever there is a meaningful change of the active interface connectivity parameters. This primitive function is used to report changes that are not under the control of the MM, like, for example, network-initiated handoffs, the switching from a UMTS to a GPRS connection, or a bit-rate change from 54 to 11 Mbps for a wireless LAN interface.
- *get_interfaces*, *get_curr_interface*. These functions are used to retrieve network parameters related to (resp.) all the interfaces and the currently active interface through the *If_info* data structure.
- call back functions *pHandoff_f*, *pWarning_f*. These functions are defined within the application and are invoked by the MM when the corresponding events occur. They receive as input information that describe the new access network.

The network parameters related to the interfaces (such as bit-rate, delay, jitter, *etc.*), contained in the *If_info* data structure, are estimated by the MM from the link technology, QoS agreements or active probes.

Notice that the MM has complete control over handoff decision and execution, since the hardware and the driver of the network interfaces are used to provide only values about link presence and quality. The CPU overhead and power consumption due to the activity of the MM is very limited, since the application does not need to communicate through the wireless interfaces and its task is limited to the periodic polling of the registries of the drivers. We studied this overhead with the FunctionCheck profiling utility for multithreaded applications on a Pentium 4 CPU at 1.7 GHz with a Linux 2.4.24 kernel. The MM monitored three interfaces (Ethernet 10 times per second, WiFi and UMTS 100 times per second) with fast variations of the wireless signal and frequent plugging and unplugging of the Ethernet cable, that caused an handoff every 2 seconds. The average ratio between the CPU time and the execution time was 0.6%. Of course, the overhead would be larger if complex handoff decision algorithms were employed: the algorithm we used in the testbed was a prioritized threshold comparison. Note, that the MM can lead to power saving if it is used to turn off non-active wireless interfaces.

C. Experimental results

Our first set of tests focused on handoff performance at network layer. We performed both *up* handoffs (from a faster, but more limited in scope, network to a slower one) and *down* handoffs, using the MM module and comparing the performance with the basic network layer triggering mechanism of Mobile IPv6 ("L3 triggering"). In Table II we only report results of up handoffs, since at network layer down handoffs are usually lossless. The handoff delay is measured as the time interval between the arrival of the first packet from the new network and the arrival of the last packet from the old network. We performed six measurements for each case. The experimental setup of our testbed included two laptops with Linux 2.4 and the MIPL 1.0 Mobile IPv6 implementation. Since the UMTS network does not natively support IPv6, we have resorted to a IPv6-inIPv4 tunnel from the MN to an IPv6 Access Router at the edge of our testbed. The maximum Router Advertisement interval (that influences the L3 triggering performance) was set to 100 ms on LANs and WLANs and 200 ms on the UMTS network.[2] We used these values, as well as standard settings for IPv6 timers, since they are representative of most typical situations. In the LAN-WLAN experiment the handoff was triggered by simply unplugging the Ethernet cable from the MN whereas, in the WLAN-UMTS experiment, the MN was moved away from the access point until the signal was lost. The LAN and WLAN load was maintained at a moderate load during the experiments. We had no information about the status of the publicly operated UMTS network. The handoff delay and packet loss due to the handoffs are comparable on the VoIP end-points. The differences are caused by the asymmetry of up-link and down-link bit-rate in the UMTS network.

Notice that, due to the difference of packet trip time along the two networks, the packet loss is not proportional to the handoff delay. In the WLAN-UMTS case with MM (a make before break handoff) there is a delay also when no packet is lost. The difference of packet trip times is not constant across the experiments, and its value is about 200 ms from the MN to the CN and 100 ms from the CN to the MN. Packet loss and handoff delay are proportional only for *break before make* handoffs between networks with comparable packet trip times.

[2]The minimum interval for RA is 50 ms. Most of the handoff delay, however, is due to the NUD procedure.

TABLE II
PERFORMANCE FOR THE UP HANDOFF AMONG DIFFERENT NETWORKS

	LAN-WLAN (MN)		WLAN-UMTS (MN)		WLAN-UMTS (CN)	
	Delay (ms)	Loss (#)	Delay (ms)	Loss (#)	Delay (ms)	Loss (#)
L3 trigger, 25 packets/s	402	10.0	351	5.0	558	3.0
L3 trigger, 50 packets/s	332	16.6	426	14.3	340	13.7
MM, 25 packets/s	96	1.3	153	0.0	262	2.0
MM, 50 packets/s	89	3.6	167	0.0	234	4.7

TABLE III
AVERAGE PACKET LOSS FOR 802.11 - UMTS MOBILITY (4 HANDOFFS)

	Average packet loss (%)
L3 triggering, 25 packets/s	$13.95 \pm 9.40\%$
L3 triggering, 50 packets/s	$13.56 \pm 4.96\%$
MM, 25 packets/s	$4.52 \pm 1.31\%$
MM, 50 packets/s	$2.38 \pm 1.43\%$

These conditions hold in the case of the LAN-WLAN handoffs of Table II and in the case of horizontal handoffs.

The results show how the MM module achieves a better handoff performance, especially for the communication from the CN to the MN. This improvement becomes even more impressive looking at the overall loss rate during the communication. We performed 15 experiments with VoIP conversations of about 180 s with 4 handoffs each (2 up and 2 down) at different hours of the day. Results are reported in Table III. The experiments were performed outdoor and each pair (with and without the MM) was measured under the same conditions. The better performance with the MM stems from several factors: *i)* up handoffs are faster; *ii)* the handoff is executed before the connectivity is lost; *iii)* the MM decreases the degradation of data flows due to zones with intermittent connectivity; *iv)* ping-pong effects can be avoided. These results lead us to conclude that, in practice, the most relevant factor for an efficient heterogeneous mobility is a smart management of wireless links, rather than the optimization of handoff delay.

IV. ADAPTATION AT SESSION LAYER

A VoIP application is made of, essentially, two parts. The first one manages the call, using a signaling protocol for session establishment like SIP, whereas the second one controls the incoming and outgoing media flows. Note that in this paper we use the term *session* to refer to sessions used by applications (in our case, by VoIP applications), not to the OSI session layer.

The signaling protocol is used by the VoIP application to determine the session content (codec, audio/video flows, etc.). One of the users (the offerer) generates an SDP (Session Description Protocol) session description block and sends it to the remote user (the answerer) who then generates a new session description and sends it to the offerer. The session description contains many attributes, among which, the IP address used by each side and the supported audio/video codecs for the session. After the offer/answer exchange, both users have a common view of the session to be established. They, at least, know the formats they can use and the transport addresses for the session.

A. Content adaptation

An audio flow is composed of packets with fixed size obtained by sampling the audio source at a predefined frequency (possible values are provided in [21]). The most common audio codecs used for IP telephony are G.711 Alaw/Ulaw (64 kbps), GSM (13 kbps) and G.729 (8 kbps) [22]. The actual bit-rate generated or received by the application is larger than the nominal bit-rate of the codec, as each packet must contain the protocol headers, whose overhead may have a significant impact.

An access network handoff usually implies a change in the available bit-rate. A mobility-aware VoIP application can react by adapting the session content in order to fulfill the new bit-rate constraints in two ways: *i)* by changing the session; *ii)* by changing the audio codec. Consider for example a video conference with a mobile device connected through a WLAN: when the mobile device must move to a 3G network the video part could be dropped to reduce the required bit-rate, and resumed later when the WLAN becomes again available. Similarly, an audio call using the G.711 codec could switch to a GSM codec after an up handoff from a WLAN to a UMTS network, as the up-link bandwidth for current UMTS network is only 64 kbps.

Session adaptation can be performed through the standard signaling protocol as long as the application is informed by the MM about network changes and available bit-rate. The agent which takes the initiative must issue a new SIP INVITE message to the other end of the conversation, specifying new session parameters. This request can contain a new codec specification and/or sub-session content.

To predict the bit-rate needed by the VoIP application the following parameters are used:

- f is the number of samples per second; the typical value is $f = 8000$.
- n is the number of samples contained in each IP packet (n samples make an *audio frame*);
- λ is the number of network packets per second, $\lambda = f/n$;
- b_0 is the bit-rate of the codec;
- h is the total size of the headers (in bytes). Each packet contains at least the RTP, UDP, IP and data link header, that gives (for IPv6 and 802.11) $h = 74$ bytes. Additional overheads may be due to the physical layer header and

to IPv6 tunneling (used when the CN is not MobileIPv6-enabled), VPN, etc.
- s is the packet size (in bytes), given by $s = h + \frac{b_0}{8f}n$; the payload size is $\frac{b_0}{8f}n$.
- b is the required bit-rate (in bps), $b = b_0 + \frac{8fh}{n}$.

The b value refers to each of the incoming/outgoing flows. To highlight the impact of the header overhead, consider a typical application using GSM codec with $f = 8000$, and $n = 160$ (meaning 50 packets per second). We have 42.6 kbps, instead of the nominal 13 kbps. This is due to the fact that the payload in this case is just 33 bytes. The application may reduce the header overhead by increasing the value for n. With $n = 480$ (each packet contains 480 samples) the bit-rate is reduced to approx. 23 kbps. This change requires no explicit signaling at session level. However, it can be performed only on the outgoing flow and then it can only help reducing the up-link bit-rate of the mobile device. A second drawback is that the loss of one packet implies in this case the loss of 60 ms of conversation, and that larger packets are more subject to transmission errors and corresponding delays. A more promising approach is to resort to header compression techniques [23], that have been widely adopted by various standardization bodies including the 3GPP and 3GPP2, as well as in CDMA 2000 Release B. In this case, the IP/UDP/RTP header is reduced to just one byte, and the computation of the actual bit-rate must be changed accordingly.

Session content adaptation is performed by the MN through the following steps:

- The VoIP application registers itself with the MM module through the *prehandoff_notification*, *handoff_notification* and *warning_notification* functions; this is done by specifying three functions to be invoked when the corresponding events occur.
- When a warning or a handoff decision is notified, the application uses information about the new network bit-rate and the available codecs to decide *whether* and *how* to adapt the session content;
- If session adaptation is necessary and the handoff is toward a network with a lower bit-rate, the application issues immediately an INVITE SIP message to the remote agent, specifying the new session parameters. The control is returned to the MM, that executes the handoff.
- If session adaptation is necessary but the handoff is toward a network with a higher bit-rate, the application waits for the *handoff_notification* before actually performing the signaling steps.

Note the use of the *prehandoff_notification* to trigger the session update before the handoff is actually performed. If the network handoff happened before the session update, the new access link could be congested (especially in case of an up handoff) and the session signaling could be lost or delayed. Moreover, the use of the MM guarantees that, whenever possible, the handoff is notified when the old network is still reachable and can be used for session signaling.

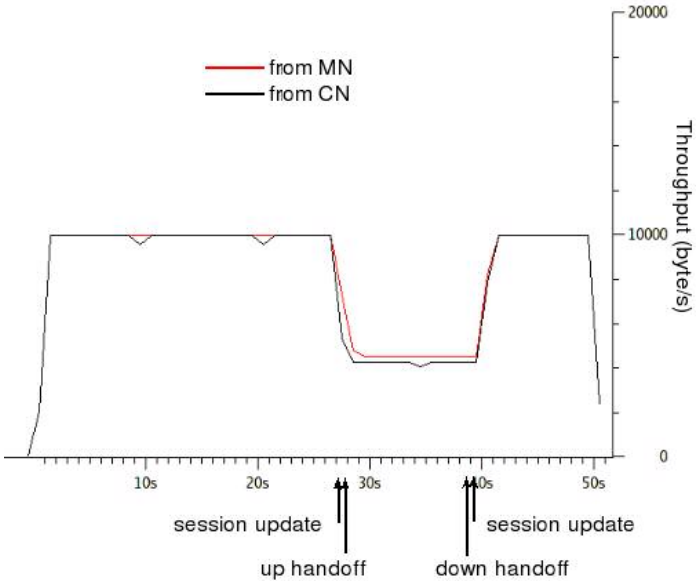

Fig. 2. Session content adaptation across two handoffs

B. Experimental results

Fig. 2 illustrates the resulting bit-rate in a real WLAN-UMTS-WLAN VoIP session with session adaptation and no header compression. The VoIP session initially uses a G.711 Alaw codec ($b_0 = 64$ kbps). We have $f = 8000$ samples per second, $n = 200$ samples per IP packet, and thus $\lambda = 40$ packets per second. The header size is $h = 76$ bytes (the sum of the RTP, UDP, IPv6 and L2 headers), the packet size is $s = 320$ bytes, the required bit-rate is $b = 9900$ bytes per second (or 79200 bps), in both directions. The overall bit-rate requirement is double, 158 kbps. This value is too large for the UMTS network (at least in up-link), so it is necessary to perform a session update. Both communicating peers have a GSM codec at $b_0 = 13$ kbps and the MN sends, just before the up handoff, a SIP message to switch the codec. As a result, since the size of the header becomes equal to 100 bytes (the additional 24 bytes contain Mobile IPv6 special options), the required bit-rate is $b = 4125$ bytes (or 33000 bps). At about 40 s the UMTS-WLAN down handoff takes place, and the session content is immediately restored with the G.711 codec. Here we do not consider the impact of additional headers required by IPv6-in-IPv4 tunneling (if the UMTS network does not support IPv6) or VPNs. As a final remark, we highlight that this kind of content adaptation is an option to trade off conversation quality against usage of network resources. When resources are scarce it can be exploited to maintain active a connection that otherwise would be interrupted. When, conversely, adaptation to a lower bit-rate is not mandatory, it can be used to implement users' policies for allocating dynamically network resources between applications.

V. Adaptation at Application Layer

At the application layer, the different jitter and packet trip times of two network technologies have more impact than the packet loss or duration of the handoff between the

two networks. To clarify this point, we briefly recall the conceptual structure of a VoIP application. At one end of the communication, the remote peer sends a stream of fixed size packets at a given pace of λ packets/s. The stream is received after an average delay d due to the packet trip time. For several reasons (*e.g.*, *jitter* effects), packets may arrive after their playout schedule, or out of order, and be dropped. In order to mitigate such problem, the receiver side generally introduces a *jitter buffer* (also called *playout buffer*) between the arrival of a packet and its playout that causes a *playout delay*.

Since there is a trade-off between buffering delay and the resulting late-packet loss, a good playout algorithm should be able to keep the buffering delay as short as possible while minimizing the number of packets dropped as a consequence of late arrival. Although there are methods that use a fixed size for the jitter buffer throughout the duration of an audio call [24], recent research employs adaptive methods to vary playout delay during a call's lifetime [7].

Adaptive playout algorithms can be grouped into three categories [7] according to the method used to select the optimal playout delay:

- algorithms that continuously estimate network delay and jitter (*reactive algorithms*) [25];
- algorithms that maintain a histogram of packet delays (*histogram algorithms*) [26];
- algorithms that monitor the packet-loss ratio or buffer occupancy;

The cross-layer approach described in Section III can provide information about the expected network behavior. This information can be used by the adaptive playout algorithm of the VoIP application. Specifically, handoff notifications and information about network delay and jitter can be used to proactively change the size of the jitter buffer and playout delay and optimize the loss/delay performance of the application.

A. Delay across handoffs

Large variations in network delay d, and jitter are likely to be observed when roaming across heterogeneous networks. In order to illustrate the problem, we consider a VoIP application with a fixed jitter buffer. A jitter buffer introduces a delay in the voice playout $d_b = S/\lambda$ where S is the size of the buffer measured as the number of audio frames that the buffer can contain. Hence, the total delay (between packet sending and packet playout) becomes not less than $D = d + d_b$.

After the handoff toward a network with a larger d, typically an up handoff, the increased delay might cause a massive packet discarding, as illustrated in Fig. 3. If d' is the new network delay, all subsequent packets such that $d' - d > d_b$ will be discarded, since their playout time has already expired, due to the larger d'. This situation is pretty common, since the difference in packet trip time between a WLAN and a UMTS access network can easily exceed 100 ms, a typical value for d_b.

Fig. 4 (left) shows an experimental plot of $d' - d$ during a handoff from a WLAN toward an UMTS network that happens

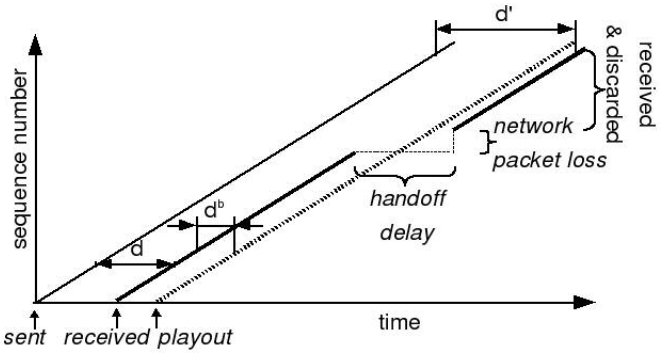

Fig. 3. Packets discarding after a handoff toward a network with a larger trip delay

at $t \approx 123.5$ s. Here we estimated $d' - d$ by measuring the delay of subsequent packets with respect to the first one. In this case, the MN receives a stream encoded by a GSM codec with $\lambda = 50$, $n = 160$, and the commercial UMTS network does not provide QoS support. We notice that when the network reaches its steady state, the additional delay with respect to the WLAN is only about 100 ms, but there is a transient peak, lasting about 5 s, in which the delay rises above 1 s. The peak lasts until a dedicated wireless link on the UMTS network, that was idle before the handoff, is set up. When the set-up is complete, at $t \approx 129.5$ s, the delay decreases, and there is an accelerated delivery of late packets. Hopefully, the introduction of QoS in UMTS networks will improve this situation. A traffic class with a bounded maximum packet delay should not exhibit this behavior. Nevertheless, it is probable that in transient handoff situations like the one depicted in Fig. 4 there will be a burst of packets with larger delays.

Handoffs toward a faster network produce a smaller value for d. This case is shown in Fig. 4 (right), where the MN, that receives a stream encoded by GSM codec with $\lambda = 50$, performs a soft handoff from a UMTS network to a WLAN at $t \approx 187.5$ s. During the handoff there is a sequence of out-of-order packets since the MN receives them from both interfaces until all packets traveling along the UMTS link arrive (this phase corresponds to the up and down bouncing part of the delay plot). No packet is lost.

Notice the difference in jitter between the UMTS network and the WLAN. This kind of handoff has no particular impact on the VoIP application provided that: *i)* the application is able to perform packet reordering; *ii)* the buffer size at the receiving side is sufficient to contain ≈ 110 ms of conversation arriving earlier than expected. The application can then reduce the jitter buffer after the handoff (there is less jitter), thus reducing the end-to-end delay.

B. Buffer management for VoIP applications

In order to gain full insight into mobility situations, we developed our own VoIP application that implements several strategies for jitter buffer management and packet dropping. Our aim was also to determine whether the notifications of the Mobility Manager could help the VoIP application to devise

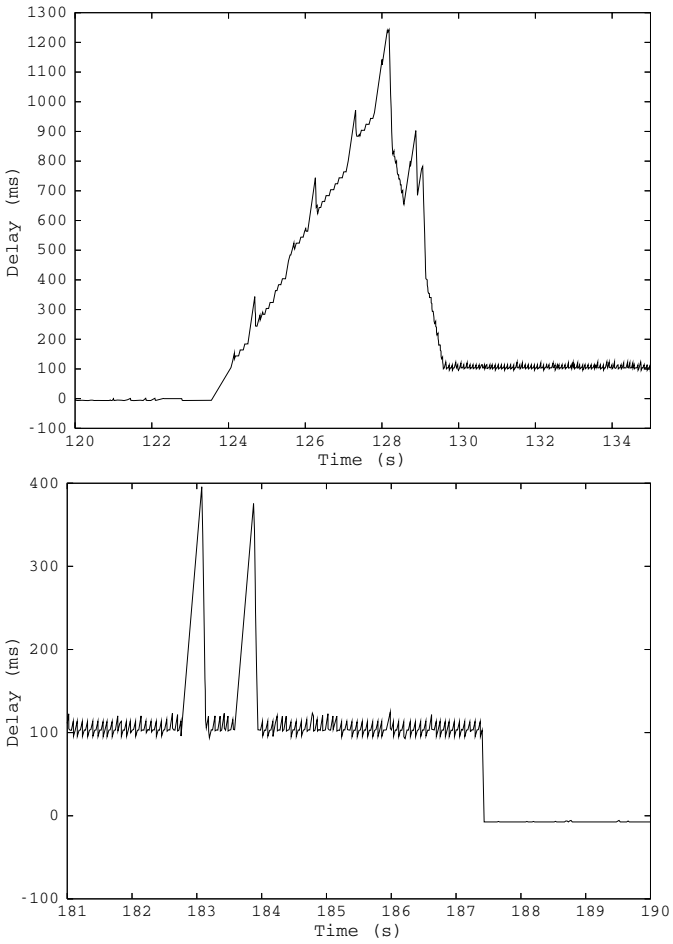

Fig. 4. Delay (with respect to the first packet) during a handoff toward (left) and from (right) UMTS

the best strategy in mobile situations. For this reason, our VoIP application is able to receive notifications from the MM about handoff and estimated delays on the destination network through the MM API. We do not describe all the details of our VoIP application. We add only that it fully supports the SIP as defined in RFC 3261 [9] and that we tested its interoperability with other SIP compliant VoIP applications.

At each endpoint the VoIP application uses a circular playout buffer B to store the audio samples arriving from the network. If b_0 is the codec bit-rate (see Section IV-A), each sample has a size of $b_0/8f$ bytes. Each network packet payload contains an audio frame of n audio samples and has a *timestamp* t_i indicating the position of the samples in the audio stream. The application reads from the buffer f samples per second, starting from the position 0 that corresponds to playout time $p = 0$. There is a correspondence between the playout time p and the location $B[k]$ in the buffer: $p = 8k/b_0$. In order to mitigate the jitter effect, we reproduce the audio samples with a delay d_b. To this purpose, the first packet is written in the buffer at the position $d_b b_0/8$, so it is played back at time $p_0 = d_b$. Each subsequent packet with timestamp t_i has playout time $p_i = p_{i-1} + t_i - t_{i-1}$. The situation is shown in Fig. 5. Notice that the index i indicates only the arrival order at the host: packets i qand $i+1$ do not necessarily have

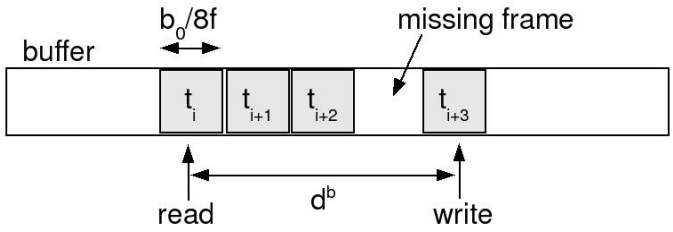

Fig. 5. Buffer management at the receiver side

consecutive timestamps. We indicate with p^* the timestamp corresponding to the current *read* position. Timely packets will be written in a buffer location corresponding to the time $p^* + d_b$. Packets with $p_i < p^*$ are discarded since their playout time has already expired.

C. Reactive algorithms

Standard reactive playout algorithms [25] are based on Jacobson's work [27] on TCP roundtrip time estimation. These algorithms estimate the network delay and its variance and use them to calculate the jitter buffer. Estimates have the form:

$$\hat{d}_i = (1-\alpha)n_i + \alpha\hat{d}_{i-1}$$

$$\hat{v}_i = (1-\alpha)|\hat{d}_i - n_i| + \alpha\hat{v}_{i-1}$$

where $\hat{d}_i$ and $\hat{v}_i$ are the *i*-th estimate for the average network delay and variance, respectively, n_i is the *i*-th packet delay, and α is a weighting factor in the range $[0..1]$ that determines how fast the adaptation is. Playout algorithms calculate the jitter buffer as $d_b = \beta\hat{v}_i$, where the parameter β controls the loss-delay trade-off. An upper limit for d_b is used to prevent end-to-end delay from growing beyond a tolerable value, denoted as D_{max}.

D. Algorithms monitoring packet-loss ratio and buffer occupancy

Reactive algorithms continuously update the value of d_b, whereas, with packet-loss and buffer occupancy methods, an application detects an increase in the network delay d by setting a threshold on the number of discarded packets. When an increase of d is detected (*deferring* phase), the last received packet, instead of being discarded, becomes a reference for the new playout schedule. To this purpose, its playout time p_i is set equal to $p^* + d_b$. Conversely, when d decreases, packets will have a playout time p_i much larger than p^*. This situation can be detected by setting a threshold on the number of consecutive packets with $p_i - p^* > T_p$, where T_p is a time threshold. When a decrease of d is detected (*anticipation* phase), it is possible to anticipate the playout schedule by skipping audio frames during a silence interval or scaling individual voice packets. Fig. 6 shows the anticipation phase after a handoff toward a faster network. Before the handoff, $d_b = 80$ms; after the handoff, packets arrive 110 ms earlier than before, and the delay between arrival and playout increases up to 190 ms. With a threshold for consecutive packets with $p_i - p^* > 150$ ms

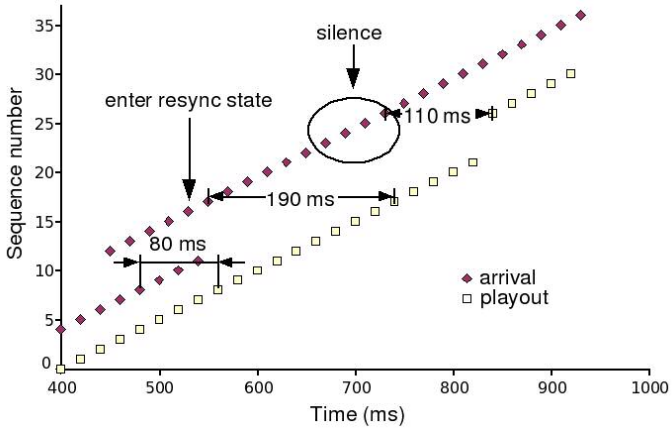

Fig. 6. Delay recovery through silence recognition and detection of anticipated packets

set equal to 5, this decrease in network delay is recognized at packet 16. Packets 23-26 are marked as "silence" packets, so they can be skipped. Playout time baseline is now packet 27, and $d_b = 110$ ms. During the anticipation phase the value of d_b depends on the duration of the "silence" interval and on its initial value[3]. These methods perform adaptation by changing the packet playout schedule baseline, and they do not directly change the initial value of d_b. Thus they adapt to changes of network delay better than to changes of network jitter.

E. Using the MM to detect network delay and jitter changes

A cross-layer approach can provide the information required to improve its behavior to the playout algorithm. If one resorts to fixed size buffering, the difference $d'-d+\beta v'$ between delay estimates can be used to adjust the jitter buffer size across the handoff. In the case of reactive algorithms, the handoff notification received from the MM can be used to change the α value. This improves the sensitivity of the estimates of the network delay and jitter in the new situation. In the case of packet-loss monitoring methods, the value $\beta\hat{v'}$ can be used as the new d_b value after each *pre_handoff* notification. The threshold on discarded packets is set equal to 1, and the playout baseline is shifted to the first packet on the new network. This is possible (when the values of d and d' are not too close) by looking at the inter-arrival times of packets and it automatically adapts the new playout schedule to the delay on the new network.

F. Experimental results

In order to validate our cross layer approach for adaptive playout, we employed standard and modified playout algorithms on traces of VoIP packets in mobility situations. Our aim here is to assess whether the information provided by the MM are useful in order to improve the behavior of representative playout algorithms, rather than to find out the best playout algorithm.

[3]Notice that it is common for the sender application to detect silence in order not to send "empty" packets, and this is generally more efficient (particularly when there are more than two participants in a session).

We collected 15 traces at the MN, each containing 180 s of conversation at $\lambda = 25$ (4500 packets) or $\lambda = 50$ (9000 packets). In each trace there are four handoffs, two from WLAN to UMTS and two from UMTS to WLAN. The CN was located few hops away from the WLAN AP, whereas the MN moved from the WLAN to the UMTS network. Note that packets from the UMTS network had to travel through a public IP network before reaching the CN, and this yields a larger difference in delays. The other experimental settings were the same as in Section IV-B.

We computed the late packet loss and average buffering delay of each trace according to the following procedures:

1) *Fixed buffer*. In this case a constant value for d_b is used, and the packet loss/delay ratio is plotted at different values of the d_b parameter.
2) *Monitoring*. The algorithm presented in Section V-D is used. The packet threshold for deferring and anticipation phases is set equal to 3, and T_p =60 ms respectively. The parameter is d_b.
3) *Reactive*. The basic reactive algorithm (without spike detection) described in [25] with $\alpha = 0.9$ is used. The parameter is β.
4) *Fixed+MM*. The handoff notification is used to set $d'_b = d' - d + \beta\hat{v'}$ after each handoff. The parameter is d'_b.
5) *Monitoring+MM*. The algorithm presented in Section V-D is used. Information from MM is used to set $d'_b = \beta\hat{v'}$ after each handoff. The parameter is d'_b.
6) *Reactive+MM*. We used a simple variant of the *dynamic* α algorithm [7], by setting $\alpha = 0.9$ for the WLAN access network and $\alpha = 0.8$ on the UMTS network. The parameter is β.

In each case, playout adaptation was performed on a *per-packet base*. Even if applications usually employ *per-talkspurt* adaptation, our choice makes the comparison simpler. Moreover, the method proposed in [28] can be used in case of *per-packet* adaptation without impairing audio quality.

The results obtained by the different playout algorithms are reported in Table IV and plotted in Fig. 7.

- For fixed-size algorithms, the MM can adapt the size of the jitter buffer to the network delay, resulting in a noticeable reduction of buffering delay at the same loss rate. This algorithm cannot adapt to spikes or delay variations that are not related to handoffs.
- Monitoring algorithms adapt to variations of delay but not of jitter. The fixed d_b results either too high in the WLAN phase or too low in the UMTS phase. The *Monitoring+MM* algorithm overcomes this drawback by changing d_b and shows better performance.
- Reactive algorithms enhanced by the MM can dynamically choose the value of α. The idea, first proposed in [7], is that the value of α is set low when the jitter is high, and vice versa. The information provided by the MM is used to this purpose. In our experiments, we used $\alpha = 0.9$ without the MM, and $\alpha_1 = 0.9$, $\alpha_2 = 0.8$ for the WLAN and UMTS phases respectively, when the MM is used.

Fig. 7 plots the data of Table IV together with the lines

TABLE IV
PERFORMANCE OF PLAYOUT ALGORITHMS WITH AND WITHOUT MM

Fixed	d_b	avg loss (%)	avg delay (ms)	Fixed+MM	d'_b	avg loss (%)	avg delay (ms)
	80	49.00	77.12		80	49.12	22.27
	120	19.13	77.76		160	6.44	39.27
	160	6.32	106.01		240	4.17	80.11
	200	5.18	144.46		280	3.06	100.52
	240	4.05	182.48		320	2.07	121.07
	280	2.95	220.13		400	0.75	163.05
Monitoring	d_b	avg loss (%)	avg delay (ms)	Monitoring+MM	d'_b	avg loss (%)	avg delay (ms)
	20	5.93	32.87		20	5.91	33.26
	40	3.40	51.63		40	3.35	42.72
	80	3.20	91.27		80	2.91	63.03
	120	2.75	128.53		120	2.62	80.72
	160	2.08	163.58		160	2.21	100.48
	200	1.54	201.23		240	0.98	141.32
	240	1.00	235.87		280	0.54	159.23
Reactive	β	avg loss (%)	avg delay (ms)	Reactive+MM	β	avg loss (%)	avg delay (ms)
	1	8.47	31.30		1	6.78	31.93
	2	3.97	45.80		2	3.47	43.63
	3	2.41	60.30		3	1.93	55.00
	4	1.84	76.01		4	1.46	67.54

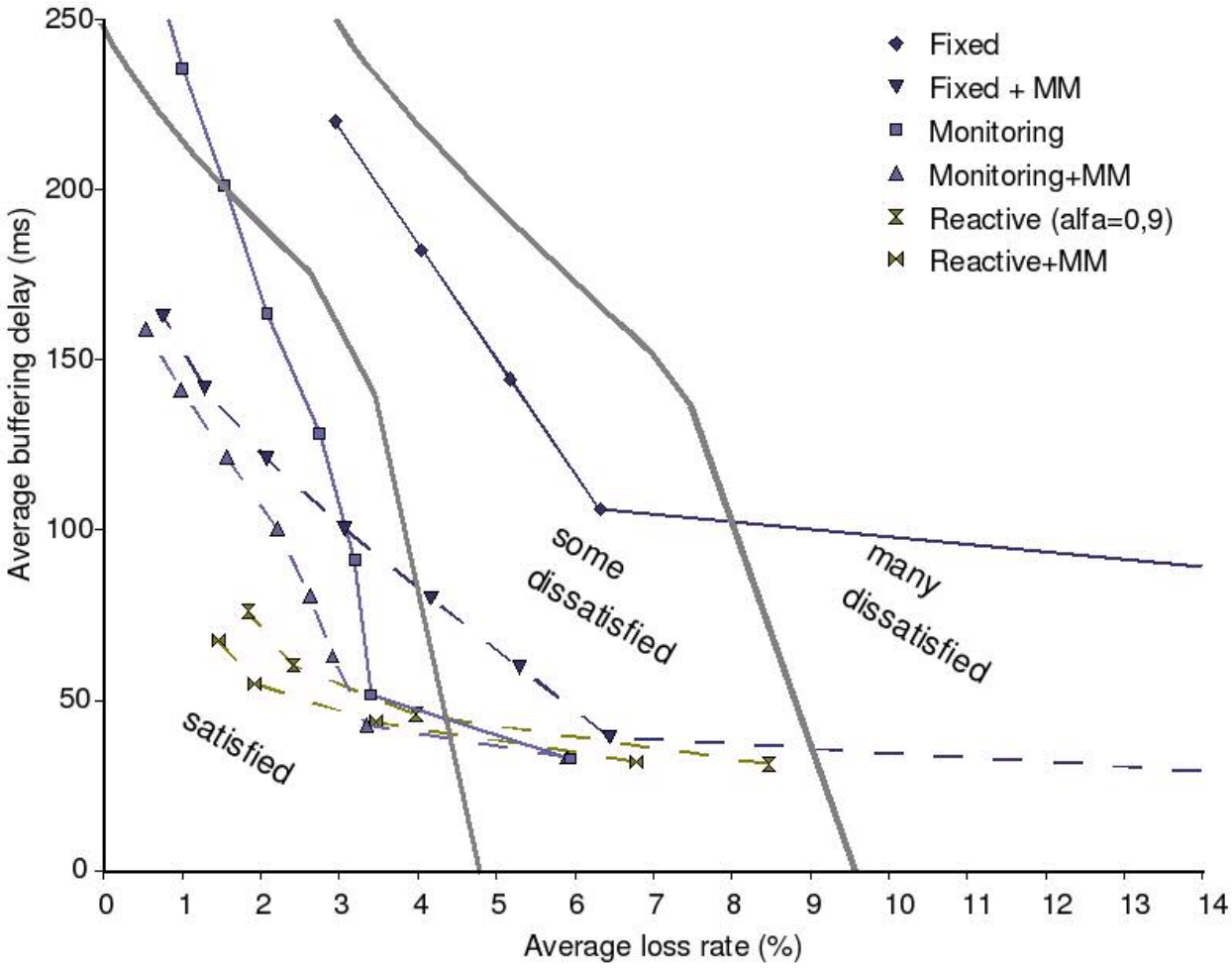

Fig. 7. Packet-loss/delay ratio and user satisfaction for adaptive algorithms

that in the STSTQA method [7] mark different levels of user satisfaction, for the G.711 encoding with packet loss concealment (PLC), and echo cancellation implemented (TELR=65 dB). The STSTQA method provides the rating factor R of the ITU-T *E-model* [29] as a function of mouth-to-ear delay and loss ratio. The R factor can be translated in terms of speech-transmission quality and corresponding user satisfaction [30]. The lines in Fig. 7 are the border between the region where $R > 90$ (users satisfied or very satisfied) with the region where $R > 70$ (some users dissatisfied), and between the latter and the region where $R < 70$ (where in the best case there are many dissatisfied users). In all cases the use of the MM improves the performance of playout algorithms.

The STSTQA method is based on mouth-to-ear delay, whereas the plots refer only to the average buffering delay on the receiving side. Thus, the vertical distance between the plot and the satisfaction lines is the maximum delay compatible with the corresponding level of satisfaction of all the remaining sources of delay (sampling, buffering, etc.). The most relevant contribution comes from network delay. This plot can indicate the maximum tolerable network delay for different playout algorithms.

To have a more direct evaluation of the benefits provided by resort to the MM at network and application level, we performed subjective listening tests. The listening tests were carried out according to ITU-T Recommendation P.800 [31]. Each speech sample, lasting for about 10 seconds, was recorded by male and female speakers under two network situations: a WLAN-UMTS and a UMTS-WLAN handoff, both occuring in the middle of the sample. Results are summarized in Table V and report the average score from all the 20 listeners for all samples. Note that in the Mean Opinion Score (MOS) Rating Scale, a rating of 5 corresponds to *excellent* sound quality, 4 to *good*, 3 to *fair*, 2 to *poor* and 1 to *bad* (neither party can communicate effectively). In the first row we used network traces (and corresponding loss) for handoffs with network layer triggering and without the MM. In this case, network losses are the predominant cause for the poor quality. In the second case, we used a standard monitoring playout algorithm with a jitter buffer of 80 ms on network traces obtained through the use of the MM. In this case, there is a substantial improvement for packet loss, but there are many smaller interruptions, due to packets discarded at the application level. The situation improves in the third set of samples, where the same network traces are elaborated by means of a monitoring algorithm that receives notifications from the MM in order to change the jitter buffer. In this case, since the MOS score does not consider the voice delay, we introduced the constraint that the average buffering delay along the whole trace must be the same as in the second case, resulting in jitter buffer sizes of 40 and 120 ms. As a last remark, we highlight the fact that the basic adaptive algorithms used in this test can be improved, for example by adding spike-recognition capabilities, thus yielding better MOS scores.

G. Comparison with related work

We have limited our comparison to a few representative approaches for playout buffering. However, an overview of recent proposals in this area suggests that the approach based on the MM can improve the performance of playout algorithms in most situations.

A drawback of reactive algorithms is the difficulty of obtaining precise values for n_i. To this end, either methods using precise time synchronization or additional algorithms using RTT values are required. In mobility situations where the network behavior changes in time, the MM can help to choose a smaller value for α to improve the sensitivity of the algorithm to delay and jitter variations. Authors of [7] propose a *dynamic* α algorithm to derive the α value through a function empirically chosen from traffic traces on wired networks, and show that this algorithm outperforms other reactive algorithms and histogram-based algorithms. A drawback is that the choice of the α function cannot be done automatically. Experimental results with the *dynamic* α algorithm show packet loss ratio between 0.5 and 4% with buffering delays of 35-20 ms [7]. However, these results have been obtained on wired geographical networks, where the impact of delay variations and jitter is much more limited. As described in the previous section, the use of the MM allows to achieve similar results in mobile situations.

An important feature of adaptive playout algorithms is their ability to react to *spikes*. A spike is a sudden, large increase in the end-to-end network delay, followed by a series of packets arriving almost simultaneously, caused by network congestion. In [32] a reactive algorithm is presented which improves the performance in mobile situations. This method aims at recognizing *handoff phases* and using a faster adaptation method during these phases. The advantage of using the MM in this context is that the existence of a handoff phase is explicitly signaled from lower layers.

Clearly, a precise estimate of the current network delay is a key factor for an effective adaptive playout algorithm. Authors of [33] propose to use session set-up messages to send information about network delays, that should be stored and updated at inner nodes of the networks. In this case, the delay value is used to compute a fixed playout delay throughout the duration of a call (*per-call* approach). However, per-call approaches are not suited to mobile situations, particularly on heterogeneous networks where the network delay can change dramatically during a call's lifetime.

A method that resorts to time synchronization is presented in [34] to combine fixed and adaptive buffer strategies. This method relies on GPS receivers and NTP servers to provide precise synchronization to the communicating end-points. Time synchronization can be used together with the MM in the following way: precise measurements for delay and jitter are carried out by a module that performs active probes on the wireless interfaces with the time synchronization method. When the MM notifies the handoff, the new values are used for playout scheduling.

VI. Conclusion

On heterogeneous networks that do not provide a uniform QoS management, a VoIP application needs the ability to adapt to changes of the network features. In this scenario it is not

TABLE V
COMPARISON OF MEAN OPINION SCORE RATINGS (5=EXCELLENT, 1=BAD)

	handoff WLAN-UMTS	handoff UMTS-WLAN
L3 handoff, adaptative playout algorithm, $d_b = 80$ ms	1.85±0.34	1.67±0.62
MM-assisted handoff, $d_b = 80$ ms	2.29±0.33	2.38±0.38
MM-assisted handoff and adaptation, $d_b = 40$ms $d'_b = 120$ms	3.17±0.25	3.00±0.48

sufficient to devise methods for low-latency handoffs, since a smart management of wireless links and adaptation to the bit-rate, delay and jitter of the networks in use may have a more dramatic impact on the behavior of a VoIP application than the handoff delay. We have shown how to introduce this ability in VoIP applications through a combination of mechanisms at network and application layer. In particular, our cross-layer approach relies on the presence of a Mobility Manager module on the mobile device to provide information about network parameters and handoff events. This solution proved to be effective for tuning parameters concerning audio stream playout and buffer management. Other advantages of this approach are that *i)* it does not require changes to existing network infrastructures; *ii)* it relies on existing protocols only. Moreover, it can be exploited for supporting other classes, *e.g.* interactive and streaming, of network applications. Methods and results presented in our work may be extended, with slight modifications, to other real-time applications, like video-conferencing, network gaming, *etc.*, that depend on network characteristics in a similar way.

ACKNOWLEDGMENT

We thank Dr. Luisa Santoro for having carefully read the manuscript. The authors are grateful to the anonymous referees for their valuable comments which greatly helped in improving the first versions of this paper. This work was funded by the Ministero dell'Istruzione, dell'Università e della Ricerca (MIUR) with the FIRB project WEB-MINDS and the QUASAR project (PRIN 2004 funding program), and by the 6th Framework Programme, Information Society Technologies, under Contracts no. FP6-IST-038423 (CONTENT) and no. FP6-IST-033516 (NETQOS).

REFERENCES

[1] 3GPP, "Feasibility study on 3GPP system to Wireless Local area Network (WLAN) Internetworking". Tech. rep. TR 22.934, v. 6.2.0, September 2003.
[2] 3GPP, "IP Multimedia Subsytem (IMS); Stage 2". Tech. rep. TS 23.228, March 2006.
[3] H. Fathi, R. Prasad, and S. Chakraborty, "Mobility Management for VoIP in 3G Systems: Evaluation of Low-Latency Handoff Schemes", IEEE Wireless Communications Magazine, vol. 12, n. 2, April 2005, pp. 96-104.
[4] M. Bernaschi, F. Cacace, G. Iannello, A. Pescapè and S. Za, "Seamless Internetwoking of WLANs and Cellular Networks: architecture and performance issues in a MobileIPv6 scenario", *IEEE Wireless Communications Magazine*, vol. 12 n. 3, June 2005, pp. 73-80.
[5] H. Yokota et al., "Link Layer Assisted IP Fast Handoff Method over Wireless LAN Networks", in *Proc. ACM MobiCom 02*, pp. 131-39.
[6] M. Bernaschi, F. Cacace, R. Clementelli, and L. Vollero, "Adaptive Streaming On Heterogeneous Networks", in *Proc. 1st ACM Workshop on Wireless Multimedia Networking and Performance Modeling (WMuNeP 2005)*, Montreal, October 2005, pp. 16-23.
[7] M. Narbutt, A. Kelly, L. Murphy, and P. Perry, "Adaptive VoIP Playout Scheduling: Assessing User Satisfaction", *IEEE Internet Computing,* July 2005, pp. 28-34.
[8] G. Camarillo, and M. A. Garcia Martin, *The 3G IP Multimedia Subsystem (IMS)*, Wiley, 2005.
[9] J. Rosenberg et al., "SIP: Session Initiation Protocol", IETF RFC 3261, June 2002.
[10] F. Galan Marquez, M. Gomez Rodriguez, T. Robles Valladares, T. de Miguel, and L. A. Galindo, "Internetworking of IP Multimedia Core Networks between 3GPP and WLAN", *IEEE Wireless Communications Magazine*, vol. 12, n. 3, June 2005, pp. 58-65.
[11] D. Johnson, C. Perkins, and J. Arkko, "Mobility Support in IPv6", RFC 3775, IETF, June 2004.
[12] S. J. Koh et al. "Architecture of Mobile SCTP for IP Mobility Support", IETF Internet Draft, June 2003.
[13] E. Wedlund, H. Schulzrinne, "Mobility Support using SIP", in *Proceedings Second ACM/IEEE International Workshop on Wireless and Mobile Multimedia WoWMoM '99*, pp. 76-82, August 1999.
[14] T. T. Kwon, M. Gerla, and S. Das, "Mobility Management for VoIP Service: Mobile IP vs. SIP", *IEEE Wireless Communications Magazine*, vol. 9 n. 5, October 2002, pp. 66-75.
[15] W. Wu, N. Banerjee, K. Basu, and S. K. Das, "SIP-Based Vertical Handoff Between WWANs and WLANs", *IEEE Wireless Communications Magazine*, vol. 12 n. 3, June 2005, pp. 66-72.
[16] N. Montavont, and T. Noel, "Analysis and Evaluation of Mobile IPv6 Handovers over Wireless LAN", *Mobile Networks and Applications,* vol. 8, n. 6, 2003, pp. 643-653.
[17] R. Hsieh, and A. Seneviratne, "A comparison of Mechanisms for Improving Mobile IP Handoff Latency for End-to-End TCP", in *Proc. ACM MobiCom 03*, pp. 29-41.
[18] X. Zhao, C. Castelluccia, and M. Baker, "Flexible Network Support for Mobility", in *Proc. ACM MobiCom 98*, pp. 145-156.
[19] M. Bernaschi, F. Cacace, and G. Iannello, "Vertical Handoff Performance in Heterogeneous Networks", in *Proc. International Workshop On Mobile and Wireless Networking (MWN 04),* Montreal, August 2004, pp. 100-107.
[20] Q. Song, and A. Jamalipour, "Network Selection in an Integrated Wireless LAN and UMTS Environment Using Mathematical Modeling and Computing Techniques", *IEEE Wireless Communications Magazine*, vol. 12 n. 3, June 2005, pp. 42-48.
[21] H. Schulzrinne, "RTP Profile for Audio and Video Conferences with Minimal Control", IETF RFC 1890, January 1996.
[22] ITU-T Technical Recommendation G.711 (1998), and Technical Recommendation G.729 (1996).
[23] C. Bormann et al., "RObust Header Compression (ROHC)", IETF RFC 3095, July 2001.
[24] Y.-J- Cho, and C.-K. Un, "Performance analysis of reconstruction algorithms for packet voice communications", *Computer Networks and ISN Systems*, vol. 26 n. 11, pp. 1385-1408, 1994.
[25] R. Ramjee, J. Kurose, D. Towsley, and H. Schulzrinne, "Adaptive playout mechanisms for packetized audio applications in wide-area networks", in *Proc. of the Conference on Computer Communiations (IEEE Infocom),* Toronto, June 1994, pp. 680-688.
[26] S. B. Moon, J. Kurose, and D. Towsley, "Packet audio playout delay adjustment: performance bounds and algorithms", ACM/Springer Multimedia Systems, vol. 6 n. 1, pp. 17-28.
[27] V. Jacobson, "Congestion avoidance and control", in *Proc. ACM SIGCOMM Conference*, Stanford, 1988.
[28] Y. J. Liang, N. Farber, and B. Girod, "Adaptive playout scheduling and loss concealment for voice communications over IP networks". *IEEE Transactions on Multimedia*, vol. 5 n.4, December 2003.
[29] *Recommendation ITU-T G.107, The E-Model, A computational Model for Use in Transmission Planning*, ITU March 2003.

[30] *Recommendation ITU-T G.109, Definition of Categories of Speech Transmission Quality*, ITU September 1999.
[31] *Recommendation ITU-T P.800, Methods for Subjective Determination of Transmission Quality*, ITU September 1996.
[32] M. Benaissa, V. Lecuire, and F. Lepage, "An algorithm for playout delay adjustment for interactive audio applications in mobile ad hoc networks", in *Proc. of the Seventh International Symposium on Computers and Communications (ISCC '02)*, Taormina, July 2002.
[33] Y. Jung, and J.W. Atwod, "Switching between Fixed and Call-Adaptive Playout: A Per-Call Playout Algorithm", *IEEE Internet Computing,* July 2005, pp. 22-27.
[34] H. Melvin, and L. Murphy, "Time Synchronization for VoIP Quality of Service", *IEEE Internet Computing*, May 2002, pp. 57-63.

Massimo Bernaschi graduated in physics in 1987 at "Tor Vergata" University in Rome. After that he joined the IBM European Center for Scientific and Engineering Computing (ECSEC) in Rome. He spent ten years with IBM working in the field of parallel and distributed computing. Currently he is with the Italian National Research Council (CNR) as chief technology officer of the Institute for Computing Applications. He is also an adjunct professor of Operating Systems in "La Sapienza" University in Rome.

Filippo Cacace graduated in Electronic Engineering at Politecnico di Milano in 1988 where he received a Ph.D. in Computer Science. His research interests include wireless and heterogeneous computer networks, network applications, database programming languages and logic programming. He is currently an Assistant Professor at the University Campus Bio-Medico in Rome.

Giulio Iannello graduated in Electronic Engineering at Politecnico di Milano in 1981 and received a Ph.D in Computer Science and Computer Engineering from the University of Napoli Federico II in 1987. Currently he is Full Professor of Computer Science and Computer Engineering at the University Campus Bio-Medico di Roma. His current research interests include wireless and high performance computer networks, multimedia data processing, biomedical data processing, design and analysis of parallel algorithms, performance evaluation of parallel and distributed systems. He has published over 90 journal and conference papers in these and related areas. He is member of several Program Commettees of international conferences. Dr. Iannello is a member of the IEEE and ACM.

Massimo Vellucci graduated in computer science in 2006 from La Sapienza University in Rome. Currently he is with the Research Group on Computer Networks and Advanced Distributed Applications of University Campus Bio-Medico in Rome. His research interests include VoIP, wireless and mobile networks.